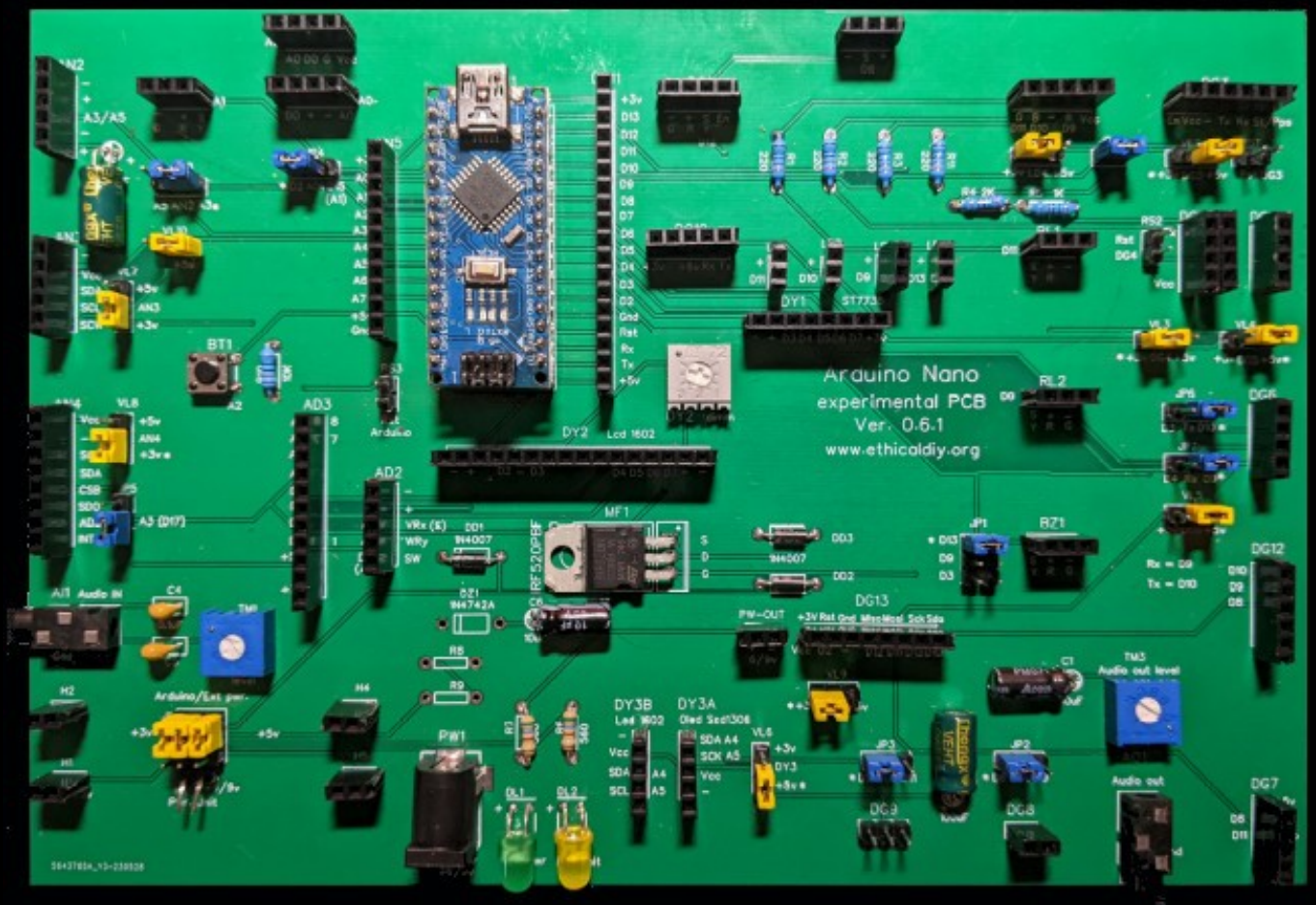
Una basetta sperimentale per Arduino Nano

di Riccardo Grosso



Come sperimentare centinaia di progetti in pochi minuti e senza problemi con il più piccolo microcontroller della famiglia di Arduino

Arduino permette di sperimentare una enorme quantità di progetti, di interagire in modo semplice con la realtà esterna, e non solo in un ambito circoscritto, come nel caso di un personal computer.
Ma spesso richiede di usare diversi moduli e sensori, che si connettono al microcontroller con una miriade di fili.
Il progetto di questa basetta sperimentale e didattica rende facile il collegamento tra decine dei più comuni moduli di Arduino praticamente senza cablaggi, in pochi minuti.
Inoltre si troveranno oltre un centinaio di programmi per testare le funzionalità della basetta sperimentale, costruendo tra l'altro una stazione meteorologica, un antifurto, una chiave elettronica, un rilevatore GPS e tanti altri...

Una basetta sperimentale per Arduino Nano

di Riccardo Grosso

Versione 0.6.1, luglio 2023

Per informazioni e commenti: info@ethicaldiy.org
Sito: https://www.ethicaldiy.org

Indice

Lo zoccolo e i programmi per AD2

Il joystick – KY-023 [2]
Moduli nativi per AN1 che possono essere connessi su AD2
Moduli nativi per AN2 che possono essere connessi su AD2
Moduli nativi per AD1a e AD1b che possono essere connessi su AD2
Rivelatori di gas: da MQ2 a MQ135 [1]

Lo zoccolo e i programmi per AD3

Programmi per Keypad 4x4 [3]
Programmi per Keypad 4x3 [1]

Gli zoccoli per moduli che usano porte analogiche:

Lo zoccolo e i programmi per AN1

Sensore DH11 – temperatura e umidità [1]
Sensore KY-039 – pulsazioni cardiache [1]
Sensore KY-018, fotoresistenza [3]
Potenziometri [2]
Sensore Sen18 (liquidi) [3]
Sensore KY-013 (temperatura analogica) [2]
Switch generici (KY-002, KY-004, KY-010, KY-017, KY-020, KY-021, KY-031, KY-035) [1]

Lo zoccolo e i programmi per AN2

Potenziometri [4]
Sensore PIR KY-007 HC-SR501 [1]
Voltmetro digitale

Lo zoccolo e i programmi per AN3

Scanner I2C [1]
modulo DS1307 (clock) [3]
display LCD 1602 + adattatore I2C [3]

Lo zoccolo e i programmi per AN4

Scanner I2C [1]
Modulo BMP280 – stazione metereologica [5]
Modulo GY-521 (accelerometro – giroscopio) [3]
Sensore Max30102 (pulsazioni/ossigenazione del sangue) [4]
Modulo GY-302 (sensore di luce) [3]
Modulo GPS NEO-7M [1]

Lo zoccolo e i programmi per AN5

Sintetizzatore sonoro analogico [2]

Ingresso Audio AI1 e relativi programmi

Programmi per l'ingresso audio SSd1306 128 x 32 [2]
Programmi per l'ingresso audio SSd1306 128 x 64 [2]

Pulsante BT1 e relativi programmi

Programmi per pulsante BT1 [3]

Gli zoccoli collegati alle porte digitali:

- **Lo zoccolo e i programmi per DG1**
  - Programmi per KY-032 Obstacle Avoiance module [2]
  - Programmi per KY-004 Button [3]
  - Programmi per DHT11 Temperature & humidity sensor [4]
  - Programmi per DS18B20 Digital temperature sensor [1]
  - Programmi per KY-008 Laser led module [1]
  - Programmi per KY-022 Infrared receiver module [4]
  - Programmi per KY-010 Photo interrupt module [2]
  - Programmi per KY-019 Relay [1]
  - Programmi per KY-002, KY-017, KY-020, KY-031 Mercury/Shock/Tap/Tilt [2]
  - Programmi per KY-033 Tracking module [3]
  - Programmi per KY-021 e KY-035 Reed switch e Hall
  - Programmi per il trasmettitore a 433 MHz [6]

- **Lo zoccolo e i programmi per DG2**
  - Programmi per HC-SR501 PIR sensor module [3]
  - Programmi per il laser detector module [1]

- **Lo zoccolo e i programmi per DG3**
  - Programmi per HC-06 bluetooth module [2]
  - Programmi per il modulo GPS NEO-7M [1]

- **Lo zoccolo e i programmi per DG4 - DG5**
  - Programmi per ESP8266 – Wi-Fi module [3]

- **Lo zoccolo e i programmi per DG6**
  - Programmi per Neo6MV2 – GPS [4]

- **Lo zoccolo e i programmi per DG7**
  - Programmi per sensore a ultrasuoni HC-SR04 [3]
  - Programmi per cella di carico + HX711 [3]
  - Programmi per il ricevitore a 433 MHz [5]

- **Lo zoccolo e i programmi per DG8**
  - Programmi per motore elettrico generico [3]

- **Lo zoccolo e i programmi per DG9**
  - Programmi per servomotore MG90S [2]

- **Lo zoccolo e i programmi per DG10**

- **Lo zoccolo e i programmi per DG11**
  - Programmi per display 5641AS + IC 7HC595 [1]
  - Programmi per rotary encoder KY-40 [3]
  - Programmi per tastiera a 16 tasti [1]
  - Programmi per lettore di SD [3]
  - Programmi per ricevitore a 433 Mhz [1]
  - Programmi per uno stepper motor [3]
  - Programmi per Keypad + Multiplexer [3]

Lo zoccolo e i programmi per DG12

programmi per DS1302 (RTC clock) [1]

Lo zoccolo e i programmi per DG13

Programmi per RFID module RC522 [6]

Lo zoccolo e i programmi per BZ1

Programmi per KY-006 e KY- 012 Buzzer [6]
Programmi per IR trasmitter KY-005 [1]

Lo zoccolo e i programmi per RL1

Programmi per il relay KY-019 (su RL1) [2]

Lo zoccolo e i programmi per RL2

Programmi per il relay KY-019 (su RL2) [2]

Gli zoccoli e i programmi per per LD1/LD2/LD3/LD5

Programmi per Led vari [8]

Gli zoccoli e i programmi per per LD4

Programmi per Led vari [8]

I display:

Lo zoccolo e i programmi per DY1 (digitale)

Programmi per il display ST7735 – TFT 1,7" [2]
Programmi le Encoder KY-40 [3]

Lo zoccolo e i programmi per DY2 (digitale)

Programmi per il display LCD1602 [3]
Programmi per il multiplexer HW-178 [8]

Lo zoccolo e i programmi per DY3a (analogico)

Scanner [1]
Programmi per il display OLED SSD1306 128x 32 [2]
Programmi per il display OLED SSD1306 128X64 [2]

Lo zoccolo e i programmi per DY3b (analogico)

Scanner [1]
Programmi per il display LCD 1602 + adattatore I2C [3]

Programmi per più sensori

Un antifurto [1]
Una piccola drum machine [2]
Progetto per controllo della temperatura [1]
Una serratura a doppia chiave [1]
Telecamera OV7670

N.b.: i numeri tra parentesi quadre [X] indicano quanti programmi sono stati inseriti per il modulo in oggetto.

Sezione IV: le librerie

Sezione V: le appendici

Sezione VI: il montaggio della bs versione 0.6.1

Sezione VII: il montaggio delle schede accessorie

Introduzione

I microcontroller della famiglia di Arduino, e in particolare il minuscolo Arduino Nano, grande quanto un singolo circuito integrato, permettono di sperimentare un numero enorme di progetti diversi, alcuni semplicemente didattici, altri che possono diventare di uso comune. Gli unici limiti sono il numero delle porte disponibili, la lunghezza del codice e… la nostra fantasia.

Quando ero giovane, gli unici metodi per assemblare un progetto elettronico consisteva nel saldare i vari componenti su di una basetta preforata o di autocostruirsi un più strutturato e complesso circuito stampato.

Poi alcuni anni fa, qualcuno ebbe la geniale idea di creare una basetta con tanti piccoli fori, del passo dei piedini di un circuito stampato, dove tutti i fori verticali sono in contatto tra loro, e orizzontalmente ad essa scorre una doppia striscia in cui collegare l'alimentazione: è nata la breadboard! Quella che abitualmente si usa per testare i vari progetti creati con Arduino, Raspberry, ecc.

Questo sistema ha grandi vantaggi di semplicità ed economia, in quanto i componenti elettronici non vengono saldati e si possono usare più volte. Ma ho sperimentato anche alcune limitazioni. Per esempio, è facile sbagliare la fila di fori in cui effettuare le connessioni, essendo molto vicini; quando si usano più moduli, e in particolare con i display, i numeri di fili necessari aumentano di parecchio; ci vuole un po' di tempo per assemblare un progetto, ed è necessario disfarlo completamente per testarne uno nuovo.

Allora ho pensato di disegnare e costruire una basetta sperimentale, che permetta di collegare nativamente decine dei più comuni moduli e componenti usati nei progetti di Arduino, senza usare fili per cablaggi, o in casi particolari di usarne un numero esiguo. Questo riduce il numero di errori e il tempo necessario per assemblare un progetto. Naturalmente c'è il rovescio della medaglia: si rende un po' più rigida l'attribuzione delle porte per ogni singolo modulo, ma con l'esperienza si riescono a trovare varie soluzioni a queste problematiche.

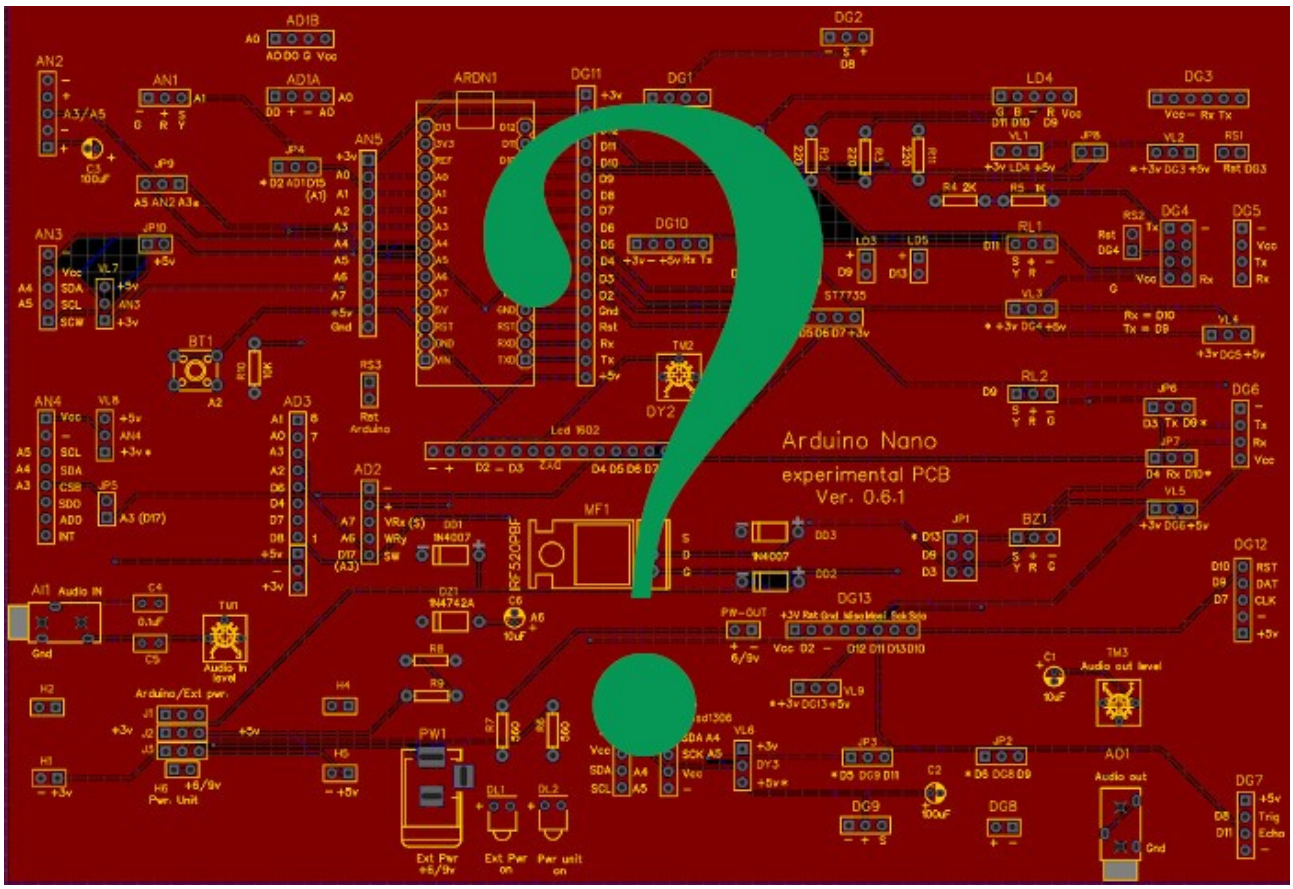
La nostra basetta sperimentale (*bs*), nasconde in uno spazio modesto un piccolo laboratorio per sperimentare decine e decine di progetti con Arduino Nano; essa ha un carattere ***didattico***, ovvero serve per impratichirsi nell'uso dei moduli e del linguaggio di programmazione di Arduino, e quindi può essere utilizzata anche nelle scuole tecniche di elettronica e programmazione; ma ha anche una ***funzione pratica***: come si vedrà in un paragrafo successivo, alcuni progetti testati e perfezionati con la *bs,* sono poi stati realizzati in modo definitivo su circuiti stampati creati ad hoc.

Sono arrivato, dopo aver disegnato e costruito diverse basette, alla versione 61, e ritengo che il progetto sia abbastanza maturo da tentare di renderlo pubblico. Spero che possa essere utile a coloro che forse leggeranno questo manuale, e che con l'esperienza maturata insieme, essendo il progetto open source, si possano apportare e *condividere* modifiche e migliorie, rendendo questo progetto sempre più utile, attuale e adattabile a svariate esigenze.

Piacenza, marzo/luglio 2023

Sezione I

Perché usare questa basetta sperimentale e didattica?



Cosa trovo in questo manuale (e cosa no)?

E' bene sapere subito cosa aspettarsi da questo manuale, per non rimanere delusi, o cercare inutilmente informazioni che non sono disponibili.

Cosa trovo in questo manuale?

Qui troverai nozioni dettagliate per conoscere la basetta sperimentale nei suoi aspetti pratici, la disposizione degli zoccoli e delle porte correlate, l'uso dei jumper che la rendono più flessibile; una breve descrizione dei moduli che sono stati testati personalmente.

Inoltre sono stati raccolti, modificati o scritti quasi duecento programmi per eseguire vasta gamma di assemblaggi di test. Molti programmi sono semplicemente didattici, ma ciò non toglie che ce ne siano alcuni di valore e piuttosto interessanti. Tutti i programmi sono stati testati a fondo.

Nelle appendici si trovano diverse tabelle, contenenti una cross-list dei vari codici relativi ai moduli utilizzati: informazioni raccolte con pazienza su Internet e assemblate in modo organico; informazioni sulle porte utilizzate e dei possibili conflitti; le istruzioni e i file Gerber per realizzare (o far stampare) la basetta sperimentale, oltre ad alcuni progetti minori di contorno; alcuni consigli, derivati dall'esperienza personale, per l'acquisto di alcuni kit relativi ai moduli utilizzati, senza alcuna sponsorizzazione da parte delle ditte indicate.

Cosa non trovo in questo manuale?

Si dà per scontato che chi intraprende questo progetto, abbia già una conoscenza di base di Arduino, e che quindi conosca, almeno a grandi linee:

- l'installazione della IDE di Arduino e il suo uso;
- che abbia una conoscenza di base della programmazione per Arduino;
- che sappia come reperire e installare le librerie necessarie;
- che in caso di necessità sappia ricercare e interpretare i datasheet dei vari moduli utilizzati;
- che sappia ricercare eventuali nuovi progetti in rete, e in base alle indicazioni fornite in questo manuale e alla sua esperienza personale, sia in grado di modificarli per adattarli alla basetta sperimentale (*bs*).

Tutte queste nozioni sono già state trattate ampiamente e in modo esaustivo in rete, per cui sarebbe inutile ripresentarle qui, sia per motivi di tempo e spazio necessario e sia per non appesantire troppo questo manuale. Lo spirito del free software aborre gli sprechi. Perché riscrivere ciò che si trova già in rete, scritto molto meglio di quello che potrei fare io?

Un utile strumento didattico

Penso che questa basetta possa essere un utile strumento didattico per le scuole tecniche in cui si insegna elettronica e programmazione.

Strumento didattico per istituti tecnici con indirizzo elettronico:

- se si utilizzando i file Gerber per far stampare esternamente la basetta sperimentale, gli allievi potranno poi assemblare i vari componenti, imparando a riconoscere e utilizzare transistor, mos-fet, diodi, resistenze, led, condensatori polarizzati e non;
- è possibile studiare il circuito elettrico,comprenderne la filosofia e (si spera!) di migliorarlo e condividerlo;
- inoltre potranno fare esperienza nel saldare i componenti necessari, oltre agli zoccoli e connettori con passo standard presenti sulla basetta e perfezionarsi nella tecnica;
- imparare a conoscere e utilizzare decine di moduli che possono essere utilizzati sulla *bs*, anche cercando i vari datasheet su Internet;
- avere la soddisfazione di costruire non una scheda fine a se stessa, ma in grado di assemblare una serie quasi illimitata di esperimenti diversi.

Strumento didattico per istituti tecnici con indirizzo di programmazione:

- la famiglia di Arduino (di cui Nano fa parte), utilizza un linguaggio di programmazione C semplificato, che permette agli allievi di imparare le regole di base della programmazione;
- inoltre quanto appreso può essere messo immediatamente in pratica, verificando sul campo se quanto scritto in modo teorico funziona veramente; se e dove si sono verificati gli errori;
- creare un progetto e verificarlo personalmente può dare ai ragazzi un senso di soddisfazione, perché vedono che i propri sforzi danno risultati pratici, creando progetti semplici, ma spesso divertenti e anche funzionali;
- nei moderni computer si dispone di una quantità di risorse e di memoria quasi illimitati, e questa caratteristica può indurre il programmatore a usare tecniche poco funzionali e scarsamente strutturate. Al contrario, in Arduino si richiede una programmazione chiara, sintetica, strutturata ed efficace, dato le modeste risorse hardware. Come tale, è un ottima palestra per i giovani e non;
- stimola la creatività degli allievi, potendo utilizzare decine e decine di moduli dal costo veramente modesto, alla portata degli Istituti e degli allievi stessi.

Inoltre, vantaggio non da poco, elettronica e programmazione possono convivere insieme in modo divertente e stimolante.

Uno strumento utile per lo sperimentatore e per l'hobbista:

- permette affinare i propri progetti, sperimentarli in pochi minuti e poi realizzare circuiti indipendenti, come nel caso di un termometro per un prototipo di una particolare stampante 3D.

Un progetto economicamente sostenibile

Spesso i giovani, le scuole secondarie, gli istituti tecnici e non solo loro, hanno un budget limitato per acquistare il materiale per fare i loro esperimenti.
Questa basetta sperimentale, insieme a un Arduino Nano e una manciata di componenti, permette di eseguire una notevole quantità di esperimenti e di impratichirsi allo stesso tempo nell'uso dell'hardware e della programmazione.

In rete si trovano tantissimi porgetti, video, tutorial gratuiti che permettono a costo zero di impratichirsi nell'uso del "nostro microcontroller.

La basetta sperimentale e tutti gli accessori presenti in questo manuale, compreso il manuale stesso, sono distribuiti sotto licenza Creative Commons 4.0, quindi nessuno deve pagare a me un euro per duplicare le schede che illustreremo di seguito. Naturalmente eventuali perfezionamenti devono essere rilasciati con la stessa licenza e indicata la paternità del progetto.

Ma ora veniamo al puro costo dei componenti:

- sui siti cinesi, quali Ali Express, Bangood, o più tradizionali come Amazon ed E-bay, un Arduino Nano compatibile può essere acquistato a meno di 4 €, spese di spedizioni e tasse comprese (luglio 2023).
- Un kit contenente un buon numero di sensori, quale Elegoo "35 in 1 kit" costa circa 38 euro. In alternativa, un kit 45 in 1, si trova su internet a 23 euro. I componenti sono in quantità maggiore che nel kit Elegoo, ma di valore inferiore. Per maggiori info, clicca qui
- L'acquisto di 5 basette sperimentali 0.6.1, usando un corriere non particolarmente rapido (comunque ho ricevuto anche con corrieri economici in circa 10 giorni), circa 35 euro, quindi circa 7 €/cadauno per una basetta costruita con qualità professionale. Personalmente ho usato la ditta JLCPCB, sponsorizzata da Easy Eda, il programma on-line che ho usato per disegnare il progetto. Ma penso che altre ditte similari offrano un servizio equivalente.
- Se si acquistano i componenti on-line in una certa quantità, il costo del materiale necessario per assemblare una basetta è di circa 7 €/cadauna.
- Una serie di moduli aggiuntivi per ampliare un poco la gamma degli esperimenti possibili: ipotizziamo 20 €, ma non sono indispensabili, almeno per iniziare.

Quindi, facendo un po' di conti:

• n. 1 Arduino Nano:	4 €	
• n. 1 kit Elegoo 35 in 1	38 €	
• n. 1 basetta sperimentale assemblata	14 €	(considerando di acquistare 5 basette e componenti all'ingrosso)
• vari moduli accessori	20 €	(opzionali)
Totale	**76 €**	

Perciò, con circa 75 euro è possibile avere un'ottima base di partenza; volendo fare una spesa iniziale ancora più all'osso (integrandola in futuro), si può partire anche con una spesa ancora minore:

• n. 1 Arduino Nano:	4 €	
• n. 1 kit 45 in 1 kit	23 €	
• n. 1 basetta sperimentale assemblata	14 €	(considerando di acquistare 5 basette e componenti all'ingrosso)
Totale	**41 €**	

Per ulteriori informazioni sui kit e sul loro contenuto, **clicca qui**

Nota: queste considerazioni sono il frutto delle mie esperienze e dei miei errori, totalmente libere, in quanto non ho alcuna sponsorizzazione dalle ditte indicate in tutto il presente manuale.

Riflessioni sui vari modelli di Arduino Nano

Relativamente a questa versione della ***bs***, inizialmente è stato considerato solamente Arduino Nano che chiameremo "base", il più comune ed economico della ormai nutrita gamma dei "Nano".

Nella prossima versione della bs, la 0.6.2, cercheremo di interagire in modo ottimale anche con le versioni più recenti, quali Nano 33 IoT e Nano 33 BLE.

Perché finora non sono ancora stati considerati nella versione attuale, la 0.6.1? Il problema è l'alimentazione. Sebbene tutte le versioni del microcontroller presentino la stessa disposizione dei piedini, la versione base fornisce in uscita e accetta in ingresso due livelli di alimentazione: +3,3v e + 5v, mentre sia il 33 IoT che il 33 BLE *nativamente* forniscono solo +3.3v; con una piccola modifica hardware si possono fornire a certe condizioni anche +5v, ma essi *accettano in ingresso solamente i + 3,3v, pena il danneggiamento dei microcontroller stessi*. Questa situazione richiede quindi una attenzione particolare, per evitare di danneggiare i nostri preziosi Arduino Nano!

Proprio nell'ottica di fornire un progetto economicamente sostenibile, specialmente per le scuole, è necessario fare alcune riflessioni. Cosa offrono di più le versioni "33" dei microcontroller, e con quali costi?

Arduino Nano "base"

Cosa ha di più Arduino Nano 33 BLE rispetto alla versione "base"?

- Modulo wi-fi integrato
- accelerometro/ giroscopio integrato
- orologio (RTC) integrato?

Arduino Nano 33 BLE

Cosa ha di più Arduino Nano 33 IOT rispetto alla versione "base"?

- Modulo wi-fi integrato
- accelerometro/ giroscopio integrato
- modulo wi-fi integrato
- orologio (RTC) integrato

Arduino Nano 33 IoT

Certamente è molto comodo avere degli importanti moduli integrati all'interno del microcontroller stesso, però si possono fare alcune osservazioni di carattere pratico ed economico:

- attualmente (luglio 2023) suo sito di Amazon, un Arduino Nano 33 BLE costa circa 25,5 €, mentre un Arduino Nano 33 IoT costa circa 29 €.
- Nello stesso periodo, sui "soliti" siti cinesi si trova:
  - Arduino Nano "base" compatibile, compreso spese di spedizioni e tasse: 4 €
  - modulo wi-fi ESP8266, compreso spese di spedizioni e tasse: 2,5 €
  - modulo bluetooth HC-06, compreso spese di spedizioni e tasse: 4 €
  - modulo orologio (RTC) DS1307, compreso spese di spedizioni e tasse: 3,5 €
  - accelerometro/giroscopio MP-6050, compreso spese di spedizioni e tasse: 2,5 €
  - Totale complessivo di Arduino *base* più i moduli accessori: **16,5** €

Quindi attualmente, usando sia un Arduino Nano che moduli compatibili ma comunque perfettamente funzionanti, il rapporto dei costi è ancora più vantaggioso con il modello base insieme con i moduli necessari per essere paragonabile – nelle funzionalità - con un Nano 33 IoT.

Naturalmente questa ultima versione è tecnologicamente più avanzata del modello precedente, più veloce e integra dei moduli importanti; tuttavia con la vecchia versione si può utilizzare una gamma di moduli esterni superiore, accettando quelli che funzionano con entrambe le tensioni di alimentazione.

Perciò si può stilare una tabella sintetica dei vantaggi/svantaggi dei vari moduli:

Arduino Nano "base"		**Arduino Nano 33 Iot**	
Vantaggi	Svantaggi	Vantaggi	Svantaggi
- doppia alimentazione - prezzo competitivo (compresi i moduli esterni) - grande numero di moduli disponibili	- moduli esterni - tecnologia più datata	- moduli interni - tecnologia aggiornata - maggiore velocità di clock e memoria	- singola alimentazione - costo superiore - accetta solo moduli alimentati a 3,3v

In questa tabella è stata omessa la comparazione con la versione 33 BLE, che presenta gli stessi inconveniente e (quasi) gli stessi vantaggi della versione IoT.

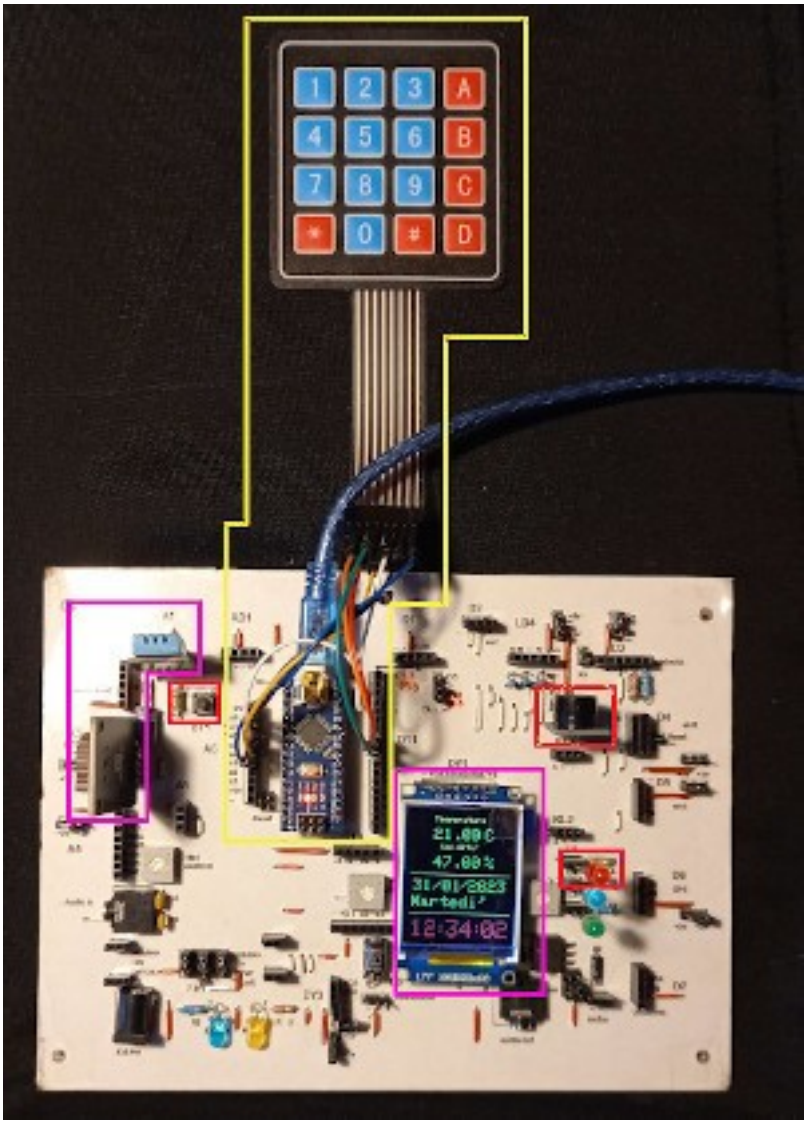
Comunque il rapporto di costi tra di esso e la versione "base" è la seguente: 25,5 € contro 14 €. Da un punto prevalentemente economico, il caro vecchio Nano è ancora più conveniente.

Naturalmente, come accennato all'inizio del paragrafo, nella prossima versione della ***bs*** integreremo una gestione dell'alimentazione perfezionata a +3,3v per poter utilizzare correttamente e senza rischi anche le versioni di Nano 33.

Attenzione! Se colleghi Arduino Nano 33 BLE e 33 IoT alla ***bs*** nella presente versione, ovvero 0.6.1, è possibile danneggiare il nostro prezioso microcontroller. Si consiglio vivamente, per avere maggiori informazione sui problemi di alimentazione, **di cliccare qui**.

Un'ultima considerazione: è probabile che l'orientamento del mercato e della tecnologia sia quello di produrre sempre più moduli che funzionano a +3,3v; tuttavia al momento presente, la maggioranza dei moduli funziona ancora a +5v, per cui molti di essi potrebbero non funzionare se alimentati a una tensione minore, oppure dare delle informazioni errate. Sarà necessario valutare caso per caso se e come utilizzare i vecchi moduli.

Flessibilità e ridondanza hardware della *bs*



A volte si avrebbe il desiderio di provare rapidamente più programmi senza smontare i componenti di un progetto per testarne un altro.
Su una breadboard con tanti cavetti e connessioni, questa operazione diventa difficile e con rammarico ci si trova costretti a smontare sensori e cavetti per testare un secondo programma, e poi ritornare al primo, oppure possedere più breadboard e moduli per verificare più sketch in parallelo.
Con la nostra *bs* è tutto più semplice: attualmente su questo prototipo della vecchi versione 4.2 convivono addirittura tre progetti hardware:
- quello contornato di giallo è un test su di una tastiera a membrana con 16 tasti;
- un secondo contornato di rosso, è relativo al test del pulsante BT1 integrato sulla basetta, che emette un suono dal buzzer e si accende un led quando premuto;
- un terzo, contornato color fuxia, che integra un sensore di temperatura DHT11, un orologio DS1307 e un display ST7735, in funzione attualmente.

Si potrebbe ancora inserire altri moduli, controllati da diversi programmi. E' sufficiente caricare di volta in volta il programma corretto.

In questa *bs*, si è fatto un grande sforzo non solo di inserire un grande numero di zoccoli, permettendo anche una cnotevole ***ridondanza***, come indicato a volte nella descrizione dei vari moduli, ma si è cercato di collocarli in modo ***ergonomico***, affinché sia teoricamente possibile inserire un modulo in ogni zoccolo senza che si intralcino l'uno con l'altro. L'unica eccezione sono i dispaly che alloggiano in DY1 (TFT 1,7") e DY2 (LCD 1602), ma è difficile che in uno sketch siano necessari entrambi. In ogni caso restano sempre i display alloggiati su DY3.

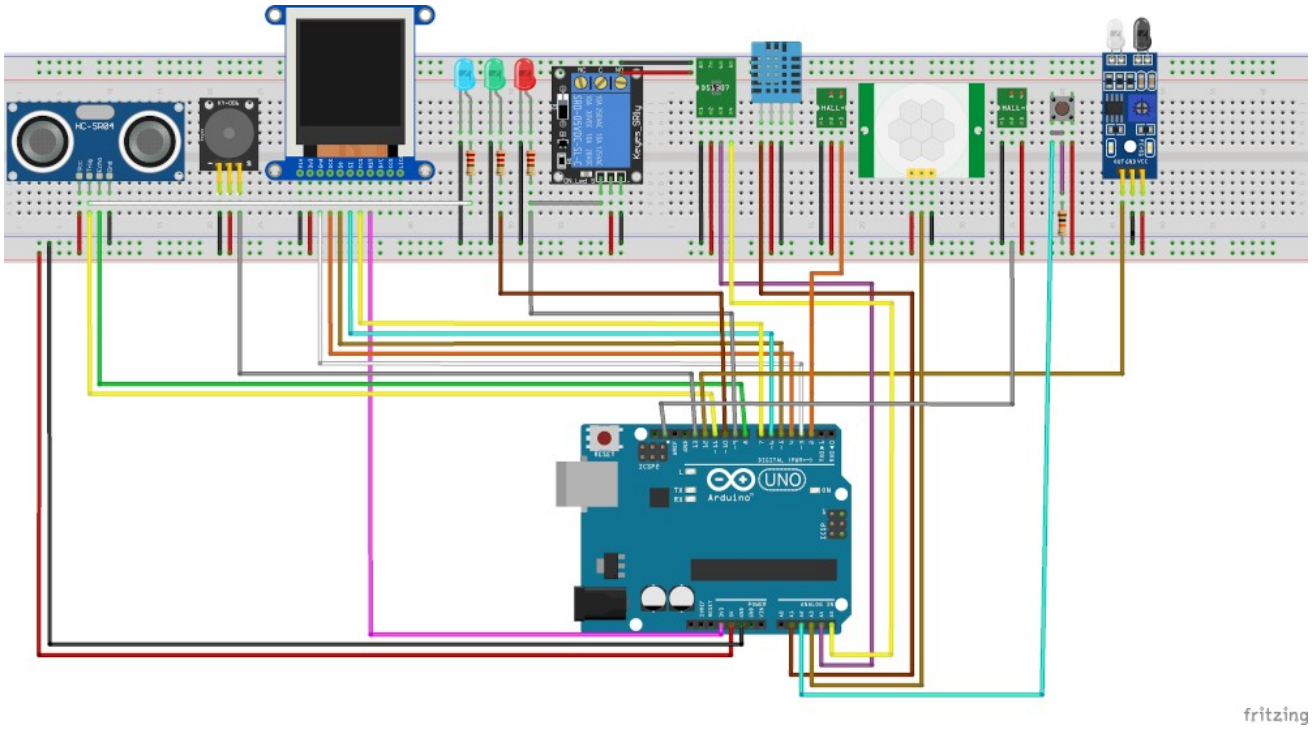
Quindi sarà possibile, nel caso sia necessario di far coesistere più progetti hardware contemporaneamente, attivandone uno per volta. Naturalmente non si ottengono miracoli, ma spesso mi è successo di poter utilizzare questa tecnica di "overload".

Nota: in foto appare una vecchia versione della bs, la 4.2.

Si possono ridurre tutti questi cavi?

Come dicevo nell'introduzione, l'invenzione della breadboard è stata una benedizione, però se si deve affrontare un progetto di una certa complessità, come ***l'antifurto*** presentato nel paragrafo successivo.

La realizzazione pratica diventa complessa, e la gestione di tanti cavi può risultare piuttosto disordinata e difficile da comprendere.



Questa è immagine dello sketch dell'antifurto che corrisponde a quello illustrato più diffusamente nel paragrafo successivo. E' stato realizzato con "**Fritzing**", un programma molto utile per disegnare i propri progetti, e in un secondo tempo permette di creare anche l'immagine della basetta da stampare.

Con la nostra ***bs***, si può realizzare lo stesso progetto in un paio di minuti, caricare il programma e vedere il risultato. Poi si può smontare il tutto altrettanto rapidamente, inserire i moduli relativi a un altro progetto, testarlo, e poi riassemblare nuovamente il programma precedente in un tempo molto breve.

Con una breadboard (lo so per esperienza) quando si riesce ad assemblare un progetto di questa complessità, dopo aver trovato e risolti tutti gli errori ed effettuati i miglioramenti necessari, si è poi molto restii a smontarlo, perché i tempi per ricostruirlo nuovamente sono rilevanti. In questo caso, o si hanno varie breadboard, Arduino, cavi, moduli per eseguire più sketch contemporaneamente, oppure si rischia di rimanere bloccati per parecchio tempo – anche per semplice pigrizia - su di uno stesso progetto, o di trovarsi nell'impossibilità di averlo rapidamente a disposizione tutte le volte che serve.

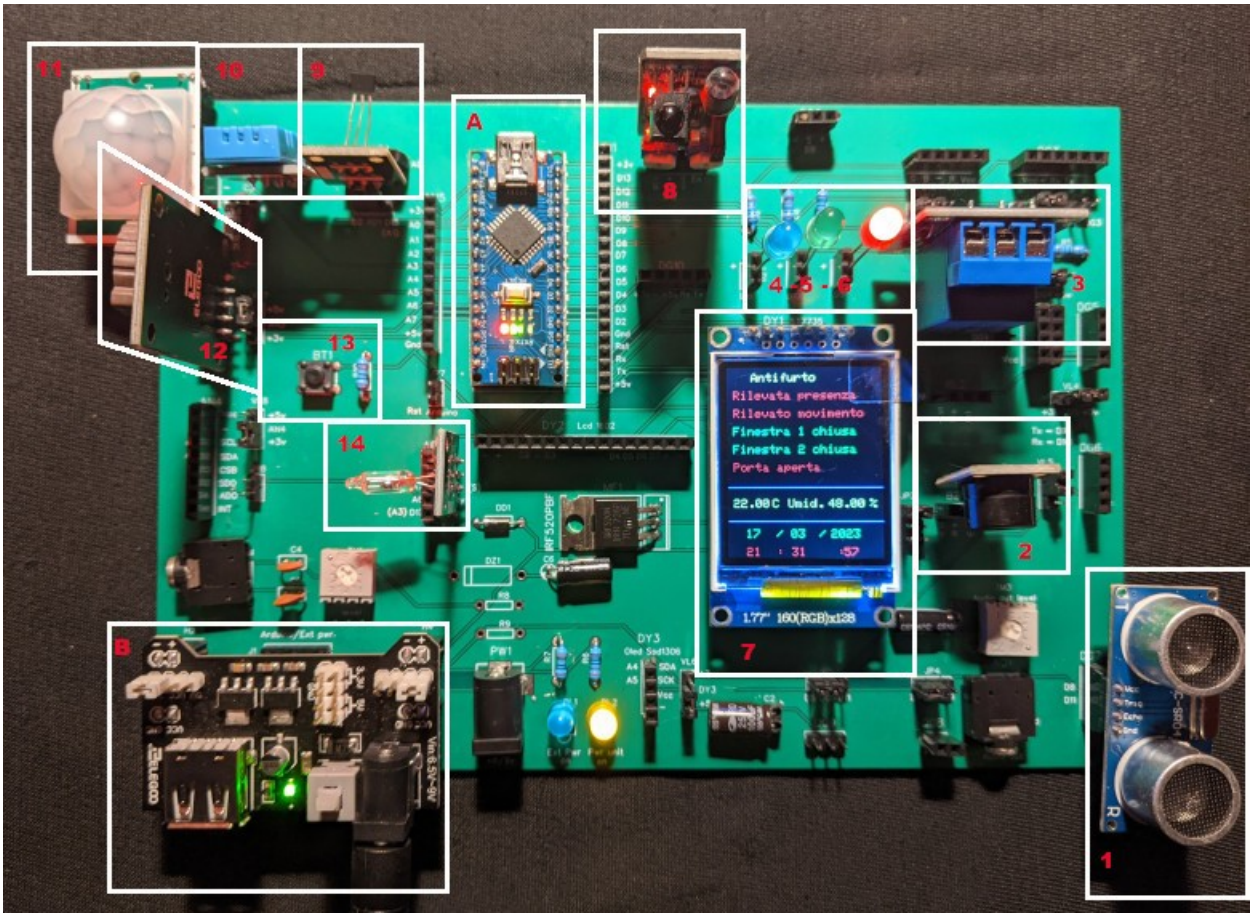
Quanti moduli possono interagire contemporaneamente in uno stesso programma? (e con un minimo uso di cavetti?)

A volte si ha in mente un progetto con un gran numero di moduli, e tutte le linee del programma già ben chiare in testa... ma si potrà realizzare con Arduino Nano?

E' necessario valutare alcuni aspetti:

- la disponibilità di ram di Arduino Nano, pari a 30720 byte, molto simile a quella disponibile sul mio amatissimo Commodore 64, un famoso microcomputer degli anni '80;
- lo spazio per le variabili (2048 byte);
- le porte che possiamo utilizzare: 13 porte digitali e 8 analogiche (di cui sei possono essere convertite in digitali, facendo attenzione a evitare conflitti);
- la struttura della ***bs***. Nel corso del tempo e dei vari prototipi (attualmente si è alla versione 6.1), si è cercato di differenziare i vari zoccoli (socket) e di inserire vari jumper, in modo di dare la massima flessibilità possibile, che potrà essere ancora migliorata con l'aiuto dei consigli di chi la testerà;
- la nostra fantasia e la nostra abilità di "incastrare" i vari moduli, quasi stessimo giocando a tetris.

Per testare questa possibilità, si è creato un piccolo progetto didattico di un **antifurto** per casa, che può controllare contemporaneamente cinque luoghi e/o accessi all'appartamento. Per i test tutti i moduli sono stati inseriti sulla bs (senza cavi aggiuntivi!). naturalmente se si volesse realizzare realmente, i sensori andrebbero disposti nei punti strategici, ovviamente collegati con dei cavi.



Un piccolo progetto didattico di un antifurto

Quanti moduli abbiamo usato in questo progetto?

Per realizzare questo progetto, abbiamo usato i seguenti moduli:

1. HC-SR04, un sensore ultrasonico di movimento e prossimità. Questo modulo ha un raggio di azione di qualche metro. Collegato sullo zoccolo DG7;
2. un buzzer (piccolo altoparlante piezoelettrico) su BZ1, che funge da allarme in caso di rilevazioni anomale da parte degli altri sensori; presenta anche un'uscita audio, per collegarsi a un amplificatore esterno, sul minijack standard da 3,5 mm, denominato AO1;
3. un relay che si attiva in caso si verifichi qualunque anomalia rilevata da uno o più sensori; su RL2;
4. un led blu che lampeggia, per indicare il funzionamento del sistema; su LD1;
5. un led verde, che segnala la mancanza di anomalie, o nel caso si siano verificati degli allarmi, si riaccende dopo il reset del sistema, ottenuto premendo BT1. Il led alloggia su LD2.
6. un led rosso, che si attiva insieme al buzzer e al relay in caso di anomalie, segnalate dai sensori. Su LD3;
7. un display TFT grafico a colori, per mostrare le informazioni raccolte dai vari sensori;
8. Un sensore "obstacle avoiance", KY-032. In genere viene usato per rilevare ostacoli, ma in questo progetto viene utilizzato per verificare la chiusura di una finestra. Zoccolo DG1.
9. KY-003, un sensore per l'effetto hall (magnetico) , per verificare la chiusura di una porta. Inserito nello zoccolo AD1, alloggiato verso sx, sul pin digitale;
10. un sensore di umidità e temperatura, DHT11, alloggiato sullo zoccolo AN1;
11. Il pir HC-SR501. E' un sensore passivo a raggi infrarossi, e rivela la presenza di persone, animali e oggetti che emettono calore, nel raggio di alcuni metri. Collegato allo zoccolo AN2;
12. un modulo datario/orologio (RTC), DS1307, collegato su AN3;
13. un pulsante di reset, alloggiato direttamente sulla basetta, BT1;
14. Un sensore a bolla di mercurio, KY-017, usato per verificare l'apertura di un'altra finestra; alloggia sullo zoccolo AD2;

A. naturalmente un Arduino Nano;

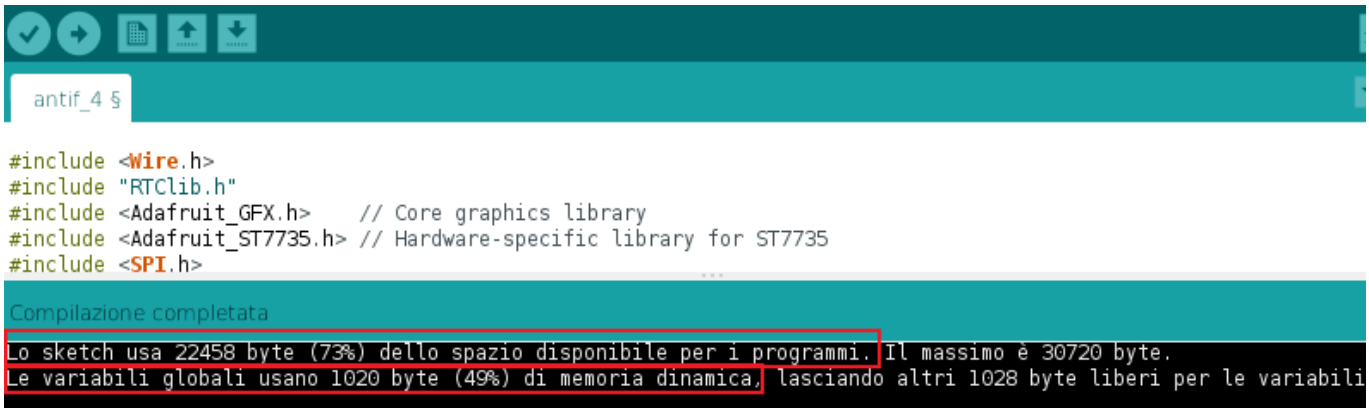
B. una power unit, perché sulla ***bs*** ci sono molti componenti, e quindi un rilevante assorbimento di corrente.

Realizzare un progetto così articolato su una breadboard diventa oltremodo complesso, a causa delle miriadi di collegamenti e di ponticelli, per cui il rischio di errori è piuttosto elevato, oltre a quello di una difficile leggibilità del progetto. Per essere eseguito richiede tempo e mooolta pazienza...

Con la nostra basetta sperimentale, questo progetto può essere assemblato in un paio di minuti, senza errori, e inoltre senza un solo cavo volante!

Nota: nel caso si desideri realizzarlo effettivamente, i sensori andranno collegati alla basetta con dei cavi di lunghezza appropriata.

Quanta memoria abbiamo usato per scrivere questo progetto?



Dalle informazioni che si rilevano lanciando il programma, rimane libera ancora una parte discreta quantità di memoria per i programmi, e anche le variabili occupano meno della metà dello spazio disponibile. Volendo potremo ancora inserire qualche sensore e ulteriori funzioni...

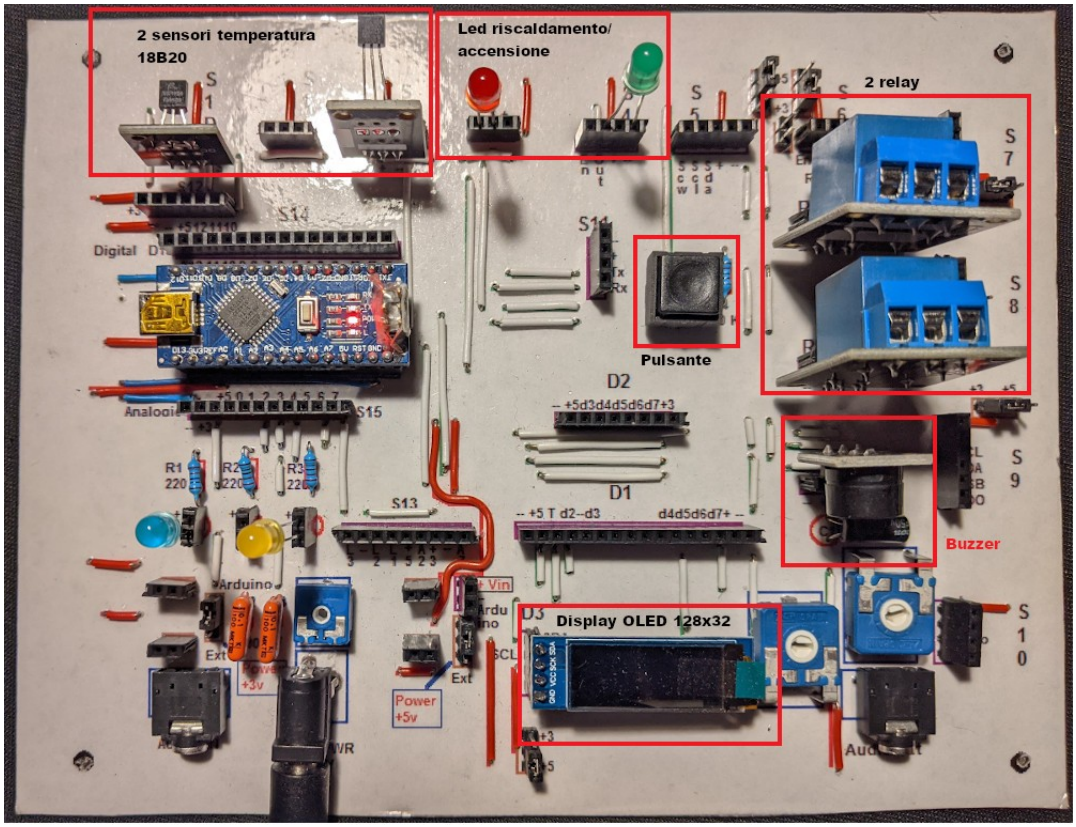
Clicca qui per ulteriori informazioni sulla realizzazione del progetto e il listato del programma.

Dal progetto sperimentale a quello definitivo

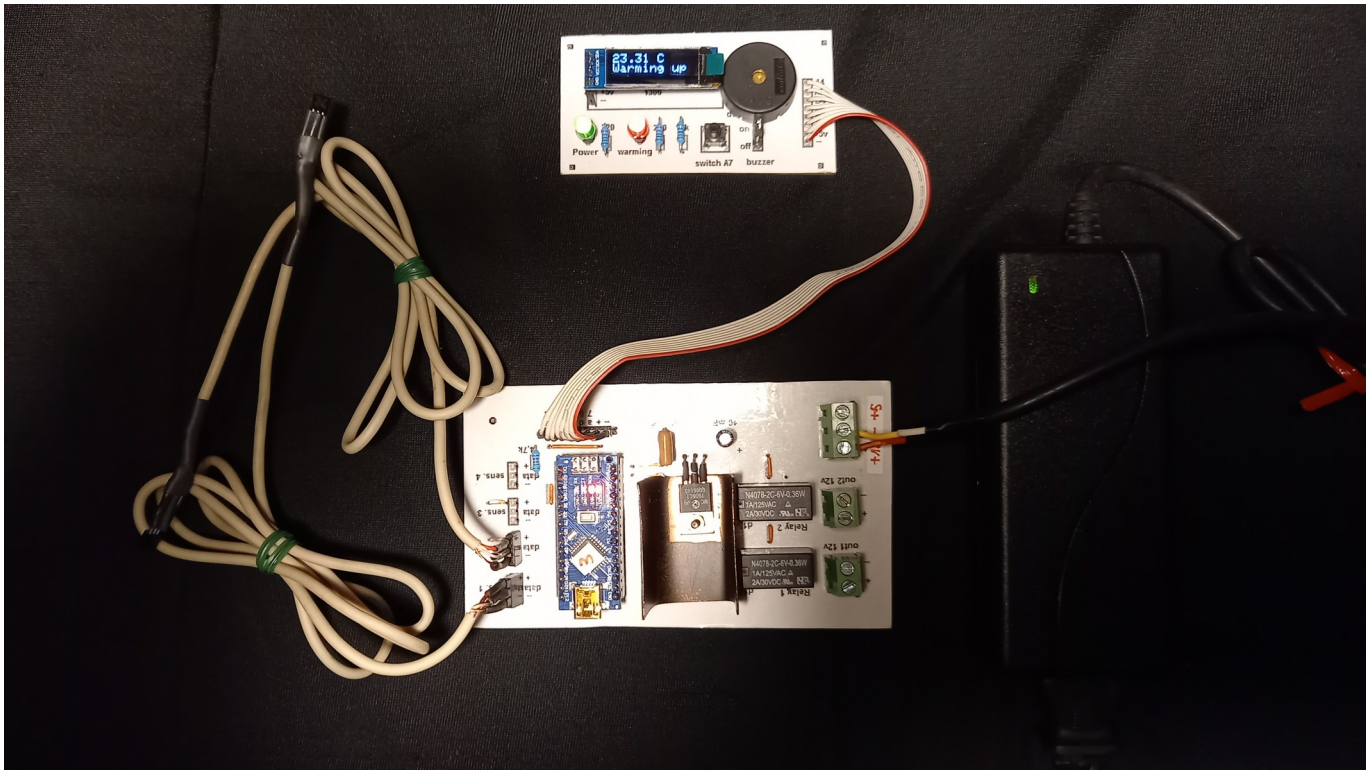
Qualche tempo fa mio figlio, che si occupa di progettazione di macchine industriali, aveva la necessità per allestire un circuito elettronico per mantenere costante la temperatura di un fluido per la stampa di un guscio su di un prototipo di una nuova stampante 3D in ambito elettro-medicale. La temperatura doveva oscillare tra un minimo e un massimo, senza raggiungere durante il riscaldamento una temperatura eccessiva per non alterare le caratteristiche del materiale. Con Arduino Nano ho progettato uno sketch con due sensori (uno per la temperatura del fluido, l'altro per quella delle resistenze). Arduino pilota le due resistenze separatamente per mezzo di due relay; all'inizio del riscaldamento entrambe sono attive per raggiungere rapidamente la temperatura di esercizio; ma quando ci si approssima ad essa, una delle due viene disattivata, per raggiungere la temperatura ottimale con gradualità e precisione.

E' stato realizzato un pannello, con un display per mostrare le due temperature fluido o resistenze) e lo stato del sistema. Abitualmente sul visore appare la temperatura del fluido; quando si preme il pulsante, o quando la temperatura raggiunge un limite critico, viene visualizzata quest'ultima sul dispay Oled 128x32. Oltre che la temperatura, sono visualizzate alcune informazioni relative allo stato di esercizio (riscaldamento, temperatura critica, ecc.).
Un led verde indica il funzionamento, un secondo, rosso, la fase di riscaldamento. Entrambi lampeggiano quando si supera la temperatura massima, insieme a un buzzer che segnala l'allarme, fino a quando la temperatura scende sotto il limite critico, grazie allo spegnimento immediato e totalmente automatico di entrambe le resistenze, che verranno riaccese quando si sarà scesi sotto al limite, in un ciclo continuo.

In un primo tempo il progetto è stato realizzato su di una vecchia basetta sperimentale, testato a fondo fino alla completa ottimizzazione del programma.



La *bs* sperimentale usata per i test, ancora una vecchia versione "casalinga", addirittura la 3.0, comunque perfettamente funzionante!



...e poi sono state progettate e realizzate le due basette per il progetto definitivo.

Ecco il progetto definitivo:

- sulla sinistra si vedono i due sensori, collegati ai cavi;
- in basso in centro la basetta principale con Arduino Nano. E' stato inserito anche un sistema di alimentazione a 12 v per le resistenze; al centro si vede il transistor che riduce la tensione per Arduino con una aletta per dissipare il calore. Sulla destra i due relay per pilotare l'alimentazione delle resistenze;
- sempre sulla destra, l'alimentatore a 12 V 3A;
- In alto, collegato con un cavo a più fili, la piccola basetta con i comandi: il visore Oled, i led di segnalazione di accensione e riscaldamento; il buzzer di allarme e il pulsante per selezionare la temperatura del fluido o delle resistenze.

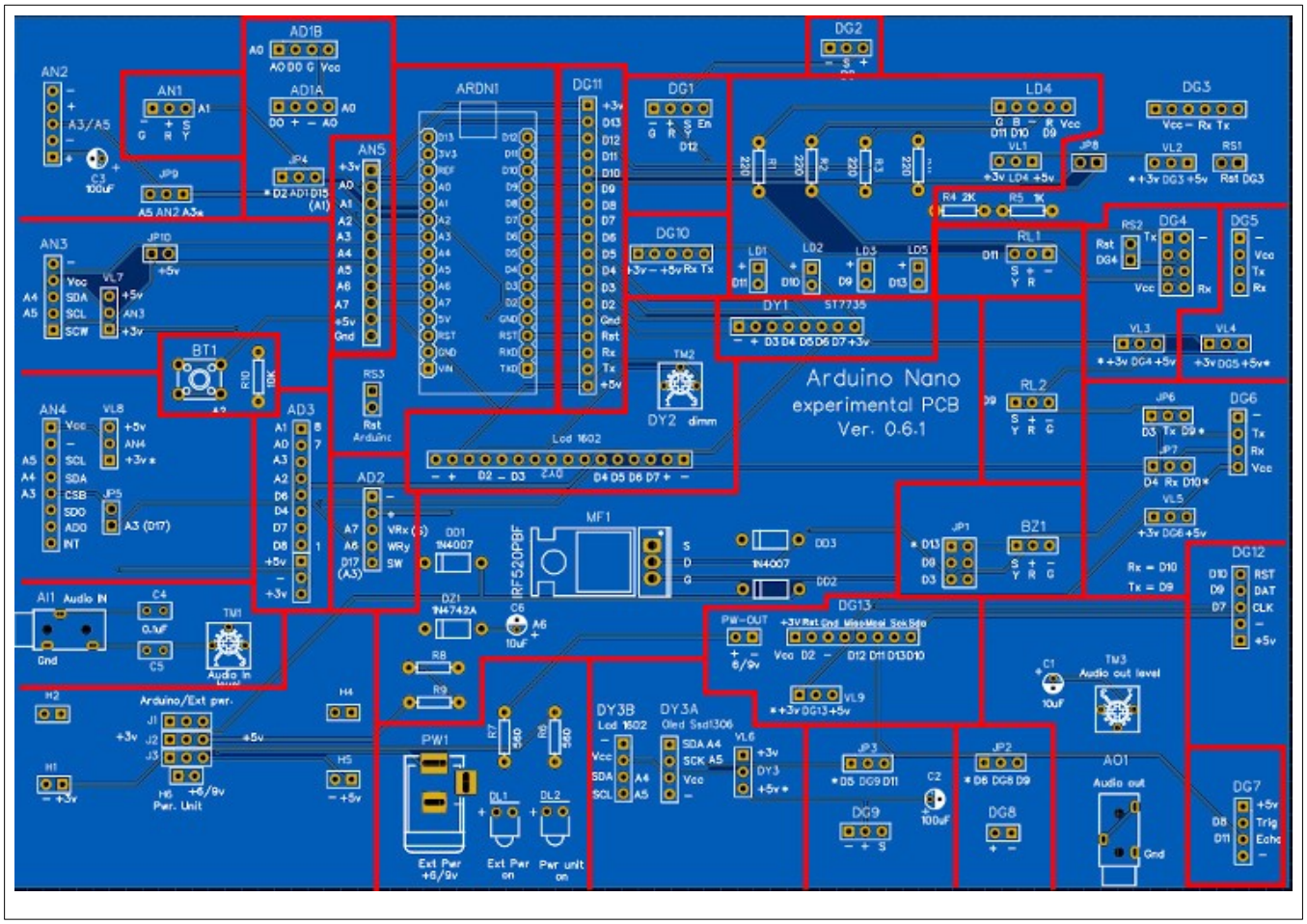
Adottando questa modalità si è riusciti rapidamente a testare il progetto, metterlo a punto e poi eseguire il circuito stampato, l'assemblaggio e la messa a punto del sistema, che in un secondo tempo è stato inserito all'interno della stampante 3D per applicazioni medicinali.

Clicca qui per andare alla pagina del progetto.

Sezione II

Conoscere la basetta sperimentale

ma non è poi così impegnativa...



Ecco l'immagine della basetta sperimentale nella versione 6.1, divisa per zone logiche. E' leggermente più piccola dell'originale, per motivi di impaginazione. Dimensione effettiva: 197 x 134 mm (l x h).

Questa è l'immagine della basetta sperimentale, nella versione 6.1. Le sue dimensioni reali sono di 197 x 134 mm (l x h), all'incirca le dimensioni di un foglio A5 in orizzontale. Apparentemente può sembrare complessa, ma leggendo le istruzioni seguenti, la comprensione del suo utilizzo diventerà sicuramente più chiara e intuitiva.

I primi prototipi della basetta sperimentale stessa (che d'ora in poi indicheremo per brevità come "***bs***") furono progettati ed eseguiti in modo tradizionale, ovvero le tracce in rame apparivano solamente sulla facciata inferiore (singola faccia) ed erano realizzati in modo casalingo (vedi la genesi del progetto). Mentre nell'ultima versione che propongo qui, è stata progettata con "Easy Eda" ed è stata stampata in modo professionale, con un circuito a doppia faccia e serigrafata dalla società JLCPCB (sponsorizzata da Easy Eda). Questa nuova versione ne rende più chiaro il montaggio e l'utilizzo, riportando sulla superficie molte preziose informazioni.

Nota importante 1: l'interfaccia (IDE) tra Arduino Nano e il personal computer non fa parte di questo manuale, tuttavia nelle appendici viene trattato rapidamente l'argomento perché, specialmente al neofita, potrebbe creare qualche difficoltà, o peggio ancora, il dubbio che la basetta sperimentale non funzioni correttamente.
Clicca qui per informazioni sull'interfaccia Arduino Nano/personal computer

Nota importante 2: **la basetta sperimentale <u>è stata sperimentata a fondo esclusivamente per l'Arduino Nano tradizionale</u>**. Sebbene Arduino Nano 33 BLE e IoT condividano la stessa piedinatura, possiedono anche alcune caratteristiche differenti nella gestione delle tensioni di alimentazione. Il rischio che qualcosa non funzioni, o addirittura si possa danneggiare il microcontroller è sicuramente presente.
Clicca qui e leggi molto attentamente le pagine seguenti, per evitare di danneggiare un Arduino Nano 33. L'autore non si prende alcuna responsabilità relativamente all'uso di un Arduino Nano 33 con la ***bs***.

Elenco delle caratteristiche della bs:

- 22 zoccoli (sockets) per porte digitali, di cui cinque permettono di scegliere con quale tensione alimentare il modulo collegato;
  - 1 (DG11) è considerato "universale" perché permette di usare tutte le porte digitali disponibili:
  - 10 (da DG1 a DG7, DG10, DG12, DG13) per moduli generici che usano porte digitali;
  - 2 (DG8 e DG9), rispettivamente per motori e servomotori elettrici;
  - 5 (da LD1 a LD5) specifiche per l'uso di led;
  - 2 (RL1 e RL2) specifiche per relay;
  - 2 (BZ1 e AO1) per l'uscita audio; BZ1 per un buzzer e AO1 con uscita jack 3,5 standard per collegarsi ad un amplificatore audio.

- 3 zoccoli per porte miste analogiche e digitali, denominati AD1a/AD1b (connettore sdoppiato), AD2, AD3;
- 5 zoccoli per porte analogiche, da AN1 ad AN5, di cui due permettono di scegliere con quale tensione alimentare il modulo collegato; uno di essi (AN5) è considerato "universale" perché permette di usare tutte le porte analogiche disponibili;
- 3 zoccoli specifici per i display:
  - DY1: per un display digitale ST7735, TFT da 1,77";
  - DY2: per un display digitale LCD 1602;
  - DY3a/DY3b (connettore sdoppiato): per display analogici OLED 128x32 o 128x64; si può collegare ad esso anche un display LCD 1602 con adattatore I2C;
- 3 possibilità di alimentazione:
  - via USB, attraverso il computer;
  - direttamente con un alimentatore esterno (o una pila), compreso tra 7v e 9 v;
  - con un alimentatore esterno, attraverso una power unit, tipo Elegoo "Power MB V2".

La bs è stata testata con oltre 80 moduli facilmente reperibili per Arduino Nano.

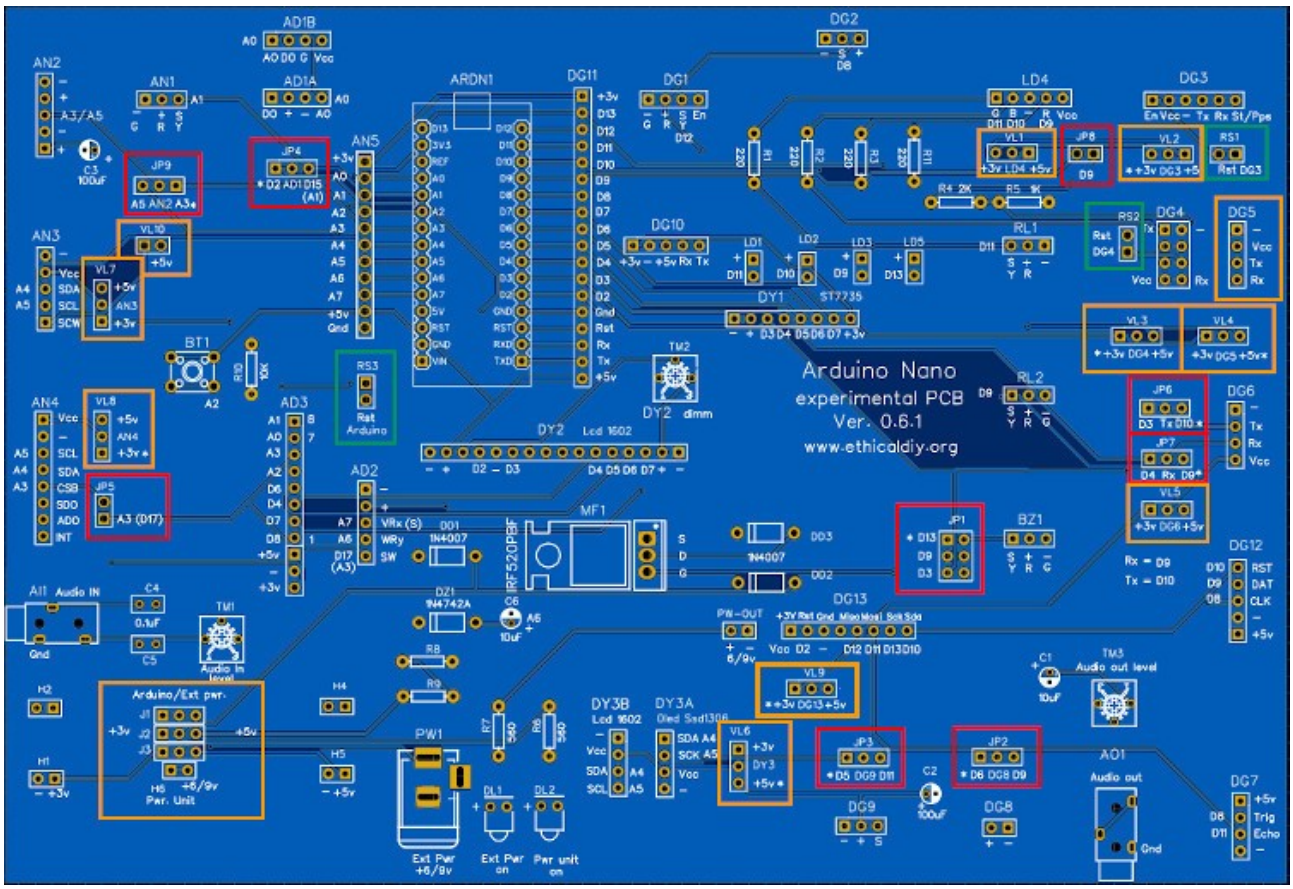
Analisi della bs in dettaglio.

Passiamo ora ad analizzare la disposizione di massima della ***bs****:* in alto, leggermente a sinistra, c'è l'alloggiamento per Arduino Nano (ARDN1), orientato con la connessione USB verso l'alto; è naturalmente in cuore del progetto.

La ***bs*** stessa è divisa idealmente, seppure in modo grossolano, in due sezioni verticali: guardandola dall'alto, sulla destra di Arduino ci sono le connessioni agli zoccoli (socket) digitali (le sigle iniziano con "Dgx", dove al posto della "x" appare il numero del connettore), oltre a BZ1/AO1 e RL1, RL2; e da LD1 a LD5 gli alloggiamenti per i led; mentre nella parte sinistra ci sono quelle analogiche (con sigla che "ANx") e analogico/digitali (sigle "ADx"). Anche l'ingresso audio "Audio in" (AI1) è analogico. Nel centro della basetta ci sono gli alloggiamenti per i display: DY1 e DY2 sono digitali, mentre DY3 è analogico. L'uscita audio "Audio out" (AO1) è anch'essa

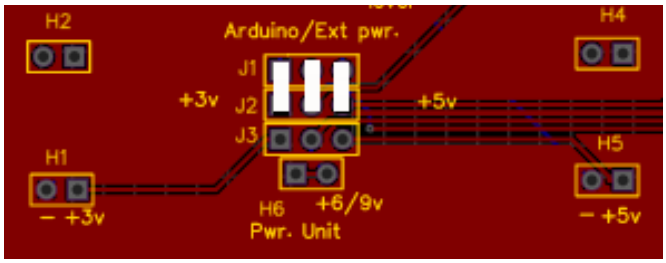
digitale. In basso sulla sinistra appare nel riquadro la sezione per l'alimentazione esterna, denominato Arduino/Ext. Pwr. I poligoni di colore blu mostrano le aree logiche in cui è divisa la *bs*.

I jumper: essenziali per la configurazione iniziale della bs



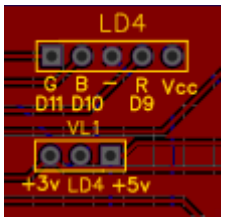
I jumper della bs evidenziati

I jumper. In gergo elettronico si chiamano "jumper" i ponticelli che possono essere impostati sulla *bs* per cambiare alcuni settaggi hardware. Ci sono quattro tipi di jumper:



Per iniziare, ponticellare i jumper come indicato.

I più fondamentali di tutti sono quelli con indicazione "Arduino/ext pwr", visibili nel riquadro in basso a sinistra.

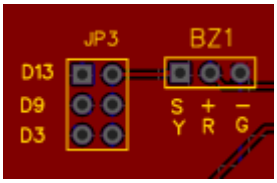


VL1 regola l'alimentazione di LD4 (vedi il pin centrale di VL1)

- **IMPORTANTE: se i jumper non vengono configurati correttamente, i moduli inseriti sulla *bs* non verranno alimentati**, e quindi si potrebbe pensare che la bs stessa non funzioni!

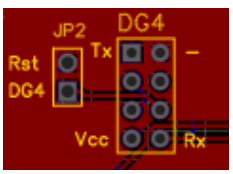
Per ulteriori informazioni, vai al **paragrafo dedicato all'alimentazione**.

- I jumper con sigla "**Vlx**" (dove "x" è un numero progressivo), permettono di variare la tensione che sarà presente sullo zoccolo di riferimento, per poter alimentare i moduli che verranno collegati a + 3,3 v oppure a +5 v, in base alle loro caratteristiche. In linea di massima verrà indicato quali funzionano a + 3,3 v; ***comunque si consiglia sempre di verificare il datasheet del modulo in oggetto***, specie se si hanno dubbi. In caso di incertezza, iniziare alimentando i moduli a 3,3 v. ***I moduli che funzionano a + 3,3 v possono essere rapidamente e irrimediabilmente danneggiati se alimentati a + 5v!***
- Ecco la lista dei jumper utilizzati per variare la tensione di alimentazione:
  - **VL1**: permette di scegliere la tensione di alimentazione dello zoccolo LD4;
  - **VL2**: permette di scegliere la tensione di alimentazione dello zoccolo DG3;
  - **VL3**: permette di scegliere la tensione di alimentazione dello zoccolo DG4;
  - **VL4**: permette di scegliere la tensione di alimentazione dello zoccolo DG5;
  - **VL5**: permette di scegliere la tensione di alimentazione dello zoccolo DG6;
  - **VL6**: permette di scegliere la tensione di alimentazione dello zoccolo DY3;
  - **VL7**: permette di scegliere la tensione di alimentazione dello zoccolo AN3;
  - **VL8**: permette di scegliere la tensione di alimentazione dello zoccolo AN4;
  - **VL9**: permette di scegliere la tensione di alimentazione dello zoccolo DG13;
  - **VL10**: questa è una eccezione: permette di alimentare con +5 v il piedino "SCW" dello zoccolo AN3. E' stato inserito per poter inserire un potenziometro su AN3.
- Quelli con sigla "**JPx**" (dove "x" è un numero progressivo), permettono di variare la porta di Arduino a cui collegare il modulo. Si è utilizzata questa soluzione, che all'inizio può sembrare più complessa che l'attribuzione di una porta univoca a un certo zoccolo, per rendere più flessibile l'uso della bs, evitare conflitti tra moduli e rendere più flessibile uno sketch (programma), utilizzando un numero maggiore di moduli. Ci saranno spiegazioni più dettagliate nella descrizione dei vari zoccoli. Ecco la lista:



JP1 permette di scegliere a quale porta digitale si collega BZ1: D3, D9 oppure D13.

  - **JP1**: permette di settare l'uscita audio alternativamente sulle porte digitali D3, D9 o D13 (default). Questo si è reso necessario perché alcuni sintetizzatori audio che usano le librerie "Mozzi" utilizzano delle porte di uscita (D3 o D9) che non sono facilmente variabili da programma;
  - **JP2**: permette di scegliere la porta di controllo per lo zoccolo DG8 (motore) tra la porta digitale D6 (default) e D9 (devono essere necessariamente porte PWM);
  - **JP3**: permette di scegliere la porta di controllo per lo zoccolo DG9 (servomotore) tra la porta digitale D5 (default) e D11 (devono essere necessariamente porte PWM);
  - **JP4**: permette di scegliere la porta di controllo per lo zoccolo AD1 (analogico-digitale) tra la porta digitale D2 (default) e D15 (fisica: A1, **vedi tabella corrispondente**);
  - **JP5**: permette di abilitare la porta logica A3 sullo zoccolo AN4. Si è scelto di attivare questa porta solo per alcuni moduli, per evitare inutili conflitti (**vedi zoccolo AN4**).



JP2 esegue il reset HW del modulo presente su DG4.

  - **JP6**: permette di scegliere la porta di trasmissione (Tx) tra D3 e D10 per gli zoccoli DG5 e DG6
  - **JP7**: permette di scegliere la porta di ricezione (Rx) tra D4 e D9 per gli zoccoli DG5 e DG6
  - **JP8**: permette di attivare la porta di ricezione (Rx) per gli zoccoli DG3 e DG4. Questo perché altrimenti restano sempre collegate delle resistenze (necessarie per ridurre la tensione) che falsano il funzionamento di D9 per altre attività.
  - **JP9**: permette di scegliere la porta di controllo per lo zoccolo AN2 (analogico) tra la porta A5 e A3 (default).

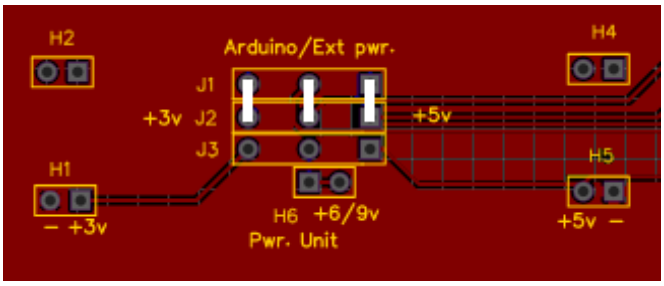
- Con la sigla "**Rsx**" sono presenti alcuni jumper che servono per eseguire un reset hardware di alcuni moduli, tra cui Arduino stesso. Ecco l'elenco:
  - **RS1**: reset dei moduli presenti sullo zoccolo DG3 (bluetooth);
  - **RS2**: reset dei moduli presenti sullo zoccolo DG4 (wi-fi);
  - **RS3**: reset hardware di Arduino Nano.

L'uso dei jumper verrà spiegato dettagliatamente nei paragrafi relativi ai singoli zoccoli a cui si riferiscono.

L'alimentazione elettrica di Arduino Nano "base"

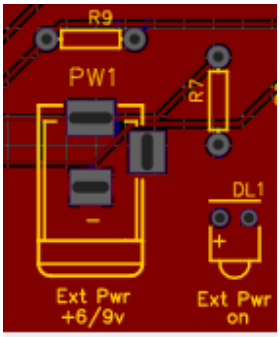
Per poter utilizzare al meglio la bs, si consiglia di leggere attentamente questo paragrafo.
Affinché la nostra basetta sperimentale possa essere attivata, è naturalmente necessario alimentarla. Sono stati previsti addirittura tre metodi di alimentazione:

- nel modo più classico, **attraverso il connettore mini USB** che la collega al computer. Questo è il metodo più comune di alimentare il microcontroller. E' molto comodo, perché per scaricare i programmi operativi su Arduino è necessario collegare la *bs* al computer, e in questo modo viene anche alimentata elettricamente. Ma ci sono anche delle controindicazioni in casi specifici: la presa USB fornisce una corrente di solo pochi mA, che diventano insufficienti quando si devono alimentare più moduli, o quando questi, come motori o sensori di gas, siano molto "voraci" di energia. Inoltre quando è stato caricato il programma, a volte per il test è necessario allontanarsi dalla postazione fissa. In questi casi, l'alimentazione attraverso il "cordone ombelicale" diventa inadeguata.



Posizione dei jumper per utilizzare i primi DUE metodi di alimentazione

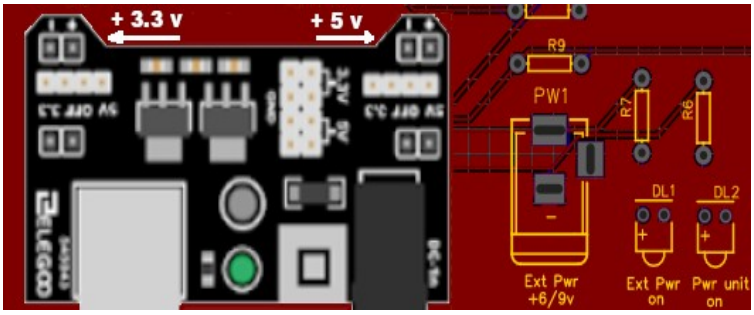
- Un secondo metodo di alimentazione, **consiste nel collegare** all'apposita presa coassiale (Ext Pwr) presente sulla parte sinistra inferiore della *bs* **un alimentatore esterno o una batteria** (vedi rettangolo figura), con una tensione compresa fra 6 e 9 volt e la possibilità di erogare 500/1000 mA. Questo secondo metodo di alimentazione ha il vantaggio di erogare una corrente decisamente più elevata di quella fornita dal connettore USB e di rendere la nostra *bs* indipendente dal computer. Quando decidiamo di usare questo sistema di alimentazione, si accende il led azzurro DL1, presso il connettore di alimentazione.
L'alimentatore porta la tensione compresa tra 6 e 9 v al piedino "VIN" di Arduino, che provvederà a ridurla a 5v, necessari per alimentare i suoi circuiti



Presa coassiale standard per l'alimentazione esterna della *bs*.

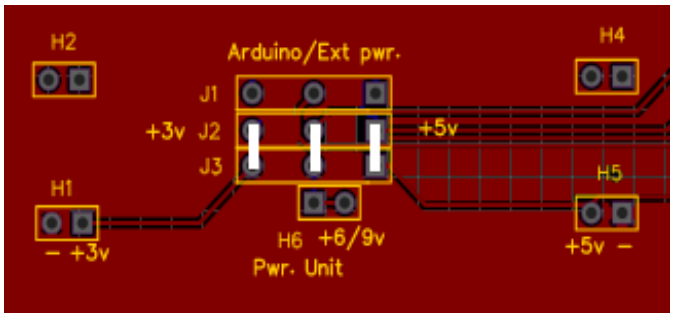
interni e i moduli collegati. Il diodo “DD1” (non visibile nell’immagine) serve ad impedire, in caso di alimentazione mista, che la tensione ricevuta da Arduino attraverso il computer possa interferire con quella prodotta dall’alimentatore esterno. La resistenza “R7” da 560 Ω impedisce che una corrente troppo alta raggiunga il led DL1, rischiando di bruciarlo in breve tempo.

Nota: sia per il primo tipo di alimentazione, con connettore USB, che con l’alimentatore esterno (seconda soluzione) i tre jumper di selezione del tipo di alimentazione devono essere tutti tassativamente spostati verso l’alto (Vedi la prima immagine della pagina precedente).
Il terzo tipo di alimentazione è più sofisticato, e richiede un modulo esterno da collegare alla nostra bs, ***dopo aver effettuato una piccola modifica***. In questo testo **si è usato il modulo 545043 della Elegoo**, presente nel kit “37 sensorkit V.2” della Elegoo stessa, che è stato trovato interessante, sia per il numero di moduli presenti, che per il rapporto qualità/prezzo. Nel caso interessi, i può trovare facilmente su internet. In questo caso, i jumper di selezione per l’alimentazione devono essere tassativamente tutti collegati verso il basso. (Vedi immagine in basso).



La power unit inserita nella sua sede (copre gli oggetti sottostanti)

Perché utilizzare anche un terzo tipo di alimentazione? L’alimentatore esterno, descritto nel secondo punto, fornisce una corrente più elevata, ma l’alimentazione dei moduli esterni, sia a +5v che a + 3,3v ***passa sempre attraverso Arduino***. Nei casi si richieda un maggiore assorbimento di corrente, potrebbe non essere in grado di fornirla, oppure essere sovraccaricato. L’utilizzo di questa power unit, studiata per alimentare motori e sistemi che richiedano una corrente maggiore, sembra proprio l’ideale. Ma mentre è stata progettata per alimentare periferiche esternamente ad Arduino, noi la utilizzeremo, con una semplice modifica, per alimentare anche il microcontroller attraverso la solita porta VIN, ***e di alimentare in modo del tutto separato i moduli aggiuntivi***, che da un punto di vista elettrico *saranno del tutto sganciati* dalle porte di alimentazione a +5v e + 3,3v di Arduino. In questo modo non si rischierà in alcun modo di danneggiare per sovraccarico i suoi delicati circuiti. Quando si collega l’alimentatore esterno alla power unit, si accende il led giallo DL2, che riceve la tensione di alimentazione attraverso la resistenza R6 (sempre da 560 Ω), per evitare che si bruci rapidamente.



Posizione dei jumper per utilizzare l’alimentazione attraverso la power unit Elegoo (o similare)

- ***Nota 1***: per inserire la power unit nella sua sede, bisogna porre attenzione a infilare correttamente le quattro file di piedini nelle sedi corrette.

- ***Nota 2***: guardando l’immagine relativa alla power unit (prima immagine di questa pagina),

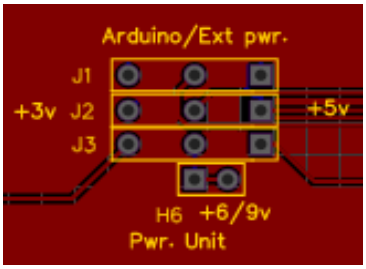
si vede che in alto è possibile selezionare in uscita la tensione desiderata. ***E' tassativo***, pena il possibile danneggiamento o il mancato funzionamento dei moduli esterni di Arduino, ***che iil ponticello n alto a sinistra sia settato su 3,3v, e che quello di destra sia settato a 5v, altrimenti moduli saranno alimentati con tensioni invertite rispetto a quelle desiderate.***

- ***Nota 3:*** come indicato prima , la power unit di Elegoo è stata disegnata per alimentare *solamente* le periferiche, ma noi dovremo usarla *anche* per alimentare Arduino. Sarà quindi necessario effettuare una piccola saldatura, collegando un cavetto con un terminale femmina che dal + di alimentazione della power unit porti tensione alla nostra *bs.,* collegandosi al piedino maschio "Pwr 6/9v". Nel caso non venga eseguita questa operazione, Arduino Nano non verrà alimentato. Vedi immagine nella pagina successiva.

I jumper dell'alimentazione

Vedremo ora il significato dei jumper di alimentazione J1, J2, J3 e la presa di alimentazione H6.

- **Jumper verticali sulla sinistra:** seleziona l'alimentazione delle periferiche a **+ 3,3 v.** Quando è selezionato verso l'alto, l'alimentazione è fornita da USB o direttamente dall'alimentatore; se è verso il basso, viene alimentato da una power unit, tipo Elegoo.



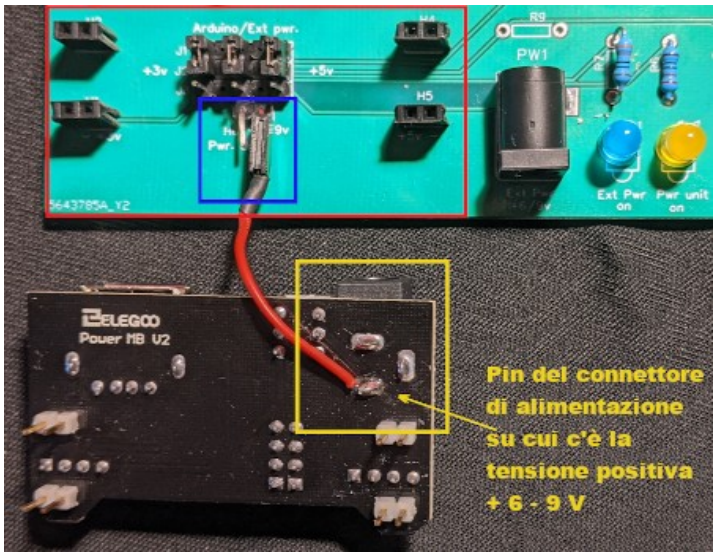
I jumper per la scelta del metodo di alimentazione della bs.

- **Jumper verticali, in posizione centrale**: seleziona l'alimentazione di Arduino attraverso la porta VIN. Quando è selezionato verso l'alto, l'alimentazione è fornita da USB o direttamente dall'alimentatore; se è verso il basso, viene alimentato dalla power unit tipo Elegoo.
- **Jumper verticali, sulla destra**: seleziona l'alimentazione delle periferiche a **+5v.** Quando è selezionato verso l'alto, l'alimentazione è fornita da USB o direttamente dall'alimentatore; se è verso il basso, viene alimentato da Elegoo Power unit (o similare). N.b: questo modulo è presente anche nel kit "45 in 1 kit"
- **Presa H6:** questo un connettore maschio, infatti ha una funzione diversa; non serve per effettuare una selezione, ma per fornire la tensione compresa tra 6 e 9 v alla porta VIN di Arduino, prelevandola direttamente dall'ingresso della power unit Elegoo (o simile) con un cavo che termina con un connettore femmina e collegandosi al connettore maschio presente sulla *bs*.

Come collegare la power unit alla *bs*.

La power unit si connette alla *bs* inserendo i pin di alimentazione ai connettori H1, H2 (3,3 v) e H4, H5 (5 v). Naturalmente prima è necessario spostare i jumper nella posizione corretta. Vedi immagine relativa. Fare attenzione a connetterli correttamente.

Come anticipato, prima di inserire la Power unit sulla bs, è necessario fare una piccola modifica su di essa, saldando un corto cavetto al polo positivo di alimentazione della Power unit stessa. All'altro capo del cavo, è necessario collegare un corto connettore femmina, che andrà inserito nel connettore maschio (denominato "H6"), presente sulla bs. Questo serve per fornire tensione al piedino "VIN" di Arduino Nano. ***Senza questa semplice operazione, Arduino non verrà alimentato attraverso la Power unit.***



Collegamento dell'alimentazione a 6/9 v dalla Power unit alla bs.

N.b.: il connettore H6 ha due connettori maschio. Collegare il cavetto proveniente dalla Power unit su uno dei due, indifferentemente, in quanto sono collegati insieme.

Nota: è possibile alimentare Arduino Nano ***contemporaneamente*** attraverso la presa USB e il connettore "Ext Pwr, oppure presa USB + alimentatore esterno tipo Elegoo; sarà poi Arduino che sceglierà automaticamente da dove prelevare l'alimentazione; quindi non ci saranno conflitti o problemi.

Alcune considerazioni sull'alimentazione elettrica di Arduino Nano 33 BLE e 33 IoT

Come anticipato nel paragrafo relativo alla presentazione di Arduino Nano 33 BLE e 33 IoT, pur presentando una piedinatura compatibile con la versione "base", essi differiscono per le possibilità di alimentazione.

Arduino Nano "base" fornisce in uscita due livelli di alimentazione per i moduli esterni: +3,3v e +5v e naturalmente accetta in ingresso segnali provenienti dai moduli stessi sia a +3,3v e a +5v.

Arduino Nano 33 BLE e 33 IoT forniscono in uscita *nativamente* solo la tensione a +3,3v e accettano in ingresso *solo* segnali provenienti dai moduli esterni a +3,3v. Segnali in ingresso a +5v rischiano di danneggiare irreversibilmente questi nuovi modelli.

Tuttavia, probabilmente per mantenere una certa retrocompatibilità con il modello base, in alcuni casi è possibile fornire in uscita anche la tensione a +5v.
Ecco come:

- effettuare una saldatura a ponte tra i due elettrodi denominati "VUSB" presenti nella parte inferiore di Arduino Nano (vedi immagine);
- alimentando esclusivamente il microcontroller via USB. Perciò, anche se si è effettuata la saldatura, ma si alimenta Arduino Nano attraverso la porte "VIN", sul piedino dei +5v non si otterranno i +5v!

Il punto dove effettuare la saldatura

A questo punto, **sembra inutile fornire in uscita la tensione di +5v**, anche perché i moduli che funzionano a questa tensione, portando segnali in ingresso a questa tensione verso il microcontroller, potrebbero danneggiarlo irreversibilmente, se non si prendono precauzioni per fornire sulle porte di Arduino una tensione massima di 3,3v! ***Perciò si consiglia caldamente di non effettuare questo ponticello.***

E' possibile danneggiare Arduino Nano 33 BLE e 33 IoT con la nostra bs versione 0.6.1?

Se *non* abbiamo effettuato il ponticello su VUSB e:

- se alimentiamo un Nano 33 con la USB, forniremo in uscita solo +3,3v, quindi non rischieremo di danneggiarlo, ***però non alimenteremo i moduli presenti sugli zoccoli che forniscono solo +5v***, come per esempio DG1, AN1, AD1, ecc,
- se alimentiamo un Nano 33 attraverso VIN con un alimentatore esterno, si comporterà esattamente come nel caso precedente e non ci sono rischi;
- se alimentiamo un Nano 33 per mezzo di un power esterno, quale il modulo 545043 della Elegoo, l'alimentazione dei moduli non sarà fornito da Arduino, come nei casi precedenti, bensì direttamente dalla power unit. Perciò in questo caso, se abbiamo spostato correttamente i ponticelli J1, J2, J3, forniremo in uscita entrambe le tensione e se collegheremo per esempio su DG1 un modulo (che funziona a +5v), **rischiamo di danneggiare il nostro Nano 33 con il segnale a +5v in ingresso**!

Se abbiamo effettuato il ponticello su VUSB e:

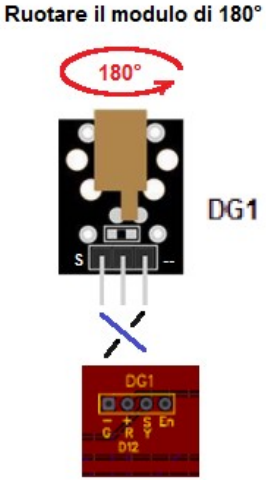
- se alimentiamo un Nano 33 con la USB, forniremo in uscita entrambe le tensioni e se collegheremo per esempio su DG1 un modulo (che funziona a +5v), **rischiamo di danneggiare il nostro Nano 33 con il segnale a +5v in ingresso**!!
- se alimentiamo un Nano 33 attraverso VIN con un alimentatore esterno, non rischieremo di danneggiarlo, perché in uscita ci sarà solo una tensione a +3,3v; però non alimenteremo i moduli presenti sugli zoccoli che forniscono solo +5v, come per esempio DG1, AD1, ecc;
- se alimentiamo un Nano 33 per mezzo di un power esterno, quale il modulo 545043 della Elegoo, l'alimentazione dei moduli non sarà fornito da Arduino, come nei casi precedenti, ma dalla power unit. Perciò in questo caso, se abbiamo spostato correttamente i ponticelli J1, J2, J3, forniremo in uscita entrambe le tensioni e se collegheremo per esempio su DG1 un modulo (che funziona a +5v), **rischiamo di danneggiare il nostro Nano 33 con il segnale a +5v in ingresso**!

Perciò da come si evince leggendo questo paragrafo, possiamo utilizzare i Nano 33 sulla bs versione 0.6.1, ma dobbiamo essere prudenti, non effettuare il ponticello su VUSB e non usare il power unit tipo Elegoo. Per contro però, ***potremo utilizzare solo i moduli che si collegano sugli zoccoli che forniscono in alternativa la tensione a +3,3v***. Vedi la lista.
Spero in questo modo di avere chiarito il problema relativo all'alimentazione dei nuovi moduli.

KY- 022

Come collegare i moduli sulla bs.

Sulla bs trovano posto un notevole numero di zoccoli, di cui si fornisce un breve elenco:
zoccoli (socket) digitali
per moduli: da DG1 a DG13; BZ1, RL1, RL2
per display: DY1, DY2 e DY3. Per i led: da LD1 a LD5.
Zoccoli analogici/digitali:
per moduli: AD1a/b, AD2, AD3
Zoccoli analogici:
per moduli: da AN1 a AN5
pulsante integrato nella bs: BT1



KY-008

Questa scelta così ampia è stata decisa per dare la possibilità di collegare un gran numero di moduli alla *bs*, senza l'uso di cavi di collegamento, o limitandoli al massimo. Sopra o sotto ogni zoccolo sono indicate le lettere relate ai vari connettori, nello stesso ordine di quelli presenti sui piedini dei moduli corrispondenti. ***Le lettere presenti sul modulo devono corrispondere a quelle dello zoccolo***. (vedi figure). E' necessario avere ogni cura nel collegare il modulo allo zoccolo: inserendolo in senso inverso o non appropriato, si rischia di danneggiare irrimediabilmente il sensore o Arduino Nano stesso!

Abbiamo inserito come esempio il sensore KY-022 (un ricevitore a raggi infrarossi). Immediatamente sotto i piedini di collegamento sono indicate le sigle "G, R, Y", in cui "G" corrisponde al polo "-", "R" al polo positivo e "Y" al segnale digitale. Come si vede dall'immagine stilizzatala sopra lo zoccolo "DG1", i primi tre contatti sulla sinistra corrispondono perfettamente alle lettere presenti sul sensore; quindi il sensore può essere installato seguendo l'ordine dei piedini. Il quarto contatto "EN", non viene utilizzato con questo sensore. La scritta di fianco allo zoccolo "D12", indica la porta digitale di Arduino che verrà utilizzata. In questo modo, anche solo guardando le scritte sulla *bs*, si hanno tutte le informazioni necessarie non solo per il collegamento, ma anche per la programmazione.

Vediamo ora un altro caso, sempre per lo zoccolo DG1, in questo caso relativo al led laser KY-008. La serigrafia sul modulo riporta da sinistra a destra i seguenti codici: "S, +, -". Mentre sullo zoccolo appare: " -, +, S". ovvero i codici sono gli stessi, ma in ordine inverso. ***Quindi il modulo andrà inserito sullo zoccolo ruotato di 180°, in modo che le scritte coincidano***. Tenuto conto di queste semplici regole, non ci saranno problemi di connessione.

Nota: acquistando i kit "37 in 1 kit" o "45 in 1 kit" a prezzi molto competitivi, mi sono accorto che alcuni moduli non seguono effettivamente la piedinatura riportata sulla serigrafia; probabilmente i componenti elettronici sono stati assemblati su basette progettate per accogliere altri tipi di componenti. Si prega di fare attenzione, altrimenti si rischierà di danneggiare il componente stesso. ***Effettuare quindi accurati controlli, specialmente la prima volta che si utilizza un nuovo modulo***. Se inserendolo, si vede affievolirsi la luce dei led presenti su Arduino Nano, oppure il programma non dà risultati, anche se pare corretto, togliere ***immediatamente*** corrente alla bs e ricontrollare attentamente il modulo in oggetto.

Come trasformare le porte analogiche in digitali.

Le porte digitali sembrano molte, ma in alcuni casi, quando si collegano per esempio i display LCD o TFT, le porte disponibili per i moduli si riducano drasticamente.

Anche in alcuni progetti presenti in questo manuale, abbiamo optato per questa soluzione, ovvero di utilizzare alcune delle porte nativamente analogiche come digitali.
Ecco la tabella da utilizzare:

Analogica		**Digitale**
A0	→	(D) 14
A1	→	(D) 15
A2	→	(D) 16
A3	→	(D) 17
A4	→	(D) 18
A5	→	(D) 19

Come si vede dalla tabella, le prime sei porte analogiche possono essere utilizzate come digitali.

Ovviamente, in questo caso, ***non possono essere usate contemporaneamente come analogiche***.

Nella sezione digitale, la "D" prima del numero della porta è tra parentesi, perché nel programma sarà sufficiente scrivere il numero della porta in oggetto.

Ecco un esempio di trasformazione, utilizzando il pulsante “BT1”, integrato sulla *bs* non come porta analogica “A2”, ma come digitale (D)16:

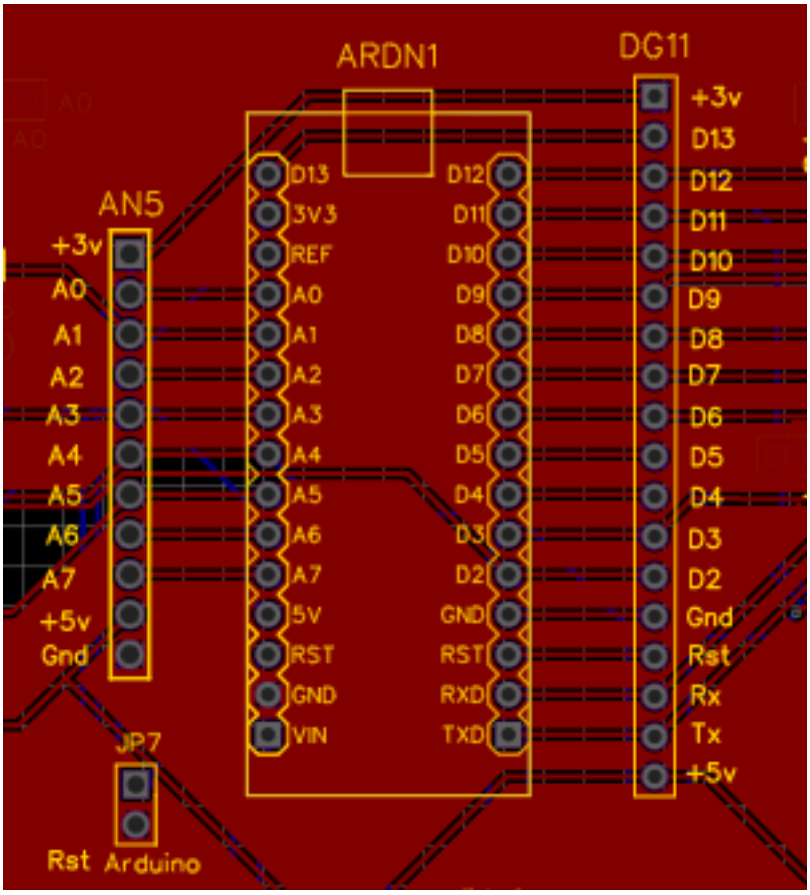
Originale: analogico	**Modificato: digitale**

`const int buttonPin = A2;`
 	
`void loop() {`
`  buttonState = analogRead(buttonPin);`
	`const int buttonPin = 16;`
	
	`void loop() {`
	`  buttonState = digitalRead(buttonPin);`

Come si vede nella riga successivava a “void loop()”, per intercettare correttamente il segnale proveniente da questo modulo, sarà necessario trasformare la funzione “analogRead” in digitalRead”.

Sezione III

Gli zoccoli (sockets) della *bs* con alcuni programmi



Programmi di test per la basetta sperimentale

Dopo aver costruito la basetta sperimentale, si ha il desiderio di utilizzarla e verificarne il funzionamento... In questa sezione sono raccolti alcuni programmi per sondarne le possibilità.
La maggior parte sono semplicemente programmi didattici, ma ce ne sono alcuni di un certo pregio.

Arduino offre 13 porte digitali e 7 analogiche. Come si è visto nella sezione dedicata alla descrizione della basetta, le porte analogiche da A0 a A5 possono essere trasformate, quando necessario, in porte digitali, da D14 a D19. Anche nella nostra basetta si è sfruttata alcune volte questa opportunità. Per approfondire, vedi i progetti relativi ai connettori AD1 e AD2.

Il problema della gestione delle porte è stato uno dei principali crucci nel disegnare questa basetta, e sono state fatte numerose modifiche dalla prima versione del progetto fino a quello attuale. Si è cercato di sfruttare al meglio le possibilità; si potranno effettuare ulteriori miglioramenti, ma purtroppo temo non ci sia una unica situazione che si possa definire ottimale.

Come ho già detto, i progetti presentati sono dei semplici esercizi, e possono essere modificati e sviluppati a proprio piacimento, il limite sono solo la propria fantasia, la potenza di Arduino e… i conflitti tra le porte utilizzate. Vedi a questo proposito il paragrafo dedicato ai conflitti tra le porte nel file esterno "tabelle".

In Internet si può trovare una selezione sterminata di progetti, digitando per esempio su Google "arduino project". In modo assolutamente indicativo (e senza alcuna responsabilità da parte mia sulla serietà dei siti in oggetto e neppure sulla qualità dei progetti mostrati) posso indicare i siti che ho trovato più interessanti, e da cui ho tratto molti programmi, poi adattati per la nostra *bs*:
https://projecthub.arduino.cc/
https://www.instructables.com/circuits/arduino/projects/
https://www.hackster.io/arduino/projects
https://howtomechatronics.com/arduino-projects/
https://www.electronicshub.org/arduino-project-ideas/

Su youtube ho trovato moltissimo materiale, vorrei segnalare principalmente i video di Paolo Aliverti, chiari e utili. Ecco il link al suo progetto: video di Paolo Aliverti
Cliccando sotto al video su "Mostra altro", si accede alla sua ricca lista di video, listati e anche un paio di libri che si possono scaricare sia gratuitamente che offrendo una donazione.
Tra gli scopi di questo manuale non c'è quello di introdurre l'uso di Arduino per principianti, tuttavia si troverà qualche rapida informazione isulla IDE e sulle librerie di Arduino.
Per istruzioni più chiare e dettagliate, usare i seguenti link.
Istruzioni per la IDE di Arduino:

- per Linux: arduino.cc/linux
- per Mac OS: arduino.cc/mac
- per Windows: arduino.cc/windows

Per scaricare la IDE più aggiornata: arduino.cc/download
la guida aggiornata: arduino.cc/guide

Attenzione: per visualizzare/scaricare i programmi che verranno presentati, è necessario essere connessi in Internet. Nel caso si sia spesso off line, si può scaricare la cartella in formato "zip"contenenti tutti i programmi tutti i programmi (circa 75 Mb). In questo modo però non verranno lanciati direttamente dal manuale e sarà necessario selezionarli manualmente accedendo alla sottocartella dello zoccolo corrispondente.

Attenzione: questi link, come i programmi che troverete nella presente sezione, sono presentati a scopo puramente indicativo, senza alcuna responsabilità dell'autore per quel che riguarda il contenuto dei siti stessi, la loro sicurezza, la qualità dei programmi, né da alcun danno che possa derivare a persone o cose dalla implementazione dei programmi (presenti in questa selezione o sul mio sito internet) e dal loro utilizzo.

Una lista di alcuni programmi interessanti proposti in questa sezione

La raccolta dei programmi per Arduino Nano è iniziata molto prima che decidessi di realizzare questa bs, perciò ho perso il link a molti di essi. Mi scuso quando gli autori non sono citati con un link; qualora li ritrovassi, li inserirò nelle prossime revisioni del presente manuale.

I programmi sono stati testati singolarmente e a fondo; tuttavia non si può escludere a priori che ci siano malfunzionamenti o che non diano i risultati sperati...

Ecco una breve lista (sicuramente non esaustiva) di quelli che ho trovato più significativi:

- Una stazione meteorologica digitale: con un modulo BMP280 e pochi altri componenti, costruiremo una stazione meteorologica, molto simile a quelle tradizionali: troveremo la pressione atmosferica, temperatura, umidità e l'altezza approssimativa del luogo.

- Il battito del cuore: uno strumento che misura il battito del cuore, molto carino anche graficamente; si ricorda però che non è un apparecchio medico!

- Creare un ambiente confortevole: con un sensore DHT11, si potrà monitorare la temperatura, e all'occorrenza accendere la caldaia o un condizionatore.

- Un preciso GPS: uno strumento molto interessante: oltre a latitudine e longitudine, fornisce molte informazioni interessanti.

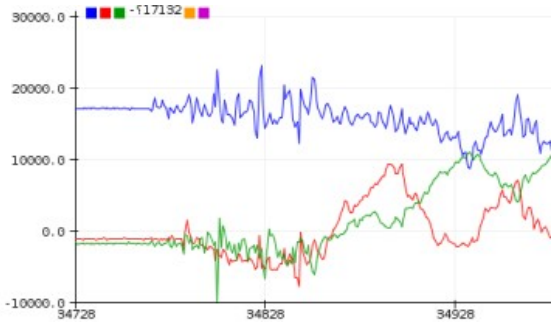
- Una serratura a doppia sicurezza: questa serratura richiede una password, lunga a piacere e dopo la presenza della corretta card Rfid.

- Un antifurto sperimentale: Un antifurto composto da cinque sensori diversi, per approfondire la tecnica. I dati vengono mostrati su di un dispaly Tft.

Le informazioni del GPS

- Un piccolo sintetizzatore musicale: Arduino con le giuste librerie, diventa anche un piccolo sintetizzatore musicale!

- Un semplice sismografo: non verrà certo usato dai laboratori che si occupano dei terremoti, ma comunque è molto divertente!



Adattamento dei programmi scritti da terzi per la basetta sperimentale

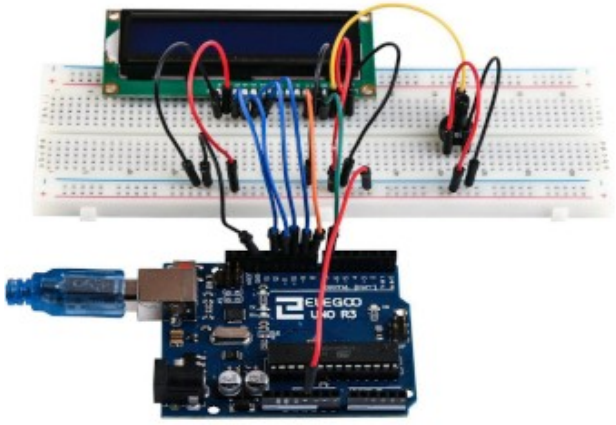
Uno dei principali problemi a chi si approccia per la prima volta a questa basetta, è adattare i propri programmi, come scritti da altri, che si possono trovare su internet, libri, riviste, ecc.

Già, perché chi scrive un singolo programma, può usare tutte le porte a disposizione, mentre noi dobbiamo sottostare ai limiti imposti dalla basetta stessa, che in cambio della possibilità di poter sperimentare centinaia di progetti quasi senza l'uso di cavi, presenta una oggettiva limitazione all'uso delle porte di Arduino. Proverò a dare qualche indicazione con un paio di esempi esplicativi.

Primo esempio – Display LCD 1602

Ecco il programma originale per gestire il dispaly LCD 1602:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}
```



Questo è il progetto realizzato su breadboard:
notare il numero di cavi utilizzati →

Il nostro dispaly deve essere alloggiato sul connettore DY2, che usa le seguenti porte di Arduino:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
VCS	VDD	VO	RS	RW	E	D0	D1	D2	D3	D4	D5	D6	D7	A	K
-	+5v	-	D2	-	D3	/	/	/	/	D4	D5	D6	D7	+5v	-

porte che sono segnate in rosso, ovvero D2, D3, D4, D5, D6, D7.

Quindi il nostro programma, sarà modificato così:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Hello, World!");
}
```

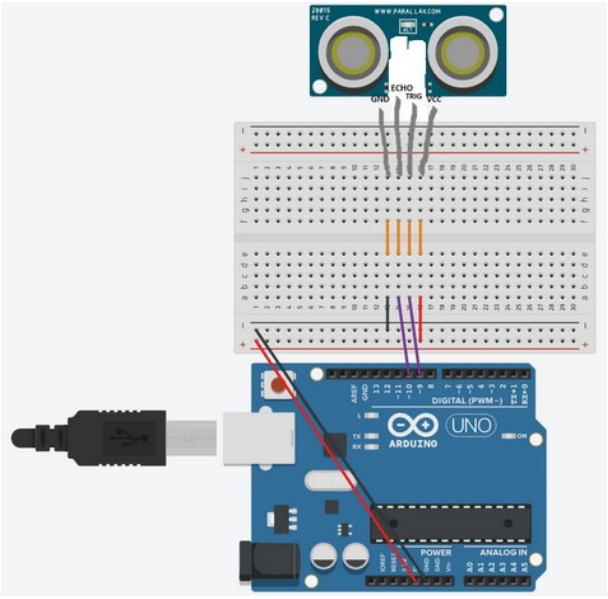
Si noti la totale assenza di cavi… →

Secondo esempio – sensore ultrasonico

Il programma originale usa le porte digitali D9 e D10:

HC_SR04.ino

```
/*
 * HC-SR04 example sketch
 *
 * https://create.arduino.cc/projecthub/
 *
 * by Isaac100
 */

const int trigPin = 9;
const int echoPin = 10;

float duration, distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
```



Sulla nostra basetta, il sensore ultrasonico HC-SR04, alloggia sul connettore DG7:

1	2	3	4
vcc (+5v)	TRIG	ECHO	GND
+	d11	d8	–

Trig usa la porta D11 ed Echo la porta D8. Quindi il programma è modificato nel modo seguente:

ultra_1

```
/*
 * HC-SR04 example sketch
 *
 * https://create.arduino.cc/proje
 *
 * by Isaac100
 */

const int trigPin = 8;
const int echoPin = 11;

float duration, distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
```

Questa procedura può essere ripetuta per tutti i moduli proposti. All'inizio ci vorrà un po' di pazienza, ma in breve tempo si diventerà capaci di adattarli (e anche creare programmi autonomi, o ampliare quelli testati) con rapidità.

Attenzione: i moduli o i componenti che usano le porte PWM, D3, D5, D6, D9, D10, D11 ovvero quando si devono ottenere variazioni continue di luminosità (led), di frequenza (buzzer), di velocità (motori e servomotori) è necessario in caso di modifiche, che si continui ad utilizzare sempre delle porte PWM, pena il non funzionamento del programma.
Nella basetta vengono usate le porte PWM per i Led (LD1, LD2, LD3); per lo zoccolo DG8 (motore elettrico); per DG9 che controlla un servomotore; per il buzzer, quando settato su D3, qualora si voglia utilizzare il programma "

Come leggere la tabella sintetica relativa ai programmi

Ogni programma che troverai in questo manuale, è accompagnato da una tabella che spiega sinteticamente tutti gli aspetti fondamentali per poter usare lo sketch in oggetto.

Come anticipato nell'introduzione, tranne che per casi particolari non si daranno spiegazioni dettagliate sul funzionamento del programma, in quanto non è lo scopo di questo manuale.

Lo schema delle schede è abbastanza intuitivo, comunque ecco una breve introduzione.

Nome del programma:	Sensor_2a
Porte: A0 (analogico) **D15** (digitale) *	**Modulo principale:** KY-024, KY-025, KY-026, KY-028, KY-036, KY-037, KY-038. Vedi descrizioni nella pagina precedente.
Porte:	**Comp. Accessori:**
D11 (digitale)	Led LD1 - verde
D9 (digitale)	Led LD3 - rosso
D9, D10, D11 (digitale)	Led LD4 – tre colori (opzionale)
D9 (digitale)	Relay KY-019 su RL2 (opzionale)
D13	Buzzer - KY-006 (KY-012) su BZ1
D2/./D7	Display LCD 1602 (16 caratteri per 2 righe) su DY2
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale i valori analogici (sopra 500 non c'è attività) e quello digitale: 0 = nessuna presenza; 1 = presenza attività. Il led rosso è acceso quando si è rilevato un'attività, in alternativa al verde, quando essa non è presente.
Librerie necessarie:	LiquidCrystal.h
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts e almeno 500 mA * Come da introduzione, questo programma usa la porta digitale (D)15. Spostare il jumper su D15
Link:	nessuno

La tabella sintetica relativa a un programma.

Nome del programma: crea un collegamento con la pagina del sito "projects.html". Abitualmente il nome del programma corrispondente appare nella rima riga in alto della pagina. Sarà possibile visualizzarlo o scaricarlo in formato zip. Nel caso si sia spesso fuori rete, tutti i progetti possono essere scaricati in blocco nella cartella zippata "Projects.zip". In questo caso purtroppo si perde il collegamento automatico con il manuale, ed è necessario aprirli manualmente. Espandendo la cartella, il programma interessato si troverà facilmente; infatti le sottocartelle sono in una sequenza gerarchica, ovvero ***zoccolo → modulo → programma***, proprio la stessa organizzazione del manuale.

Porte (del) Modulo principale: nella maggior parte dei programmi è indicato, a volte arbitrariamente, un modulo principale, su cui si incentra il programma. In questo riquadro troverai le porte utilizzate dal modulo principale, che alloggia sullo zoccolo relativo al paragrafo (in questo caso AD1).

Porte (dei) componenti accessori: in molti sketch leggermente più complessi, è possibile che oltre il modulo principale ce ne siano atri accessori, come ulteriori sensori, attuatori, displa, ecc. In questo riquadro sono indicate le porte utlizzate, i nomi dei moduli e lo zoccolo su cui sono alloggiati.

Monitor/plotter seriale: questa riga indica se si usa il monitor o il plotter seriale e la velocità di collegamento. Il monitor seriale è molto utile per sapere come si sta svolgendo il programma, visualizzando i valori delle variabili intercettate dal comando "Serial.print".

Scopo del programma: una descrizione sintetica dello scopo e dell'uso dello sketch.

Librerie necessarie: nel caso siano necessarie delle librerie per per il regolare funzionamento dello sketch, sono elencate in questo riquadro. Nel caso non fossero installate, possono essere scaricate all'interno della IDE, da strumenti → gestione libreria; oppure cercandole su internet; o ancora scaricando la cartella zippata "libraries.zip", salvarle nella cartella "libraries" in cui è installata la IDE di Arduino e scompattare il file. In questo modo si avranno a disposizione tutte le librerie necessarie per eseguire i programmi presenti in questo manuale. Vedi la sezione "librerie".

Note: nel caso ci sia qualche condizione particolare da evidenziare.

Link: ho raccolto in Internet la maggior parte dei programmi che ho modificato e poi utilizzato in questo manuale nel corso di parecchi mesi, e all'inizio non pensavo di usarli per questo progetto, ma solo per mio uso personale e non ho trascritto i siti di origine. Per cui purtroppo ci sono tantissime lacune nell'attribuzione dei programmi usati. Me ne scuso moltissimo, cercherò di riempire le lacune, e se qualcuno sa darmi i riferimenti mancanti, li inserirò con piacere.

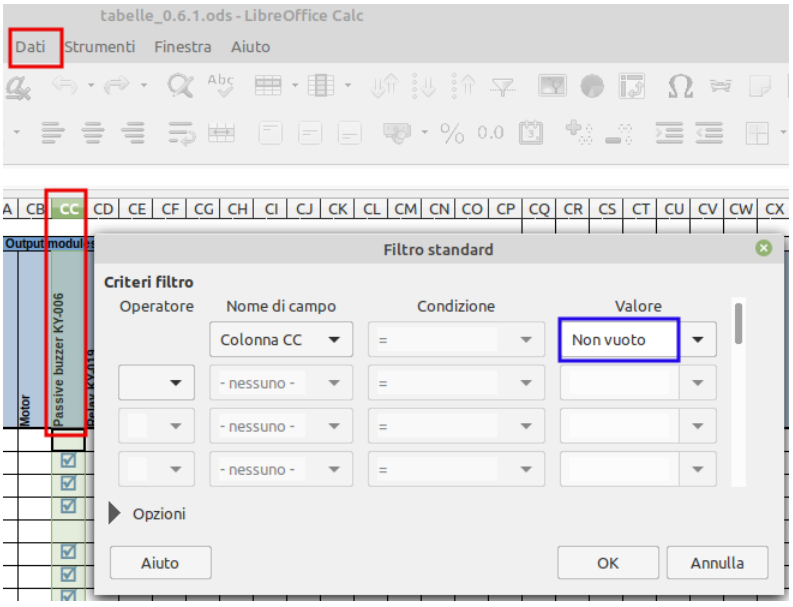
Ho alcuni moduli per Arduino. Come faccio a trovare i programmi che posso realizzare con essi?

Per la creazione di questo manuale ho utilizzato circa ottanta moduli diversi, ed è naturale che tu non li abbia tutti, o che cerchi rapidamente un programma particolare. Come fare?
Un grande aiuto è fornito da un file esterno, che si chiama "tabelle.0.6.1.ods" (in formato libero, es. LibreOffice/OpenOffice), oppure "tabelle.0.6.1.xlsx" (in formato Microsoft Excel), in cui ci sono molte informazioni interessanti (vedi paragrafo).

In particolare, vorrei segnalare due pagine molto utili di questa tabella:

- applicazioni
- socket

Applicazioni. In questa pagina, troverai sull'asse X tutti i moduli che sono stati usati e sull'asse Y tutti i programmi presenti in questo manuale. Se hai un minimo di esperienza con LibreOffice Calc o con Microsoft Excel, non farai fatica a destreggiarti. Immaginiamo che tu abbia a disposizione il modulo "passive uzzer" KY-06 e desideri sapere in quali programmi è utilizzato. Utilizzando il ***filtro automatico*** (dati → filtro automatico) oppure il più accurato ***filtro standard*** (dati → altri filtri → filtro standard), sarà possibile visualizzare tutti i programmi che contengono quel particolare modulo.



Filtro standard

Il filtro standard: dati → altri filtri → filtro standard

Scegliere la colonna del modulo di cui si vuol sapere in quali programmi viene utilizzato.
In "nome campo" apparirà il nome della colonna selezionata "CC", corrispondente a "Passive Buzzer KY-006". In "valore" inserire "Non vuoto" e premere "Ok"-

Dopo aver premuto "OK", apparirà la lista di tutti i programmi che usano il "Passive Buzzer KY-006". Questa procedura vale per qualsiasi modulo, o anche per selezioni multiple.

Ecco uno stralcio del risultato, perché KY-006 si usa in tanti progetti:

3	Socket	Progetti	Tilt Switch KY-020	Ultrasonic HC-SR04 KY-050	Water level sensor Sen18	Weather st. BMP280	433 MHz transmitter	Active buzzer KY-012	Laser led emit KY-008	IR emission KY-005	Motor	Passive buzzer KY-006	Relay KY-019
4	AD1	Sensor_1											☑
5		Sensor_2						☑				☑	☑
6		Sensor_2a						☑				☑	☑
7		Sensor_3						☑				☑	☑
9		Gas_2						☑				☑	☑
10		Gas_3						☑				☑	☑
11		Gas_4						☑				☑	
16	AD2	Joy_2						☑				☑	☑
28	AN1	Water sensor_3			☑			☑				☑	☑
31		Switch_1	☑					☑				☑	☑
35	AN2	Pot_3						☑				☑	
36		Pir_1a						☑				☑	
65	AN5	Granular synth						☑				☑	
66		Granular synth_1						☑				☑	
72	DG1	Button_2						☑				☑	
79		Hall_reed_2						☑				☑	
88		Mercury_Tilt_Shock_Tap2	☑					☑				☑	
90		Photo_int_2						☑				☑	

Risultati per K-006

Il passive buzzer viene utilizzato in molti programmi… Nella colonna "Progetti" appare la lista degli sketch (divisi per socket) che lo usano per ottenete qualche segnalazione sonora. Naturalmente il procedimento è valido per qualsiasi modulo (o anche gruppi di moduli).

C'è anche una legenda per interpretare il significato dei colori dello spunto.

Legenda	
☑	Componente secondario
☑	Componenti secondari in alternativa
☑	Componente principale
☑	Moduli principali in alternativa

Socket. In questa ulteriore pagina si trovano le tabelle (riportate anche nella presentazione dei vari zoccoli), relativa ai vari moduli che si possono inserire in un determinato zoccolo.

Lista zoccoli - moduli

Socket AD1/a (digital/analogic)			1	2	3	4
Description	Code	Note	DO	+ (+5v)		AO
Big Sound	KY-037 - HW485		d2 (d15)	+	-	a0
Small Sound	KY-038 - HW496					
Linear Hall	KY-024 – SS49E – HW509					
Metal Touch	KY-036 - HW494					
Magnetic Spring	KY-025 - HW484					
Flame	KY-026 – HW491					
Digital Temperature	KY-028 – HW503					

Socket AD1/b (digital/analogic)			1	2	3	4
Description	Code	Note	AO	DO	GND	VCC (+5v)
Gas sensor	MQ2 - KY030		a0	d2 (d15)	-	+
Gas sensor	MQ3					
Gas sensor	MQ4					
Gas sensor	MQ5					
Gas sensor	MQ6					
Gas sensor	MQ7					
Gas sensor	MQ8					
Gas sensor	MQ9					
Gas sensor	MQ135					
Soil moisture sensor		Ruotare di 180° rispetto ai sensori del gas				

Lista di alcuni zoccoli (in ordine alfabetico)

La pagina dei socket.

In questa pagina si puo' vedere la lista completa degli zoccoli e dei moduli che si possono montare su ognuna di essi.

***

Nel file "tabelle.0.6.1" si possono trovare molte altre informazioni interessanti, **quali una cross reference tra i codici dei vari moduli, alcuni suggerimenti relativi ai kit di moduli, la tabella delle porte utilizzate dai vari zoccoli e gli eventuali conflitti.**

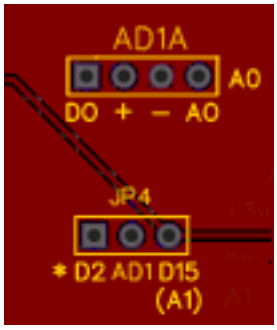
Gli zoccoli per moduli che usano sia porte digitali che analogiche.

Sulla nostra *bs* ci sono quattro zoccoli adatti per moduli che usino sia porte analogiche che digitali. Essi sono AD1a, AD1b, AD2 e AD3, presenti sulla parte sinistra della basetta stessa.

Come collegare i moduli agli zoccoli. Sulla *bs* i vari pin (4 per AD1a/AD1b, 5 per AD2, 11 per AD3), sono contrassegnati con dei codici, gli stessi che si trovano sul modulo che si desidera collegare. Naturalmente le sigle devono corrispondere, pena il non funzionamento del progetto e nei casi più gravi, il danneggiamento del modulo stesso o addirittura di Arduino.

Lo zoccolo AD1/a

Nella versione precedente della *bs*, AD1 indicava un singolo zoccolo, su cui alcuni moduli sarebbero stati collegati in modo nativo, mentre un'altra nutrita serie di moduli (nello specifico i rilevatori di gas), pur utilizzando le stesse connessioni, avevano i piedini in sequenza diverse (come sarebbe bello se ci fosse uno standard!), per cui dovevano essere collegati con un connettore a quattro cavetti. In questa successiva versione, ho pensato di sdoppiarli, in modo che tutti questi moduli si possano collegare direttamente sullo zoccolo adeguato: i sensori ambientali su AD1a; quelli sensibili ai gas su AD1b.



AD1a si trova nella parte alta a sinistra della bs, vicino alle porte analogiche di Arduino. La porta analogica usata è A0.
Subito sotto di esso si trova il jumper per selezionare quale porta digitale da usare: D2 (default) oppure D15 (A1).
Questa scelta è stata fatta per evitare il conflitto di D2 con i display DY2.
Selezionando invece la porta D15, essa va in conflitto con lo zoccolo AN1, che usa la porta analogica A1.
AD1a e AD1b usano le stesse porte, per cui non si possono usare contemporaneamente. Quante scelte difficili...

Nota: tutti i moduli che montano sullo zoccolo AD1b, possono essere collegati allo zoccolo AD1a, utilizzando un connettore a quattro cavetti per adattarsi alla piedinatura diversa.
Su questo zoccolo, avendo a disposizione una porta analogica, possono essere collegati anche tutti i moduli che alloggiano sugli zoccoli AN1 e AN2, usando un connettore a tre cavetti, per adattarsi alla diversa piedinatura.

Il modulo deve essere ruotato per connetterlo correttamente

AD1a. AD1a supporta una nutrita serie di sensori, che utilizzano sia una porta digitale che una analogica. Sono principalmente sette sensori ambientali di vario tipo, (di fiamma, temperatura digitale, suoni, ecc.). Sono montati tutti sulla stessa basetta e hanno un funzionamento analogo. Molto utile il fatto che abbiano un uscita analogica, con una gamma di valori da 0 a 1023 (2^{10}) e un uscita digitale, con il livello di soglia regolato da un trimmer, utile per attivare un allarme, oppure una periferica esterna, quale una caldaia, un condizionatore o un attuatore generico, controllato da un relay. Essi possono essere inseriti ***direttamente*** nello zoccolo AD1a, facendo attenzione al verso di inserimento.

Il funzionamento. Dalla porta analogica si ottengono segnali discreti, in una gamma da 0 a 1023 (2^{10}),mentre la porta digitale restituisce un'informazione

binaria, “0” oppure “1”, ed è utile per esempio per attivare un relay. Di default la porta analogica utilizzata è D2, ma la stessa porta è utilizzata anche dal display LCD 1602. Purtroppo questo display è molto affamato di porte digitali (da solo ne utilizza sei, da D2 a D7), per cui nella configurazione “base” di AD1 non è possibile usarlo per visualizzare le informazioni; infatti in molti programmi relativi a questo sensore si utilizzerà in alternativa il display TFT ST7735, che non usa la porta D2. Però si è pensato che il visore LCD è molto facile da configurare e pratico per semplici informazioni testuali. Per questo motivo si è deciso di inserire un *jumper* per poter scegliere se usare la porta D2 oppure la D15, permettendo in questo modo di usufruire del nostro amato display LCD 1602.

Si ricorda rapidamente che nativamente in Arduino le porte digitali terminano con D13; però indicandolo nel programma, le porte analogiche da A0 ad A5 (sei porte), possono diventare porte digitali, da D14 a D19. Naturalmente se si usa “D15” (corrispondente ad “A1”), non si potrà collegare contemporaneamente sulla *bs* anche un sensore sullo zoccolo analogico AN1, perché andrebbero in conflitto, in quanto usano *fisicamente* entrambi la porta analogica A1. La vita è piena di scelte difficili… Nel caso stessimo scrivendo un bellissimo programma, in cui è necessario tassativamente usare un “flame sensor” (KY-026), e contemporaneamente - per esempio – anche un “photo resistor” (KY-018) o un “water level sensor” (KY-039), non ci faremo prendere dal panico; modificando leggermente il programma, cambieremo la porta analogica utilizzata e potremo collegare il nostro photo resistor su AD2 (porta A7), oppure con un cavetto, causa la piedinatura leggermente diversa, su AN2 (porta A3). La nostra *bs* è molto versatile e ridondante; sta solo alla nostra fantasia ed ingegno plasmarla alle nostre necessità.

Ecco come modificare i programmi:

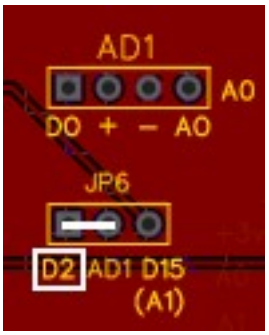
Programma originale: D2

```
int  sensorAnalogPin = A0;
int  sensorDigitalPin = 2;
int  analogValue = 0;
```

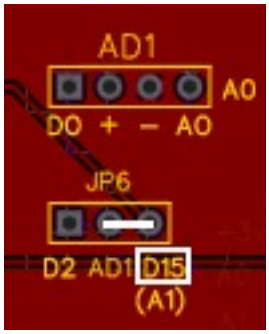
Programma modificato: (D)15

```
int  sensorAnalogPin = A0;
int  sensorDigitalPin = 15;
int  analogValue = 0;
```

Ed ecco come spostare il jumper sulla basetta per le due soluzioni:



Scelta D2 con il jumper



Scelta D15 con il jumper

Lista dei moduli (testati) che utilizzano lo zoccolo AD1a

Socket AD1a (digital/analogic)			1	2	3	4
Description	Code	Note	DO	+ (+5v)		AO
Big Sound	KY-037 - HW485		d2 (d15)	+	-	a0
Small Sound	KY-038 - HW496					
Linear Hall	KY-024 – SS49E – HW509					
Metal Touch	KY-036 - HW494					
Magnetic Spring	KY-025 - HW484					
Flame	KY-026 – HW491					
Digital Temperature	KY-028 – HW503					
Soil moisture sensor		Usare connettore – diversa piedinatura				

Sensori che alloggiano nativamente su AD1a

KY-37 – big sound

KY-38 – small sound

KY-024 – linear hall

KY-036 – metal touch

KY025 - magnetic spring

KY-026 –

flame sensor

KY-028 digital temperature

I programmi per i sensori ambientali
KY-024, KY-025, KY-026, KY-028, KY-036, KY-037, KY-038.

N.b.: I sensori **KY026, KY024, KY037, KY038, KY025, KY028, KY036** hanno tutti la stessa struttura fisica, cambiano solamente i sensori montati. Per cui i programmi sono identici, cambiano solo le scritte che identificano il tipo di controllo e i dati rilevati.
Ecco una lista delle loro funzioni:

- **KY-024** linear hall – **KY025** magnetic spring: questi due sensori usano dei principi leggermente diversi, ma sono entrambi sensibili ai campi magnetici.
- **KY-026** flame sensor: questo sensore è sensibile alla presenza di fiamme.
- **KY-028** digital temperature: come dice il nome stesso, registra con precisione la temperatura.
- **KY-036** metal touch: esso è sensibile al contatto con un oggetto metallico.
- **KY-037** big sound – **KY-038** small sound: agiscono nel campo audio. KY-037 ha un microfono amplificato e quindi è più sensibile di KY-038 che monta un semplice microfono.

Sono stati scritti programmi con diversi livelli di complessità, ma identici per tutti i sensori. Il programma ***sensor_1*** è totalmente generico, mentre ***sensor_2, sensor_2a*** e ***sensor_3*** possono avere scritte personalizzate sui display in base al sensore usato. E' sufficiente aprire i programmi e attivare le scritte più adatte a ogni scheda utilizzata.

Qui di seguito spieghiamo come fare.

```
  tft.println(" Flame control");          //usare questa dicitura se si usa il sensore KY026, flame control
//  tft.println("Sound control");           //usare questa dicitura se si usa i sensori KY037/8,big/small sound
//  tft.println("Linear Hall control");     //usare questa dicitura se si usa il sensore KY024, linear hall
//  tft.println("Metal touch control");     //usare questa dicitura se si usa il sensore KY036, metal touch control
//  tft.println("Magnetic control");        //usare questa dicitura se si usa il sensore KY025, magnetic spring control
//  tft.println("Temperature control");     //usare questa dicitura se si usa il sensore KY028, digital temperature control
```

Abbiamo preso come esempio il programma "sensor_2", ma questo è valido anche per i successivi. Come si vede dal rettangolo evidenziato, in questo caso nel programma apparirà la scritta "Flame control".
N.b.: in un programma, quando una linea è preceduta da "//", significa che ciò che segue è semplicemente un commento, e *non verrà eseguito dal programma*.

Se invece stessimo usando il modulo Metal touch, modificheremo il programma nel modo seguente:

```
//  tft.println(" Flame control");          //usare questa dicitura se si usa il sensore KY026, flame control
//  tft.println("Sound control");           //usare questa dicitura se si usa i sensori KY037/8,big/small sound
//  tft.println("Linear Hall control");     //usare questa dicitura se si usa il sensore KY024, linear hall
  tft.println("Metal touch control");     //usare questa dicitura se si usa il sensore KY036, metal touch control
//  tft.println("Magnetic control");        //usare questa dicitura se si usa il sensore KY025, magnetic spring control
//  tft.println("Temperature control");     //usare questa dicitura se si usa il sensore KY028, digital temperature control
```

N.b.: controllare tutto lo sketch. E' possibile che modifiche analoghe debbano essere fatte anche in altre parti del programma, per mantenere la coerenza dei messaggi che appariranno sui display.

***

Nome del programma:	**Sensor_1**
Porte:	**Modulo principale:**
A0 (analogico)	**KY-024, KY-025, KY-026, KY-028, KY-036, KY-037, KY-038.**
D2 (digitale) alternativa: D15	Vedi descrizioni nella pagina precedente.
Porte:	**Comp. Accessori:**
D11 (digitale)	Led LD1 - verde
D9 (digitale)	Led LD3 - rosso
D9, D10, D11 (digitale)	Led LD4 – tre colori
D9 (digitale)	Relay KY-019 su RL2
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale i valori analogici (sopra 500 non c'è attività) e quello digitale: 0 = nessuna attività; 1 = presente attività. Il led rosso è acceso quando c'è un'attività, in alternativa al verde, quando non è presente alcuna attività. Se è inserito il relay sulla porta RL2, si attiva in presenza dell'attività.
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts e almeno 500 mA
Link:	nessuno

Nome del programma:	**Sensor_2**
Porte:	**Modulo principale:**
A0 (analogico)	**KY-024, KY-025, KY-026, KY-028, KY-036,**
D2 (digitale)	Vedi descrizioni nella pagina precedente.
Porte:	**Comp. Accessori:**
D11 (digitale)	Led LD1 - verde
D9 (digitale)	Led LD3 - rosso
D9, D10, D11 (digitale)	Led LD4 – tre colori (opzionale)
D9 (digitale)	Relay KY-019 su RL2 (opzionale)
D13 (dig.) alternativa: D3; D9	Buzzer - KY-006 (KY-012) su BZ1
D3/./D7	Display TFT 1.77" 160(RGB)x128 su DY1
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale i valori analogici (sopra 500 non c'è attività) e quello digitale: 0 = nessuna presenza; 1 = presenza attività. Il led rosso è acceso quando si è rilevato un'attività, in alternativa al verde, quando essa non è presente. Se è presente il relay sulla porta RL2, si attiva in parallelo al led rosso. Le stesse informazioni sono presenti sul Display TFT. Il buzzer emette una serie di note quando è stata rilevata un'attività.
Librerie necessarie:	Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts e almeno 500 mA
Link:	nessuno

Nome del programma: Sensor_2a

Porte:
A0 (analogico)
D15 (digitale) *

Modulo principale:
KY-024, KY-025, KY-026, KY-028, KY-036, KY-037, KY-038. Vedi descrizioni nella pagina precedente.

Porte:
D11 (digitale)
D9 (digitale)
D9, D10, D11 (digitale)
D9 (digitale)
D13
D2/./D7

Comp. Accessori:
Led LD1 - verde
Led LD3 - rosso
Led LD4 – tre colori (opzionale)
Relay KY-019 su RL2 (opzionale)
Buzzer - KY-006 (KY-012) su BZ1
Display LCD 1602 (16 caratteri per 2 righe) su DY2

Monitor Seriale: sì (9600 baud)

Plotter seriale: no

Scopo del programma:
Mostra sul monitor seriale i valori analogici (sopra 500 non c'è attività) e quello digitale: 0 = nessuna presenza; 1 = presenza attività. Il led rosso è acceso quando si è rilevato un'attività, in alternativa al verde, quando essa non è presente. Se è presente il relay sulla porta RL2, si attiva in parallelo al led rosso. Le stesse informazioni sono presenti sul Display LCD. Il buzzer emette una serie di note quando è stata rilevata un'attività.

Librerie necessarie:
LiquidCrystal.h

Note:
Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts e almeno 500 mA
* Come da introduzione, questo programma usa la porta digitale (D)15. Spostare il jumper su D15

Link:
nessuno

Nome del programma: Sensor_3

Porte:
A0 (analogico)
D2 (digitale)

Modulo principale:
KY-024, KY-025, KY-026, KY-028, KY-036, KY-037, KY-038. Vedi descrizioni nella pagina precedente.

Porte:
D11 (digitale)
D9 (digitale)
D9, D10, D11 (digitale)
D9 (digitale)
D13 (digitale)
D3/./D7 (digitale)
A2 (analogico)

Comp. Accessori:
Led LD1 - verde
Led LD3 - rosso
Led LD4 – tre colori (opzionale)
Relay KY-019 su RL2 (opzionale)
Buzzer - KY-006 (KY-012) su BZ1
Display TFT 1.77" 160(RGB)x128 su DY1
Pulsante BT1

Monitor Seriale: sì (9600 baud)

Plotter seriale: no

Scopo del programma:
Mostra sul monitor seriale i valori analogici (sopra 500 non c'è attività) e quello digitale: 0 = nessuna presenza; 1 = presenza attività. Il led rosso è acceso quando si è rilevato un'attività, in alternativa al verde, quando essa non è presente. Se è presente il relay sulla porta RL2, si attiva in parallelo al led rosso. Le stesse informazioni sono presenti sul Display TFT. Il buzzer emette una serie di note quando è stata rilevata un'attività. A differenza del programma precedente, anche se la termina l'attività, continua ad

	esserci la segnalazione sonora, visiva, l'attivazione del relay, fino a quando non si effettua il reset del sistema, premendo per qualche secondo il pulsante presente sulla scheda denominato BT1.
Librerie necessarie:	Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts e almeno 500 mA
Link:	nessuno

Soil moisture sensor

Questo sensore misura l'umidità del suolo, per cui può essere utilissimo nell'organizzare un piccolo giardino e/o orto con un irrigazione automatica "intelligente".

Per collegarlo allo zoccolo AD1/a, è necessario usare un connettore a quattro cavetti. E' più semplice collegarlo sullo zoccolo AD1/b, ruotando semplicemente la scheda di 180° gradi rispetto ai sensori del gas.

Soil moisture sensor

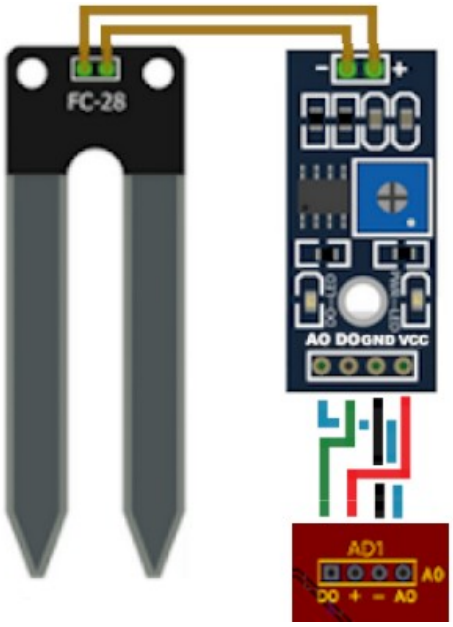
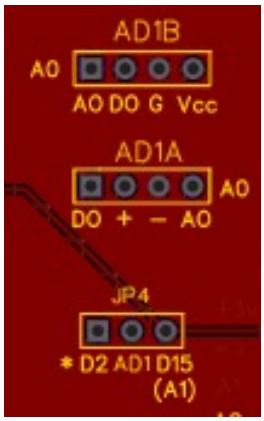


Immagine del collegamento del sensore dell'umidità del terreno (soil moisture sensor) allo zoccolo AD1a.

Per i programmi relativi, vedi lo zoccolo AD1b.

Lo zoccolo AD1b



AD1b si trova nella parte alta a sinistra della bs, vicino alle porte analogiche di Arduino, subito sopra lo zoccolo AD1a.
La porta analogica usata è A0.

Subito sotto di esso si trova il jumper per selezionare quale porta digitale da usare: D2 (default) oppure D15 (A1).

Questa scelta è stata fatta per evitare il conflitto di D2 con i display DY2. Selezionando invece la porta D15, essa va in conflitto con lo zoccolo AN1, che usa la porta analogica A1.
AD1a e AD1b usano le stesse porte, per cui non si possono usare contemporaneamente.

Per lo zoccolo AD1b, valgono tutte le stesse considerazione che per AD1a.
Si ricorda il jumper JP4, che permette di scegliere tra le porte digitali D2 e D15.

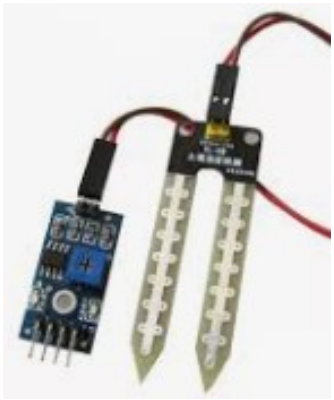
Lista dei moduli che utilizzano lo zoccolo AD1b

Socket AD1b (digital/analogic)			1	2	3	4
Description	Code	Note	AO	DO	GND	vcc (+5v)
Gas sensor	MQ2 - KY030		a0	d2 (d15)	-	+
Gas sensor	MQ3					
Gas sensor	MQ4					
Gas sensor	MQ5					
Gas sensor	MQ6					
Gas sensor	MQ7					
Gas sensor	MQ8					
Gas sensor	MQ9					
Gas sensor	MQ135					
Soil moisture sensor		ruotare a 180° rispetto ai sensori di gas.				

Sensori che alloggiano nativamente su AD1b

MQ2/./MQ135Gas sensor

MQ7 Gas sensor



Soil moisture sensor

Nota: tutti i moduli che montano sullo zoccolo AD1a, possono essere collegati allo zoccolo AD1b, utilizzando un connettore a quattro cavetti per adattarsi alla piedinatura diversa.

Su questo zoccolo, avendo a disposizione una porta analogica, possono essere collegati anche tutti i moduli che alloggiano sugli zoccoli AN1 e AN2, usando un connettore a tre cavetti, per

adattarsi alla diversa piedinatura.

Programmi per i rivelatori di gas: da MQ2 a MQ135 (AD1b)

I programmi che seguiranno sono relativi ai sensori di gas, essi hanno delle specifiche diverse, ma utilizzano tutti la stessa scheda e hanno quindi un comportamento analogo.
Ecco una descrizione sintetica:

- MQ-2: GPL, propano, butano, metano, alcol;
- MQ-3: Alcol
- MQ-4: Metano (alta sensibilità); alcol, fumo (bassa sens.)
- MQ-5: GPL, metano, gas di città; alcol, fumo (bassa sens.)
- MQ-6: GPL, iso-butano, propano; alcol, fumo (bassa sens.)
- MQ-7: Monossido di carbonio
- MQ-8: Idrogeno (H_2); alcol, GPL, fumi di cucina (bassa sens.)
- MQ-9: monossido di carbonio
- MQ-135: gas di ammoniaca, toluene, idrogeno, anidride solfidrica, fumo

Nota: i programmi, variando solo alcune descrizioni, sono valide per tutti i sensori di gas.

Nome del programma:	**Gas_1**
Porte:	**Modulo principale:**
A0 (analogico)	**MQ-2**, **MQ-3**, **MQ-4**, **MQ-5**, **MQ-6**, **MQ-7**, **MQ-8**, **MQ-9**, **MQ-135.**
D2 (digtale)	
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Prima che il sensore sia attivo, sono necessari circa 20 secondi, perché esso si deve riscaldare. Quando è attivo, se si superano le 300 parti si un milione di gas, sul monitor seriale appare la scritta "Smoke detected!", e il valore digitale passa da 1 a 0.
Note:	Il sensore, specialmente nella fase di riscaldamento, richiede molta corrente. E' vivamente consigliabile alimentare Arduino con un alimentatore appropriato, che fornisca almeno 1000 mA. Il jumper di selezione della porta digitale deve essere settato su "D2"
Link:	https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

Nome del programma:	**Gas_2**
Porte:	**Modulo principale:**
A0 (analogico)	**MQ-2**, **MQ-3**, **MQ-4**, **MQ-5**, **MQ-6**, **MQ-7**, **MQ-8**, **MQ-9**, **MQ-135.**
D2 (digtale)	
Porte:	**Comp. Accessori:**
D11 (digitale)	Led LD1 - verde
D13	Buzzer - KY-006 (KY-012) su BZ1
Monitor Seriale: sì	**Plotter seriale: no**

(9600 baud)

Scopo del programma: Prima che il sensore sia attivo, sono necessari circa 20 secondi, perché esso si deve riscaldare. Quando è attivo, se si superano le 300 parti si un milione di gas, sul monitor seriale appare la scritta "Smoke detected!", e il valore digitale passa da 1 a 0. Quando si supera la soglia, il led lampeggia e il buzzer emette una sequenza di suoni. Appena l'atmosfera torna normale, gli allarmi si spengono.

Note: Il sensore, specialmente nella fase di riscaldamento, richiede molta corrente. E' vivamente consigliabile alimentare Arduino con un alimentatore appropriato, che fornisca almeno 1000 mA.
Il jumper di selezione della porta digitale deve essere settato su "D2"
Il jumper del buzzer deve essere settato su "D13".

Link: nessuno

Nome del programma: **Gas_3**

Porte:	**Modulo principale:**
A0 (analogico)	**MQ-2**, **MQ-3**, **MQ-4**, **MQ-5**, **MQ-6**, **MQ-7**, **MQ-8**, **MQ-9**, **MQ-135.**
D2 (digtale)	
Porte:	**Comp. Accessori:**
D11 (digitale)	Led LD1 - verde
D9 (digitale)	Relay KY-019 su RL2
D10(digitale)	Led LD2 - rosso
D9 (digitale)	Led LD3 - rosso
D13	Buzzer - KY-006 (KY-012) su BZ1

Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)

Scopo del programma: Prima che il sensore sia attivo, sono necessari circa 20 secondi, perché esso si deve riscaldare. Quando è attivo, se si superano le 300 parti si un milione di gas, sul monitor seriale appare la scritta "Smoke detected!", e il valore digitale passa da 1 a 0. Quando si supera la soglia, il buzzer emette una sequenza di suoni. Questo programma è un piccolo perfezionamento di quello precedente. Quando si supera la soglia, i due led rossi lampeggiano. Il led verde, normalmente acceso, si spegne quando scattano gli allarmi. Il serie al led rosso (LD3) si può collegare un relay, per attivare un allarme remoto. Appena l'atmosfera torna normale, gli allarmi si spengono.

Note: Il sensore, specialmente nella fase di riscaldamento, richiede molta corrente. E' vivamente consigliabile alimentare Arduino con un alimentatore appropriato, che fornisca almeno 1000 mA.
Il jumper di selezione della porta digitale deve essere settato su "D2"
Il jumper del buzzer deve essere settato su "D13".

Link: nessuno

Nome del programma:	**Gas_4**
Porte:	**Modulo principale:**
A0 (analogico)	**MQ-2**, **MQ-3**, **MQ-4**, **MQ-5**, **MQ-6**, **MQ-7**, **MQ-8**, **MQ-9**, **MQ-135.**
D2 (digtale)	
Porte:	**Comp. Accessori:**
D2/./D7	Dispaly LED 1602 su DY2
D11 (digitale)	Led LD1 - verde
D9 (digitale)	Relay KY-019 su RL2
D10(digitale)	Led LD2 - rosso
D9 (digitale)	Led LD3 - rosso
D13	Buzzer - KY-006 (KY-012) su BZ1
A2	Pulsante sulla basetta (BT1)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Prima che il sensore sia attivo, sono necessari circa 20 secondi, perché esso si deve riscaldare. Quando è attivo, se si superano le 300 parti si un milione di gas, sul monitor seriale appare la scritta "Smoke detected!", e il valore digitale passa da 1 a 0. Quando si supera la soglia, il buzzer emette una sequenza di suoni. Questo programma è un piccolo perfezionamento di quello precedente. Quando si supera la soglia, i due led rossi lampeggiano. Il led verde, normalmente acceso, si spegne quando scattano gli allarmi. Il serie al led rosso (LD3) si può collegare un relay, per attivare un allarme remoto. Al contrario del programma precedenti, anche se la'atmosfera torna normale, gli allarmi restano attivi. Al programma precedente è stato aggiunto il pulsante BT1, che quando premuto resetta gli allarmi. Sul Dispaly appaiono i valori riscontrati e i messaggi di allarme.
Note:	Il sensore, specialmente nella fase di riscaldamento, richiede molta corrente. E' vivamente consigliabile alimentare Arduino con un alimentatore appropriato, che fornisca almeno 1000 mA. Il jumper di selezione della porta digitale deve essere settato su "D2" Il jumper del buzzer deve essere settato su "D13".
Link:	nessuno

P.s.: variare le scritte in base al gas rilevato.

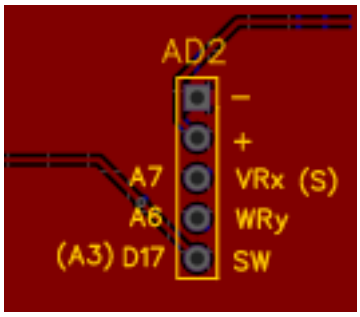
Programmi relativi al soil moisture sensor (AD1b)

Nome del programma: **Moisture_1**
Porte: **Modulo principale:**
A0 (analogico) **Moisture sensor**
D2 (digtale)
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Questo programma didattico mostra sul monitor seriale il valore compreso tra 0 e 1023 dell'umidità del suolo; se il valore è inferiore a 300, il suolo è secco; se è superiore a 700, esso è troppo umido
Note: nessuna
Link: https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

Nome del programma: **Moisture_2**
Porte: **Modulo principale:**
A0 (analogico) **Moisture sensor**
D2 (digitale)
Porte: **Comp. Accessori:**
A4 SDA A5 SCL (analogico) Dispay OLED SSD1306 su DY3
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Questo programma didattico mostra sul monitor seriale il valore compreso tra 0 e 1023 dell'umidità del suolo; se il valore è inferiore a 300, il suolo è secco; se è superiore a 700, esso è troppo umido. Le informazioni possono essere visualizzate sul display OLED SSD1306.
Note: nessuna
Link: nessuno

Nome del programma: **Moisture_2a**
Porte: **Modulo principale:**
A0 (analogico) **Moisture sensor**
D2 (digitale)
Porte: **Comp. Accessori:**
D2/./D7 Dispay LCD 1602 (16 caratteri x 2 linee)
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Questo programma didattico mostra sul monitor seriale il valore compreso tra 0 e 1023 dell'umidità del suolo; se il valore è inferiore a 300, il suolo è secco; se è superiore a 700, esso è troppo umido. Le informazioni possono essere visualizzate sul display LCD1602
Note: nessuna
Link: https://arduinopoint.com/soil-moisture-sensor-arduino-project/

Lo zoccolo AD2



Lo zoccolo AD2 si trova a sinistra della bs, a metà altezza sotto e a sinistra di Arduino, tra AN4 e DY2.

Questo connettore è stato studiato appositamente per il joystick, che può essere inserito direttamente. Ma può ospitare direttamente, usando i primi tre piedini in alto, anche i vari moduli che si collegano a AN1 (rispettando le polarità "-", "+",e "S") e su AD1, collegando questi ultimi con in cavetto perché i piedini hanno una diversa disposizione. Naturalmente è necessario variare anche il programma: per i moduli di AN1 la porta sarà A7 invece che A1; per AD1 la porta analogica sarà anche in questo caso A7, mentre quella digitale diventerà D17, invece che D2 o D15.
Questo zoccolo va in conflitto con "Audio in" (A6) e con AN2 (se si usa la porta analogica A3).

AD2. E' uno zoccolo per certi versi più complesso di AD1. Esso è stato pensato principalmente per poter accogliere direttamente il joystick KY-023, che usa due porte analogiche (A6 e A7) per gli assi x e y del joystick, e una porta digitale per il pulsante (D17). Se si osserva lo schema, questo zoccolo è connesso *fisicamente* a tre porte analogiche: A6, A7 e A3; ma utilizzando il "trucco" illustrato per AD1, la porta A3 viene trasformata nel programma in D17. **Vedi come fare**.
Con qualche accorgimento, su questo zoccolo si possono inserire moduli e sensori che si collegano nativamente su AN1, AN2, AD1a e AD1b.

Ecco la lista sintetica dei moduli che potremo inserire su AD2:

Socket AD2 (digital/analogic)			1	2	3	4	5
Description	Code	Note	GND	VCC (+5v)	VRX	VRY	SW
Joystick	KY-023 – HW504		-	+	a7	a6	a3 (d17)
			-	+	S	/	/
Sensori su AN1			-	+	a7	/	/
			GND	Vcc	AO	/	/
Sensori su AN2			-	+	a7	/	/
			GND	Vcc	AO	/	DO
Sensori su AD1a e AD1b			-	+	a7	/	a3 (d17)

Il joystick KJ-023 è il modulo che monta nativamente su questo zoccolo.

Però potremmo collegare su di esso anche i moduli per **AN1**, per **AN2** e anche quelli per **AD1a** e **AD1b**.

KY-023 -joystick

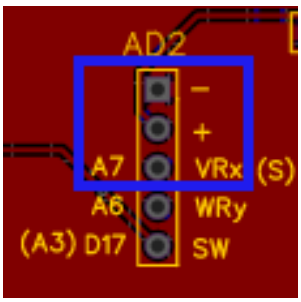
Questo connettore è stato studiato principalmente per il joystick, che può essere alloggiato nativamente, utilizzando due porte analogiche (assi X e Y) e uno digitale per il pulsante.

Programmi per il Joystick (AD2)

Nome del programma:	**Joy_1**
Porte:	**Modulo principale:**
D17 (digitale)*	Joystick - KY023
A6 (analogico)	
A7 (analogico)	
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Muovendo il Joystick, si vede sul monitor seriale lo spostamento sugli assy X e Y. Premendo il joystick si cambia lo stato della porta digitale.
Note: *	Si utilizza la porta analogica A3 come ingresso digitale. In questo specifico caso, non si potrà usare questo programma e contemporaneamente un modulo, come un potenziometro, inserito sul connettore analogico AN2, perché la sua porta è la A3 e "Audio In", perché andrebbe in conflitto con la porta A6.
Link:	nessuno

Nome del programma:	**Joy_2**
Porte:	**Modulo principale:**
D2 (digitale)	Joystick - KY023
A6 (analogico)	
A7 (analogico)	
Porte:	**Comp. Accessori:**
D9 (digitale)	Led3 – rosso su LD3
D10 (digitale)	Led2 – blu su LD2
D13 (digitale)	Buzzer KY-006 (KY-012) su BZ1
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Muovendo il Joystick, si vede sul monitor seriale lo spostamento sugli assy X e Y. Premendo il joystick si cambia lo stato della porta digitale. Incrementando l'asse X, il led rosso aumenta di luminosità; diminuendo il valore, la luminosità del led si affievolisce. Idem per il, led blu per l'asse Y. Quando si preme il joystick, il buzzer emette una nota.
Note:	Si utilizza la porta analogica A3 come ingresso digitale. In questo specifico caso, non si potrà usare questo programma e contemporaneamente un modulo, come un potenziometro, inserito sul connettore analogico AN2, perché la sua porta è la A3 e "Audio In", perché andrebbe in conflitto con la porta A6.
Link:	nessuno

Moduli nativi per AN1 che possono essere connessi su AD2



Il joystick usa tutte le porte dello zoccolo AD2, in quanto è stato progettato proprio per questo uso. Tuttavia si possono collocare su questo zoccolo tutti i moduli che montano sullo zoccolo AN1, naturalmente modificando il programma corrispondente, usando la porta analogica A7 invece che la porta A1, ponendo attenzione alla posizione e al senso di inserimento dei moduli stessi. (Vedi immagine). I moduli analogici a tre pin possono essere inseriti nella parte superiore del connettore, facendo attenzioni alle polarità. Il pin A7/Vrx corrisponde a S/Y dello zoccolo DG1. Questo permette di ampliare la gamma di possibilità di applicazioni della *bs*.

Ecco la lista dei moduli nativi per AN1 che potremo inserire su AD2:

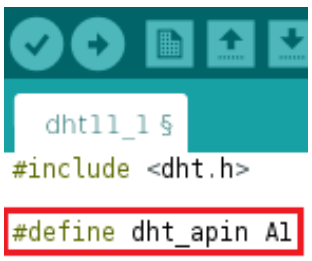
Socket AD2 (digital/analogic)			1	2	3	4	5
Description	Code	Note	GND	VCC (+5v)	VRX	VRY	SW
			-	+	S	/	/
Photo resistor	KY-018 – HW486		-	+	a7	/	/
Heart bit sensor	KY-039 - HW502						
Temperature Humidity	KY-015 – DHT11 – HW507						
Tap Module	KY-031 – HW500						
Shock sensor	KY-002 – HW513						
Tilt switch	KY-020 – HW501						
Button	KY-004 – HW483						
Photo interrupt	KY-010 – HW487	piedinatura ruotata					
Reed switch sensor	KY-021 – HW-497	piedinatura ruotata					
Mercury tilt module	KY-017 -HW505	piedinatura ruotata					
Hall magnetic sensor	KY-035 – HW492	piedinatura ruotata					
Analog temperature	KY-013 – HW498	piedinatura ruotata					
Water level sensor	Sen18	piedinatura ruotata					

Clicca qui per vedere le immagini dei moduli e per i programmi relativi ad AN1.

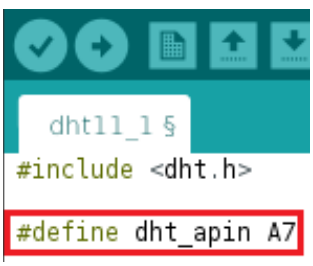
Si possono usare gli stessi programmi indicati per i moduli presenti su AN1, ***avendo cura di cambiare nei listati la porta A1 con la porta A7.*** Ecco come fare:

Come modificare i programmi pensati per lo zoccolo AN1 e adattarli a AD2:

Programma per AN1:



Programma modificato per AD2:



Si può creare una semplice tabella di conversione, per inserire i moduli che alloggiano su AN1 e adattarli su AN2, senza l'uso di cavetti aggiuntivi:

AN1	**AD2**
S	A7
+	+5V
-	-

Moduli nativi per AN2 che possono essere connessi su AD2

Lista dei moduli nativi per AN2 che potremo inserire su AD2:

Socket AD2 (digital/analogic)			1	2	3	4	5
Description	Code	Note	GND	VCC (+5v)	VRX	VRY	SW
			-	+	S	/	/
Linear Potentiometer	Linear – 10 KΩ	Usare connettore piedinatura diversa			a7	/	/

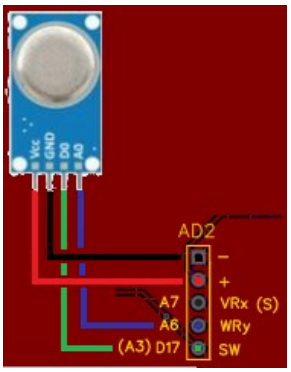
I potenziometri permettono di variare con continuità la resistenza.
Abitualmente a sinistra si collega la massa (-); a destre la tensione (Vcc, +5v); il piedino centrale verrà collegato alla porta analogica del segnale.
Se collegato ad An2, sarà A3 0 A5; se verrà collegato su AD2, ***bisognerà variare la porta in A7.***

Moduli nativi per AD1a e AD1b che possono essere connessi su AD2

Lista dei moduli nativi per AD1a/AD1b che potremo inserire su AD2

Socket AD2 (digital/analogic)			1	2	3	4	5
Description	Code	Note	GND	VCC (+5v)	VRX	VRY	SW
			-	+	S	/	/
			GND	Vcc	AO	/	DO
Gas sensor	MQ2 - KY030	Usare connettore – diversa piedinatura					
Gas sensor	MQ3	Usare connettore – diversa piedinatura					
Gas sensor	MQ4	Usare connettore – diversa piedinatura					
Gas sensor	MQ5	Usare connettore – diversa piedinatura					
Gas sensor	MQ6	Usare connettore – diversa piedinatura					
Gas sensor	MQ7	Usare connettore – diversa piedinatura					
Gas sensor	MQ8	Usare connettore – diversa piedinatura					
Gas sensor	MQ9	Usare connettore – diversa piedinatura					
Gas sensor	MQ135	Usare connettore – diversa piedinatura	-	+	a7	/	a3 (d17)
Big Sound	KY-037 - HW485	Usare connettore – diversa piedinatura					
Small Sound	KY-038 - HW496	Usare connettore – diversa piedinatura					
Linear Hall	KY-024 – SS49E – HW509	Usare connettore – diversa piedinatura					
Metal Touch	KY-036 - HW494	Usare connettore – diversa piedinatura					
Magnetic Spring	KY-025 - HW484	Usare connettore – diversa piedinatura					
Flame	KY-026 – HW491	Usare connettore – diversa piedinatura					
Digital Temperature	KY-028 – HW503	Usare connettore – diversa piedinatura					
Soil moisture sensor		ruotare a 180° rispetto ai sensori di gas.					

Naturalmente nulla vieta di usare questo zoccolo per connettere dei moduli che usino nativamente ***tre porte analogiche***, nel caso ne avessimo qualcuno. Ma anche i sensori che alloggiano su AD1 possono essere collegati a questo; semplicemente avendo una diversa disposizione di porte, richiedono di usare un connettore a quattro cavi per connettersi adeguatamente. Vedi esempio nell'immagine qui di fianco. In questo caso, il sensore MQ2 è collegato con un cavetto allo zoccolo AD2. La porta analogica "A0" è collegata ad A6 (ma potrebbe essere collegata anche su A7, modificando adeguatamente il programma; mentre quella digitale "D0" è collegata a D17. E' sufficiente effettuare le giuste variazioni nel programma affinché tutto funzioni correttamente. In questo modo, per esempio, si può collegare il modulo "flame" su AD1 e Il sensore di fumo MQ2 su Ad2 e in un solo programma si possono ottenere allarmi sia per il fumo che per una fiamma. Qui di seguito un esempio di trasposizione del programma per MQ2 da AD1 ad AD2



Collegamento su AD2

Nota: quando si usa la porta D17 (che fisicamente corrisponde ad A3), non si potrà connettere contemporaneamente un potenziometro sullo zoccolo AN2, perché andrebbe in conflitto.

Tabella generale di conversione:

DO	→ A3-D17
AO	→ A6
non utilizzato	A7
VCC	→ +
GND	→ -

Come indicato, tutti i moduli che si possono inserire su AD1a/AD1b, possono essere utilizzati anche su AD2, e ciò rende questo zoccolo molto versatile. E' sufficiente collegare i moduli con un connettore a quattro fili seguendo la giusta sequenza e modificare il programma come indicato, ovvero:

Programma originale, per AD1:

```
MQ2_1 §
#define MQ2pin (0)          //porta analogica per AO
int sensorDigitalPin = 2;   //porta digitale  per DO
int  digitalValue;
float sensorValue;  //variable to store sensor value
```

In cui: ***#define MQ2pin (0)***, indica che usa la porta analogica A0
e ***int sensorDigitalPin = 2;*** indica che usa la porta digitale D2

Programma modificato, per utilizzarlo con AD2:

```
MQ2_1a §
#define MQ2pin (6)          //porta analogica per AO
int sensorDigitalPin = 17;  //porta digitale  per DO
int  digitalValue;
float sensorValue;  //variable to store sensor value
```

In cui: ***#define MQ2pin (6)***, indica che usa la porta analogica A6
e ***int sensorDigitalPin = 17;*** indica che usa la porta digitale D17

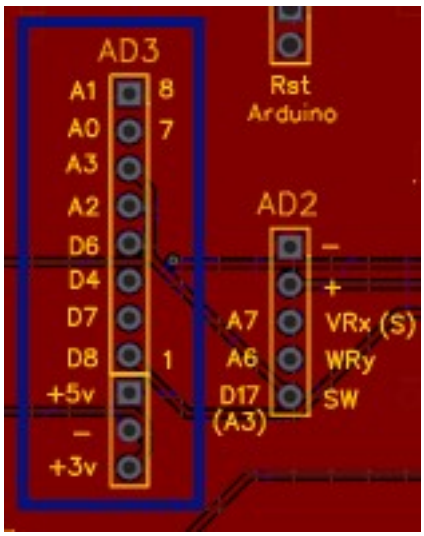
… e voilà, il gioco è fatto! Anche in questo caso, è sufficiente usare la fantasia e la logica per poter collegare più sensori dello stesso tipo su zoccoli diversi per realizzare progetti complessi.

Esempio: programma per rilevatore di gas

A titolo di esempio, è stato inserito anche il primo programma per il sensore di gas, a cui opportunamente sono state cambiate le porte di collegamento. Sarà necessario collegarlo con un connettore a 4 cavi, perché le connessioni stra sensore e zoccolo non corrispondono.

Nome del programma:	Gas_1a
Porte: A0 (analogico) D2 (digtale) **Monitor Seriale: sì** (9600 baud)	**Modulo principale:** **MQ-2**, **MQ-3**, **MQ-4**, **MQ-5**, **MQ-6**, **MQ-7**, **MQ-8**, **MQ-9**, **MQ-135.** **Plotter seriale: no**
Scopo del programma:	Prima che il sensore sia attivo, sono necessari circa 20 secondi, perché il sensore si deve riscaldare. Quando è attivo, se si superano le 300 parti si un milione di gas, sul monitor seriale appare la scritta "Smoke detected!", e il valore digitale passa da 1 a 0. La prta analogica usata è A6; quella digitale D17 (A3). Quindi A7 resta inutilizzata.
Note:	Il sensore, specialmente nella fase di riscaldamento, richiede molta corrente. E' vivamente consigliabile alimentare Arduino con un alimentatore appropriato, che fornisca almeno 1000 mA.
Link:	https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

Lo zoccolo AD3



Lo zoccolo AD3 è stato introdotto nella nuova versione della *bs*, ovvero la 0.6.1.
Esso è specifico per le keypad, sia la 4x3 (connessioni da 1 a 7) che per la 4x4 (connessioni da 1 a 8).

E' posizionata a sinistra della *bs*, più o meno al centro, di lato ad AD2.

Utilizza un mix di porte analogiche (da A0 ad A3) e digitali (D4, D6, D7, D8)

Le keypad non richiedono alimentazione, ma sono stati aggiunti tre piedini per l'alimentazione (+5v, Gnd, +3,3v), per altri moduli.

AD3, come si può vedere nello specchietto di lato all'immagine dello zoccolo, è stato aggiunto solo con questa nuova versione della *bs*. Esso è stato introdotto specificamente per accogliere nativamente sia le keyboard da sedici tasti (4x4), che usano i connettori da 1 a 8, che quelle da dodici tasti (4x3), che utilizzano i connettori da 1 a 7.

Si è sentito questa esigenza perché le keyboard sono piuttosto avide di connessioni (7 o 8 porte, a seconda della versione). Naturalmente possono essere usate tutte porte digitali, ma in questo modo ne restano ben poche per collegare eventuali altri moduli. Si è quindi scelto un compromesso, utilizzando quattro porte digitali e quattro analogiche.

Come si può notare, sono rimaste libere le porte A4 e A5, per cui ad esse si possono collegare tutti i moduli I2C, come il clock RTC, i display Oled, il display LCD 1602 con modulo I2C, ecc.

Utilizzando questa configurazione, è stato ideato un programma con una serratura a doppia combinazione, inserendo una password da tastiera, e poi avvicinando una scheda Rfid al lettore RF522. **Vedi programma**.

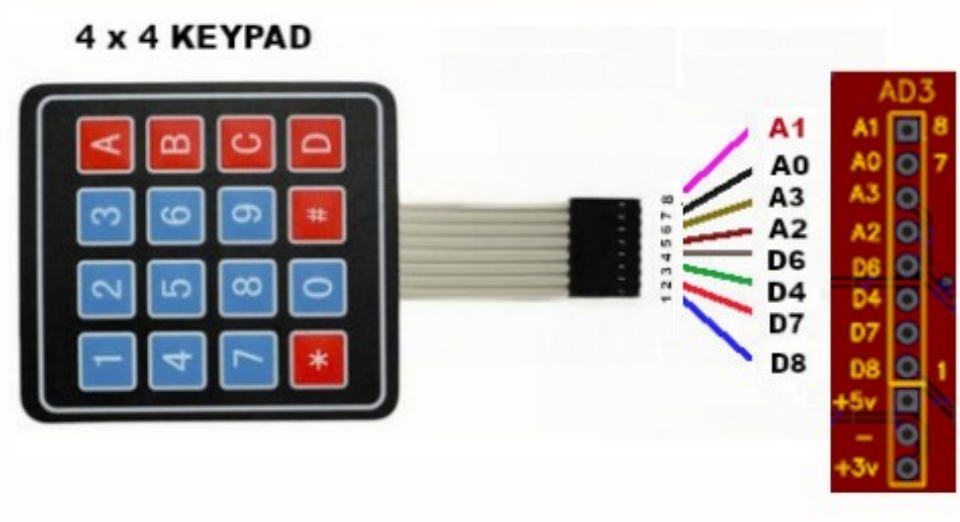
Naturalmente potremo usare questo zoccolo anche per collegare moduli che non sono stati previsti in questo paragrafo.

Lista sintetica dei moduli che potremo inserire su AD3:

Socket AD3 (digital/analogic)			1	2	3	4	5	6	7	8	9	10	11
Description	Code	Note	1	2	3	4	5	6	7	8	+5v	Gnd	+3,3v
Keypad module 4x3 *			D8	D7	D4	D6	A2	A3	A0	/	/	/	/
Keypad module 4x4 *			D8	D7	D4	D6	A2	A3	A0	A1	/	/	/

Immagini delle due tastiere.

Lista dei programmi per le keypad



Ecco come connettere le tastiere alla porta AD3. La keypad 4x3 non usa il piedino 8 (connesso ad A1)

Nome del programma:	**Key_16_1**
Porte:	**Modulo principale:**
D7, D8, D11, D12 (digitali) A0, A1, A2, A3 (analogiche)	**Tastiera a 16 cifre** (8 connettori)
Monitor Seriale: sì	**Plotter seriale: no**
Scopo del programma:	Questo programma dimostrativo mostra sul monitor seriale il valore dei tasti premuti, che va da "0" a "9", più "*", "#", e "A", "B", "C", "D".
Note:	Attenzione a rispettare la corretta sequenza delle connessioni su Arduino
Link:	nessuno

Nome del programma:	**Key_16_2**
Porte:	**Modulo principale:**
D7, D8, D11, D12 (digitali) A0, A1, A2, A3 (analogiche)	**Tastiera a 16 cifre** (8 connettori)
Porte:	**Comp. Accessori:**
A4 (SDA), A5 (SCL) anal.	Display LCD1602 + HW61 (adattatore per LCD 1602 al protocollo I2C)
Monitor Seriale: sì	**Plotter seriale: no**
Scopo del programma:	Questo programma dimostrativo mostra sul monitor seriale e sul display LCD 1602 con ***protocollo I2C*** il valore dei tasti premuti, che va da "0" a "9", più "*", "#", e "A", "B", "C", "D".
Librerie richieste:	Keypad.h, LiquidCrystal_I2C.h
Note:	Attenzione a rispettare la corretta sequenza delle connessioni su Arduino
Link:	https://projecthub.arduino.cc/mckean0/2d99999f-d375-4b8b-89b5-19b01005444d

Nome del programma:	**Key_16_3**
Porte:	**Modulo principale:**
D7, D8, D11, D12 (digitali) A0, A1, A2, A3 (analogiche)	**Tastiera a 16 cifre** (8 connettori)
Porte:	**Comp. Accessori:**
A4 (SDA), A5 (SCL) anal.	Display LCD1602 + HW61 (adattatore per LCD 1602 al protocollo I2C)
D9 (digitale)	Led rosso su LD3
D9 (digitale)	Relay su RL2 (contestualmente al led su LD3)
D10	Led verdi su LD1
D13	Buzzer KY-06 (o KY-012) su BZ1
Monitor Seriale: sì	**Plotter seriale: no**
Scopo del programma:	Questo programma dimostrativo permette di azionare un servomotore che può sbloccare una porta/un cassetto e attivare contemporaneamente e un relay che può azionare un qualsiasi meccanismo, trasformando di fatto Arduino in una chiave a combinazione. Sul monitor seriale e sul display LCD 1602 con protocollo I2C

appare il valore dei tasti premuti, che va da “0” a “9”, più “*”, “#”, e “A”, “B”, “C”, “D”. Se si eccede il numero di lettere/numeri che compongono la combinazione, resta acceso un led rosso e il buzzer emette un suono a bassa frequenza, il servomotore resta su posizione “chiuso”; se invece si inserisce la combinazione corretta, si accende un led verde, il servomotore si sposta in posizione “aperto”, si attiva il relaiy e il buzzer emette una nota più acuta. Premendo i tasti “*” o “#”, si richiude la “serratura” elettronica.

Librerie richieste: Keypad.h, LiquidCrystal_I2C.h

Note: Attenzione a rispettare la corretta sequenza delle connessioni su Arduino

Link: https://projecthub.arduino.cc/mckean0/2d99999f-d375-4b8b-89b5-19b01005444d

Come impostare la password e la sua lunghezza:

```
char* password = "427D";
int lenpsw = 4;
int keylen = 0;
int position = 0;
```

“char* password” salva la password, che attualmente è uguale a ***“427D”***, ma che può essere lunga a piacere (si consiglia di non uperare i 16 caratteri, altrimenti non appare tutta sul display). Si ossono usare i numeri da 0 a 9 e le lettere A, B, C, D.
“lenpsw” salva la lunghezza della password. Nel caso se ne usi una più lunga o più corta, è necessario modificare anche questo parametro.

Disposizione dei piedini per la keypad da 12 tasti.

Porre molta attenzione al corretto collegamento dei piedini, pena malfunzionamenti e frustrazioni!

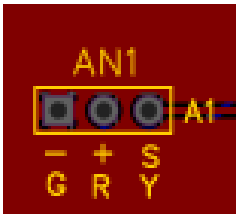
Nome del programma:	**Key_12**
Porte:	**Modulo principale:**
D7, D8, D11, D12 (digitali) A1, A2, A3 (analogiche)	**Tastiera a 12 cifre** (7 connettori)
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	Questo programma dimostrativo mostra sul monitor seriale il valore dei tasti premuti, che va da “0” a “9”, più “*” e “#”.
Note:	Attenzione a rispettare la corretta sequenza delle connessioni su Arduino
Link:	nessuno

Come si vede c’è un solo programma dimostrativo per la keypad a 12 tasti, mentre ce ne sono diversi per quella a 16 tasti. Con poche modifiche i programmi per quest’ultima possono essere adattati anche per quella con un minor numero di tasti.

Gli zoccoli per moduli che usano porte analogiche.

Sulla parte sinistra della *bs* sono stati inseriti 5 zoccoli che utilizzano le porte analogiche di Arduino Nano. Ora verranno analizzati singolarmente.

Lo zoccolo AN1



Lo zoccolo AN1 si trova sulla parte alta a sinistra della *bs* e si connette alla porta analogica A1.

Va in conflitto con la porta AD1, se il jumper è spostato su D15, che corrisponde fisicamente alla porta A1.

AN1. Questo zoccolo ha tre connettori: massa (-), alimentazione +5v (+) e segnale (S), collegato alla porta analogica A1. Su di esso si possono collegare alcuni sensori, che appariranno nella tabella apposita. Il suo uso è piuttosto semplice, non ci sono segnalazioni particolari.

Moduli che si collegano su AN1

Socket AN1 (analogic)			1	2	3
Description	Code	Note	-	vcc (+5v)	S
Photo resistor	KY-018 - HW486				
Heart bit sensor	KY-039 – HW502				
Temperature Humidity	KY-015 – DHT11 - HW507				
Tap Module	KY-031 – HW500				
Shock sensor	KY-002 – HW513				
Tilt switch	KY-020 – HW501				
Button	KY-004 – HW483				
Water level sensor	Sen18	piedinatura ruotata	-	+	a1
Photo interrupt	KY-010 – HW487	piedinatura ruotata			
Reed switch sensor	KY-021 – HW-497	piedinatura ruotata			
Mercury tilt module	KY-017 -HW505	piedinatura ruotata			
Hall magnetic sensor	KY-035 – HW492	piedinatura ruotata			
Analog temperature	KY-013 – HW498	piedinatura ruotata			
Linear Potentiometer	10 KΩ	Usare connettore piedinatura diversa			

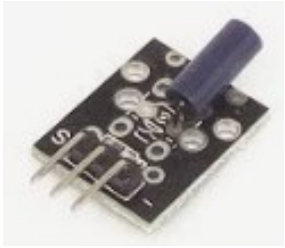
Immagini dei moduli che si connettono su AN1

KY-018 Photo interrupt

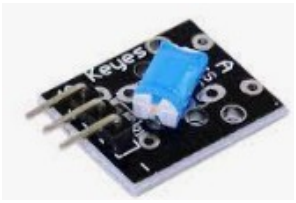
KY-039 heart bit sensor

KY-015 –DHT11 temp/humid. sensor

KY-031 tap module



KY-002 shock sensor



KY-020 tilt switch

KY-004 button

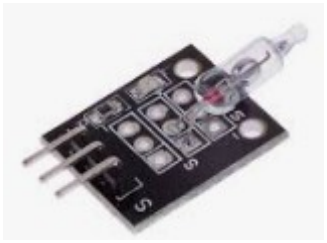
Questa prima serie di moduli, visti frontalmente, hanno i piedini così configurati, da sinistra a destra: **"S", "+", "-"**.

Mentre i moduli successivi, sempre visti frontalmente da sinistra a destra, i piedini hanno una posizione rovesciata, ovvero: **"-", "+", "S"**, perciò i moduli vanno inseriti ruotati di 180° gradi rispetto ai precedenti.

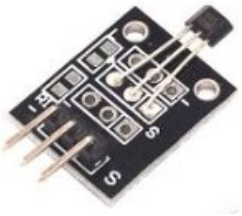
Sen18 water sensor level

KY-010 photo interrupt

KY-021 reed switch sensor



KY-017 mercury tilt module



KY-035 hall magnetic sensor

KY-013 analog temperature

Nota: osservare bene comunque i moduli prima di inserirli, perché a volte, in base alle case costruttrici, queste caratteristiche non sono sempre rispettate. Qualche attimo di attenzione evita di bruciare irrimediabilmente un modulo.

L'ultimo modulo che può essere alloggiato anch'esso su AD2 è un potenziometro, utile in tanti programmi. Ma sia sullo zoccolo AN1 che su AD2 richiede di essere connesso con un cavetto a tre poli, perché la piedinatura è diversa. Il potenziometro può essere alloggiato nativamente sullo zoccolo AN2, creato appositamente.

Un potenziometro con i codici dei piedini.

Nota: i moduli seguenti sono nativamente analogici:

Photo resistor	KY-018 - HW486
Heart bit sensor	KY-039 – HW502
Temperature Humidity	KY-015 – DHT11 - HW507
Water level sensor	Sen18
Analog temperature	KY-013 – HW498
Linear Potentiometer	Linear – 10 KΩ

E quindi nella sezione dei programmi, si troveranno degli sketch per i singoli moduli.

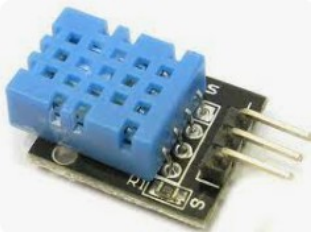
Invece i seguenti moduli che con sistemi diversi sono comunque degli switch, forniscono solo una risposta digitale “0” oppure “1”, ma possono efficacemente essere usati anche su questo zoccolo:

Tap Module	KY-031 – HW500
Shock sensor	KY-002 – HW513
Tilt switch	KY-020 – HW501
Button	KY-004 – HW483
Photo interrupt	KY-010 – HW487
Reed switch sensor	KY-021 – HW-497
Mercury tilt module	KY-017 -HW505
Hall magnetic sensor	KY-035 – HW492

Per cui nella sezioni dei programmi dedicati a AN1, si troveranno un paio di programmi didattici sotto il nome di “**generic switch**”, validi per tutti i moduli indicati. Potranno poi essere personalizzati singolarmente, in base al loro uso.

I programmi per lo zoccolo AN1

Programmi per il sensore DH11 – temperatura e umidità - AN1



DHT11 è un sensore che può essere collegato sia alle porte analogiche che digitali, quindi è molto duttile. Misura sia la temperatura che l’umidità. Oltre che nel programma che segue, semplicemente didattico, viene usato anche in altri progetti sicuramente più interessanti.

Nome del programma: **DHT11_1**

Porte:
A1 (analogico)

Modulo principale:
DHT11 - KY-015 (sens. Temperatura e umidità)

Monitor Seriale: sì (9600 baud)

Plotter seriale: no

Scopo del programma: mostra sul monitor seriale la temperatura e l'umidità.

Note: nessuna

Librerie necessarie: dht.h

Link: https://projecthub.arduino.cc/arcaegecengiz/12f621d5-055f-41fe-965d-a596fcc594f6

Nome del programma: **digital_temp_1**

Porte:
A1 (analogico)

Modulo principale:
DHT11 - KY-015 (sens. Temperatura e umidità)

Porte:
D2/./D7 (digitale)

Comp. Accessori:
Multiplexer HW-178 su zoccolo DY2

Monitor Seriale: sì (9600 baud)

Plotter seriale: sì (9600 baud)

Scopo del programma: Questo progetto, basato sul sensore DHT11, mostra sul monitor seriale come sul progetto precedente. Però avendo collegato un multiplexer, si può accendere uno dei 16 led collegati ad esso, dando una visione grafica della temperatura, in una gamma di 16 gradi. Se la temperatura è inferiore a quella minima, lampeggiano alternativamente i due led a sinistra del multiplexer; se è superiore alla massima, lampeggiano i due presenti alla destra.

Note: La variabile "y" indica il valore minimo di temperatura visualizzata dai led. Per esempio, se Y = 0, la temperatura gestita va da 0 a 15 gradi; se Y = 15, la gamma è compresa tra 15 e 30 gradi e così via.

Link: nessuno

Nome del programma: **digital_temp_2**

Porte:
A1 (analogico)

Modulo principale:
DHT11 - KY-015 (sens. Temperatura e umidità)

Porte:
D2/./D7 (digitale)
A4, A5 (analogiche

Comp. Accessori:
Multiplexer HW-178 su zoccolo DY2
Display Oled 128x64 su zoccolo DY3b

Monitor Seriale: sì (9600 baud)

Plotter seriale: sì (9600 baud)

Scopo del programma: Questo progetto, basato sul sensore DHT11, mostra sul monitor seriale come sul progetto precedente. Però avendo collegato un multiplexer, si può accendere uno dei 16 led collegati ad esso, dando una visione grafica della temperatura, in una gamma di 16 gradi. Se la temperatura è inferiore a quella minima, lampeggiano alternativamente i due led a sinistra del multiplexer; se è superiore alla massima, lampeggiano i due presenti alla destra. Il display Oled mostra la temperatura e umidità.

Note: Vedi nota del programma precedente

Link: nessuno

Programmi per il sensore KY-039 – pulsazioni cardiache - AN1

Il sensore **KY-039** permette di visualizzare la frequenza delle pulsazioni cardiache, come numero sul monitor seriale oppure come grafico sul plotter seriale. Non è particolarmente preciso, ma è comunque interessante ed è un sensore è molto economico. Anche questo programma è molto di base, ma può essere ulteriormente sviluppato. Se si desiderano misurazioni più affidabili, vedere i programmi relativi al sensore MAX30102 (zoccolo AN4).

Nome del programma:	**KY039_1**
Porte:	**Modulo principale:**
A1 (analogico)	KY039 (misura le pulsazioni del cuore)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	mettendo un dito tra il led e il sensore, mostra approssimativamente le pulsazioni del cuore espresso in numero (monitor) o grafico (plotter).
Note:	nessuna
Link:	https://electropeak.com/learn/interfacing-ky-039-finger-heartbeat-measuring-sensor-module-with-arduino/

Programmi per il sensore KY-018, fotoresistenza - AN1

Questo sensore, **KY-018** è basato su di una foto resistenza che cambia la sua resistenza interna in base alla luce, è facile da utilizzare e si presta a molti utilizzi. In questo programma dimostrativo, un led lampeggia in modo progressivo. Nel secondo programma illustrato qui di seguito, può essere collegato un relay che rende possibile accendere una luce di emergenza, attivare un allarme, ecc. Il terzo programma, dotato di un visore LCD, lo rende un pratico strumento di misura.

Nome del programma:	**PHOTO_1**
Porte:	**Modulo principale:**
A1 (analogico)	KY-018 (photoresistor)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Il led lampeggia più velocemente quando c'è luce, lentamente quando è buio, in modo progressivo. Sul monitor seriale appaiono numeri alti (es: 900) quando è buio, numeri bassi (es.: 500) quando c'è luce. Il plotter mostra un grafico alto in mancanza di luce, uno basso in sua presenza.
Note:	nessuna
Link:	nessuno

Nome del programma:	**PHOTO_2**
Porte:	**Modulo principale:**
A1 (analogico)	KY-018 (photoresistor)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
D9 (digitale)	Led L3 - rosso
D9, D11	Led L4 – tre colori
D11	Relay su porta RL1 (contestualmente al Led verde su LD1)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Il led lampeggia più velocemente quando c'è luce, lentamente quando è buio, in modo progressivo. Il led rosso si accende quando c'è buio. Sul monitor seriale appaiono numeri alti (es: 900) quando è buio, numeri bassi (es.: 500) quando c'è luce. Il plotter mostra un grafico alto in mancanza di luce, uno basso in sua presenza. Inserendo un relay sula porta "RL1", si può attivare un allarme o una luce di emergenza in caso di buio.
Note:	Usando un relay, è necessario collegare ad Arduino un alimentatore esterno, che fornisca una tensione compresa tra 6 e 9 volt, e fornisca una corrente di almeno 500 mA.
Link:	nessuno

Nome del programma:	**PHOTO_3**
Porte:	**Modulo principale:**
A1 (analogico)	**KY-018 (photoresistor)**
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
D9 (digitale)	Led L3 - rosso
D9, D10, D11 (digitale)	Led L4 – tre colori
D9 (digitale)	Relay 2 (insieme al led L3, rosso)
D2/./D7 (digitale)	Display LCD 16x2 (connettore DY2)
Monitor Seriale: sì ***(9600 baud)***	**Plotter seriale:** sì (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono numeri alti (es: 900) quando è buio, numeri bassi (es.: 500) quando c'è luce. Il plotter mostra un grafico alto in mancanza di luce, uno basso in sua presenza. Il led L1 (verde) si accende se c'è luce buona (valore: inferiore a 500); spento in caso di luce sufficiente. In questo caso si accende anche il led L3 (rosso) a indicare un anomalia e si attiva il relay, che può pilotare per esempio un allarme oppure fare accendere una luce di emergenza. Il display LCD (16 caratteri, su 2 linee), indica il valore della fotoresistenza. Se il valore è inferiore a 550, appare sulla seconda riga "Luce buona"; altrimenti "Luce scarsa".
Note:	La luminosità del visore LCD può essere variata agendo sul trimmer "TM2" (dim). Il display LCD insieme al relay ha un discreto assorbimento di corrente. Nel caso Arduino non riuscisse a fornire una corrente sufficiente, sarà necessario collegare su "Ext. Power" un alimentatore (tensione compresa tra 6 e 9 v, min. 500 mA)
Librerie necessarie:	LiquidCrystal.h
Link:	nessuno

Programmi per i potenziometri - AN1

I potenziometri sono oggetti molto comuni nella maggior parte degli apparecchi elettronici, infatti modificandola resistenza in base alla rotazione di un cursore, possono variare luminosità, velocità di rotazione di un motore, il volume…
Il secondo programma proposto, permette di sperimentare le potenzialità PWM (Pulse Width Modulation) presenti in Arduino: Ruotando la manopola del potenziometro, i led cambieranno luminosità. Il potenziometro verrà usato anche per gestire i motori (zoccoli DG8 e DG9). Alloggiandolo su questo zoccolo, deve essere collegato alla bs con un connettore a tre cavi.

Nome del programma:	**POT_1**
Porte:	**Modulo principale:**
A1 (analogico)	Potentiometer (10 KΩ, lineare)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. Il led L1 lampeggia con frequenza inversamente proporzionale ai valori.
Note:	nessuna
Link:	nessuno

Nome del programma:	**POT_1a**
Porte:	**Modulo principale:**
A1 (analogico)	Potentiometer (10 KΩ, lineare)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
D10 (digitale)	Led L2 - blu
D9 (digitale)	Led L3 - rosso
D9, D10, D11 (digitale)	Led L4 – tre colori
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra 1023 e 0. I tre led, lampeggiano alternativamente, con una frequenza che varia in base alla rotazione del potenziometro
Note:	nessuna
Link:	nessuno

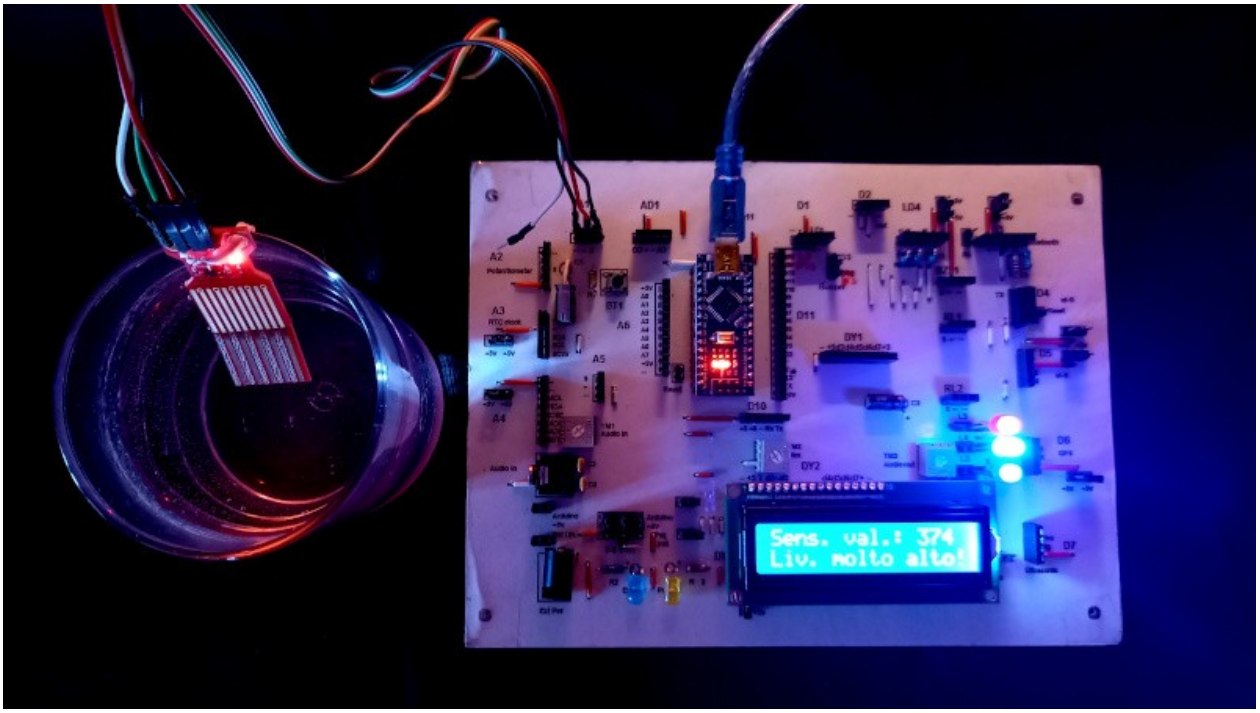
Nome del programma:	**POT_2**
Porte:	**Modulo principale:**
A1 (analogico)	Potentiometer (10 KΩ, lineare)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
D10 (digitale)	Led L2 - blu
D9 (digitale)	Led L3 - rosso
D9, D10, D11 (digitale)	Led L4 – tre colori
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. I tre led, collegati alle porte PWM, cambiano tutti insieme intensità di luce. Spenti a valore 0, luminosità massima a valori alti.
Note:	nessuna
Link:	nessuno

Programmi per il sensore Sen18 (liquidi) - AN1

Il sensore di liquidi **Sen18** cambia il valore presente sulla porta analogica A1 in base a quanto sia immerso nel liquido. La variazione è più sensibile nella parte bassa del sensore e decresce immergendolo di più. Verificare sperimentalmente i valori riscontrati con liquidi diversi dall'acqua. Attenzione ai liquidi corrosivi, che possono danneggiare il sensore!

Nome del programma:	**Water Sensor_1**
Porte:	**Modulo principale:**
A1 (analogico)	Water Sensor Sen18. Questo sensore varia il valore della porta analogica in proporzione al livello del liquido in cui è immerso.
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 - rosso
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. Superata una certa soglia (variabile treshold), il led, collegato alla porta PWM, cambia intensità di luce in base al livello del liquido.
Note:	nessuna
Link:	nessuno

Nome del programma:	**Water Sensor_2**
Porte:	**Modulo principale:**
A1 (analogico)	Water Sensor Sen18. Questo sensore varia il valore della porta analogica in proporzione al livello del liquido in cui è immerso.
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 - rosso
D2/./D7	Dispaly LCD 1602 (16 caratteri x 2 linee)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. Superata una certa soglia (variabile treshold), il led, collegato alla porta PWM, cambia intensità di luce in base al livello del liquido. Sul display LCD appaiono i valori di immersione nel liquido (sensorValue) e quelle del valore quando si è superata la soglia (outputValue).
Librerie richieste:	LiquidCrystal.h
Note:	nessuna
Link:	nessuno



Nome del programma:	**Water Sensor_3**
Porte:	**Modulo principale:**
A1 (analogico)	Water Sensor Sen18. Questo sensore varia il valore della porta analogica in proporzione al livello del liquido in cui è immerso.
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 – rosso
D10 (digitale)	Led L2 - blu
D11 (digitale)	Led L1 verde
D11 (digitale)	Relay KY-019 (opzionale) su RL1
D13 (digitale)	Buzzer KY-006) oppure KY-012 su BZ1
D2/./D7	Dispaly LCD 1602 (16 caratteri x 2 linee) du DY2
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023 per il livello del liquido. Vengono anche indicati alcuni suggerimenti (livello basso…). Superata una certa soglia (variabile treshold), appaiono anche questi nuovi valori e il led, collegato alla porta PWM, cambia intensità di luce in base al livello del liquido. E' stato aggiunto un buzzer. Emette una nota bassa quando il livello è insufficiente e il led rosso è acceso; è silenzioso quando il livello è corretto; emette una nota alta quando il livello è alto e una ancora più alta quando il livello è critico. Quando il livello è alto, il relay si attiva e si accende il led verde. Il led blu aumenta di intensità in base al livello del liquido, mentre lampeggia quando il livello è critico e si accenda anche il led rosso. I tre led possono anche essere sostituiti dal led a tre colori, da inserire (attenzione alla piedinatura!) su LD4. I dati del valore del livello del liquido e dei suggerimenti appaiono anche sul display LCD.
Librerie richieste:	LiquidCrystal.h
Note:	Se si usa il Relay, è consigliabile alimentare Arduino con un alimentatore esterno. Attenzione: la tensione di rete è pericolosa, e può essere anche mortale!
Link:	nessuno

Programmi per il sensore KY-013 (temperatura analogica) – AN1

Il modulo **KY-013** (HW498) è un sensore analogico di temperatura, utile per programma re apparecchi che devono attivarsi/disattivarsi raggiunta una certa soglia di temperatura.

Nome del programma: Analog_temp_1
Porte: **Modulo principale:**
A1 (analogico) Sensore analogico di temperatura KY-013
Porte: **Comp. Accessori:**
Monitor Seriale: sì **Plotter seriale: sì** (9600 baud)
(9600 baud)
Scopo del programma: Sul monitor seriale appaiono i valori di temperatura, sia in gradi Celsius che Kelvin, oltre che il valore analogico e la tensione in Volt.
Librerie necessarie: math.h
Note: nessuna
Link: nessuno

Nome del programma: Analog_temp_2
Porte: **Modulo principale:**
A1 (analogico) Sensore analogico di temperatura KY-013
Porte: **Comp. Accessori:**
D2/./D7 (digitali) Display LCD 1602 (16 caratteri su 2 linee) su zoccolo DY2
Monitor Seriale: sì **Plotter seriale: sì** (9600 baud)
(9600 baud)
Scopo del programma: Sul monitor seriale appaiono i valori di temperatura, sia in gradi Celsius che Kelvin. Gli stessi dati appaiono anche sul dispaly LCD
Librerie necessarie: math.h; LiquidCrystal.h
Note: nessuna
Link: nessuno

Programmi per switch generici (KY-002, KY-004, KY-010, KY-017, KY-020, KY-021, KY-031, KY-035) - AN1

Ci sono molti semplici sensori che hanno una funzione di interruttore o pulsante (quindi funzione acceso/spento), che possono essere utilizzati con profitto anche al connettore analogico AN1. Visto che hanno tutti lo stesso principio di funzionamento, e che sono numerosi, si è pensato di usare un programma generico per gestirli tutti; chi lo desidera potrà personalizzarli singolarmente, in base alle proprie esigenze.
Per semplicità, ecco la lista:

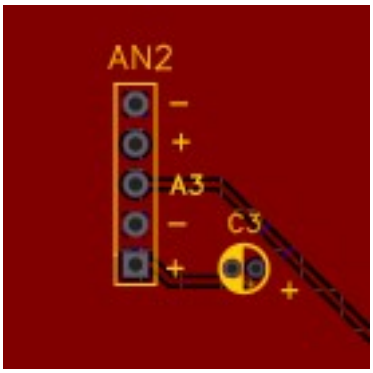
Generic switch

Socket AN1 (analogic)			1	2	3
Description	Code	Note	-	vcc (+5v)	S
Tap Module	KY-031 – HW500				
Shock sensor	KY-002 – HW513				
Tilt switch	KY-020 – HW501				
Button	KY-004 – HW483				
Photo interrupt	KY-010 – HW487	piedinatura ruotata			
Reed switch sensor	KY-021 – HW-497	piedinatura ruotata			
Mercury tilt module	KY-017 -HW505	piedinatura ruotata			
Hall magnetic sensor	KY-035 – HW492	piedinatura ruotata			

Nome del programma:	**Switch_1**
Porte:	**Modulo principale:**
D15 (digitale). Porta: A1*	Vedi lista “generic switch”
Porte:	**Comp. Accessori:**
D2/./D7 (digitale)	Dispaly LCD 1602 (16 caratteri x 2 righe di testo)
D9 (digitale)	Led L3 - rosso
D9 (digitale)	Relay KY-019 su RL2
D11 (digitale)	Led L1 - verde
D13 (digitale)	Buzzer KY-006 (KY,012)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. In stato di quiete il led LD1 (verde è acceso). Superata una certa soglia (variabile treshold), il led LD3 (rosso) si accende; nel caso fosse inserito ul relay sulla porta RL2, si attiverebbe. Inoltre il buzzer suona una nota. Le stesse informazioni sono mostrate sul display LCD.
Note:	nessuna
Link:	nessuno

Questo è un programma generico, che può funzionare con ognuno di questi moduli. Naturalmente si può perfezionare per ogni sensore, e può essere inglobato in programmi più complessi.

Lo zoccolo AN2



AN2 si trova in alto sul lato sinistro della bs (come la maggior parte delle porte analogiche) e si connette alla porta A3.
Va in conflitto con la porta AD2, perché usa la porta D17, che fisicamente corrisponde appunto ad A3.
Un potenziometro è molto utile per variare la luminosità di un led, la velocità di un motore elettrico (collegati su una porta digitale PWM), ecc.
Correggendo adeguatamente i programmi per il PIR (posizione standard: zoccolo DG2), è possibile inserirlo anche su questo zoccolo, utilizzando i piedini centrali, modificando nel programma la porta digitale usata, che non sarà più D8, bensì D17.

Socket AN2 (analogic)			1	2	3	4	5
Description	Code	Note	**vcc** (+5v)	**vcc** (+5v)	**S**	-	-
Potentiometer	---		**+5v**	/	**a3**	/	-
PIR	KY-007/HC-SR501		/	**+5v**	**d17***	-	/

* = utilizzando la possibilità di usare una porta analogica (A3) come una digitale (D17)

AN2. Questo zoccolo è stato progettato per ospitare nativamente un potenziometro (abitualmente lineare da 10 Kohm), utile per esempio per controllare la velocità di un motore elettrico collegato su DG8, l'inclinazione dell'albero di un servomotore collegato su DG9, la luminosità di un led, ecc. Gli utilizzi sono svariati. Si collega sulla porta analogica A3, quindi potrebbe entrare in conflitto con AD2. Fare attenzione durante la programmazione. Un potenziometro potrebbe essere connesso anche su AN1, come segnalato nell'elenco precedente, ma in questo caso è necessario usare un connettore a tre cavetti, perché il passo dei fori non corrisponde a quello di un potenziometro e anche la disposizione dei piedini è diversa.

Potenziometro

PIR - KY-007/HC-SR501

Nel piccolo riquadro relativo ad AN1 si vede la presenza in parallelo allo zoccolo di un condensatore elettrolitico (quindi polarizzato) da 100 µF. Esso è stato inserito in previsione del controllo di un motore o di un servomotore. Quando essi iniziano a muoversi, utilizzano più corrente, causando un calo di tensione nella scheda. Il condensatore, detto di disaccoppiamento, compensa questo calo. Ce n'è uno uguale anche nei pressi di DG9, a cui si può collegare un servomotore.

Su questo zoccolo è possibile montare anche il modulo PIR (sensore di movimento) che abitualmente si pone sullo zoccolo DG3 (porta digitale D8), modificando il programma, che userà ora la porta D17 (fisica: A3).

Come per altri zoccoli, AN2 può accogliere i moduli che montano su AN1, naturalmente è necessario usare un connettore perché la disposizione dei piedini è diversa. E' necessario, come indicato per AD2, di modificare la porta analogica di collegamento, da A1 a A3.

Programmi per potenziometri - AN2

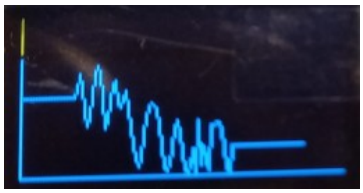
Nome del programma:	**POT_1a**
Porte:	**Modulo principale:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. Il led L1 lampeggia con frequenza inversamente proporzionale ai valori.
Note:	nessuna
Link:	nessuno

Nome del programma:	**POT_2a**
Porte:	**Modulo principale:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
D10 (digitale)	Led L2 - blu
D9 (digitale)	Led L3 - rosso
D9, D10, D11 (digitale)	Led L4 – tre colori
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. I tre led, collegati alle porte PWM, cambiano tutti insieme intensità di luce. Spenti a valore 0, luminosità massima a valori alti.
Note:	nessuna
Link:	nessuno

Nome del programma:	**POT_2c**
Porte:	**Modulo principale:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
Porte:	**Comp. Accessori:**
D11 (digitale)	Led L1 - verde
D10 (digitale)	Led L2 - blu
D9 (digitale)	Led L3 - rosso
D9, D10, D11 (digitale)	Led L4 – tre colori
D13	Buzzer KY-006 (KY-012) su BZ1
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Sul monitor seriale appaiono dei valori analogici, tra o e 1023. I tre led, collegati alle porte PWM, cambiano tutti insieme intensità di luce. Spenti a valore 0, luminosità massima a valori alti. Allo stesso modo, il buzzer emetterà un suono da frequenza bassa ad alta.
Note:	nessuna
Link:	nessuno

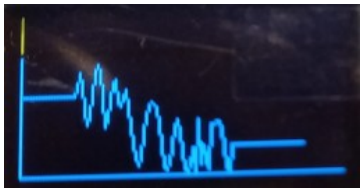
Nota: i programmi “Pot_1a” 2 “Pot_2a” hanno un suffisso “a”, perché sono identici a quelli presenti per il connettore A1, semplicemente è stata cambiata la porta su cui è presente il potenziometro.

Nome del programma: **POT_3_32**
Porte: **Modulo principale:**
A1 (analogico) Potentiometer (10 KΩ, lineare)
Porte: **Comp. Accessori:**
DY3 A4 (DTA); A5 (SCL) Display Oled 128x32
Monitor Seriale: sì **Plotter seriale: sì** (9600 baud)
(9600 baud)


Il grafico sul dispaly

Scopo del programma: Sul piccolo display OLED appare un grafico, proporzionale alla rotazione della manopola del potenziometro, con una tensione che varia da 0 a 5 v, potendo diventare la base di un piccolo tester.
Librerie richieste: Wire.h; Adafruit_GFX.h; Adafruit_SSD1306.h
Note: nessuno
Link: nessuno

Nome del programma: **POT_3_64**
Porte: **Modulo principale:**
A1 (analogico) Potentiometer (10 KΩ, lineare)
Porte: **Comp. Accessori:**
DY3 A4 (DTA); A5 (SCL) Display Oled 128x64
Monitor Seriale: sì **Plotter seriale: sì** (9600 baud)
(9600 baud)


Il grafico sul dispaly

Scopo del programma: Sul piccolo display OLED appare un grafico, proporzionale alla rotazione della manopola del potenziometro, con una tensione che varia da 0 a 5 v, potendo diventare la base di un piccolo tester.
Librerie richieste: Wire.h; Adafruit_GFX.h; Adafruit_SSD1306.h
Note: Inessuno
Link: nessuno

I programmi per il modulo KY-007 HC-SR501 (sensore di movimento) - AN2

Il modulo PIR monta nativamente sullo zoccolo DG2. Nel caso però che la porta D8 sia già utilizzata da un altro modulo, oppure che in un progetto (per esempio di un antifurto per casa) se ne desideri montare un paio, ecco che può essere utile poterne montare un secondo su di un altro zoccolo, in questo caso AN2, trasformando la porta analogica in A3 in una porta digitale, ovvero D17.
E’ sufficiente modificare la porta gestita dal programma, perché il progetto funzioni. Vedi modifiche.

Programma originale:

```
pir_1
#define PIR_PIN 8
#define LED1_PIN 9
#define BUZZER_PIN 13
```

Il programma originale, progettato per DG2,
usa la porta digitale **D8**

Programma modificato:

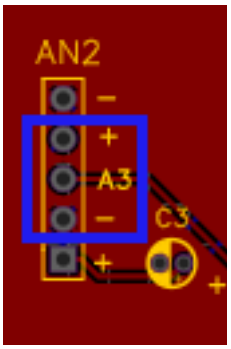
```
pir_1a
#define PIR_PIN 17
#define LED1_PIN 9
#define BUZZER_PIN 13
```

Il programma modificato per funzionare su AN2,
usa la porta **digitale D17** (fisica: A3)
Se necessario, vedi **come trasformare**
le porte analogiche in digitali.

Nome del programma:	**Pir_1a**
Porte:	**Modulo principale:**
D8 (digitale)	**HC-SR501,** sensore pir di movimento, sensibile ai raggi infrarossi.
Monitor Seriale: no	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	L2 – led rosso
D13 (digitale)	Buzzer – KY-006 (KY-012)
Scopo del programma:	Il sensore PIR segnala un movimento in prossimità (max. 2/3 metri), producendo un segnale "High", ovvero 1, quando percepisce un movimento. La sensibilità si regola con un trimmer; la durata del segnale con un secondo. Quando si verifica un movimento, lampeggia il led rosso e se su RL2 è stato inserito un relay, esso può chiudere un circuito di allarme; contemporaneamente il buzzer emette una serie di note di allarme, che possono essere trasmesse attraverso la linea "audio out" a un amplificatore esterno.
Note:	nessuna
Link:	nessuno

Anche gli altri programmi per il modulo PIR (vedi DG2), possono essere modificati allo stesso modo e funzionare sullo zoccolo AN2.

Se si intende utilizzare il modulo PIR su AN2, vanno usati i pin ***centrali*** del connettore, facendo attenzione alle polarità



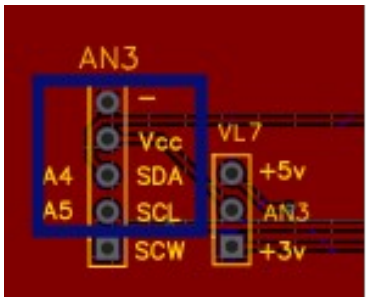
Un semplice voltmetro digitale

Nome del programma:	**volt_1**
Porte:	**Modulo principale:**
A1 (analogico)	Misura diretta della tensione in ingresso
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì** (9600 baud)
Scopo del programma:	Di tensione misurati sulla porta A1
Link:	nessuno

Nota: Questo progetto è stato inserito solo in via sperimentale, quindi utilizzarlo con attenzione.
Sulle porte di Arduino è possibile applicare una tensione massima di 5 volt, altrimenti si rischia di bruciare la porta di ingresso o addirittura Arduino stesso.
Applicare la tensione positiva sul connettore "S" di A1 e quella negativa sul connettore "-".
Attenzione a non invertire la polarità, pena il danneggiamento di Arduino.

In futuro inseriremo qualche sicurezza per evitare danneggiamenti, ed eventualmente la possibilità di misurare tensioni superiori a 5 volt.

Lo zoccolo AN3



AN3 si colloca sulla parte sinistra della bs, sotto ad AN2. Le porte utilizzate sono A4 e A5, che possono essere condivise con i moduli presenti su AN4 e DY3 a e b. E' sufficiente che i moduli interessati abbiano indirizzi interni differenti.
Attenzione a utilizzare la tensione giusta per il modulo che si inserirà!
Sia DS1307 che il display LCD102 con adattatore I2C usano solo i primi quattro pin.

AN3. Questo zoccolo si connette a due porte un po' speciali, A4 (SDA - dati) e A5 (SCL o SCK – clock). A queste porte possono essere connessi ***contemporaneamente*** più moduli, a patto che abbiano indirizzi interni diversi. Questa porta è stata pensata per un modulo particolare, ovvero RTC clock module, DS1307. Collegandolo ad Arduino, possiamo ottenere ora, data, giorno della settimana con grande precisione. Perciò il nostro microcontroller si può trasformare, abbinandolo a un display, in un orologio digitale; oppure lo possiamo utilizzare in tutti i programmi in cui la precisa misura del tempo sia necessaria. Questo modulo funziona a +5v; è stato inserito nella prossimità dello zoccolo anche un jumper per la selezione del voltaggio, utile nel caso si volesse collegare su di esso qualche modulo che funziona a +3,3v. Il piedino contrassegnato "SCW" non è utilizzato nella nostra *bs*.
Ponticellando correttamente VL7, si possono alimentare correttamente i moduli alla giusta tensione.

Moduli che si collegano su AN3

Socket AN3 (analogic)			1	2	3	4	5
Description	Code	Note	GND	vcc (+5v)	SDA	SCL	SCW
RTC Digital Clock	DS-1307		-	+	a4	a5	/
Display LCD	1602A	richiede un adattatore I2C.	-	+	A5	A4	/

Immagini dei moduli che si connettono su AN3

DS1307 alloggia nativamente sullo zoccolo AN3

Display LCD 1602 adattatore I2C

Programmi per lo zoccolo AN3

Scanner per trovare l'indirizzo del modulo I2C

Nome del programma:	**I2C_scanner**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	**Moduli IC2**. Modulo che restituisce temperatura, pressione e fa un calcolo approssimativo dell'altitudine.
Porte:	**Comp. Accessori:**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale l'indirizzo del modulo. Sulle porte SDA/SCK (IC2) possono essere collegati più moduli, sempre che abbiano indirizzi diversi. Adatto anche per altri moduli IC2.
Note:	nessuna
Librerie richieste:	Wire.h
Link:	nessuno

Il programma "I2C_scanner" serve per verificare l'indirizzo per i moduli che montano su questo zoccolo, come anche per AN4 e DY3.

Programmi per modulo DS1307 (clock) - AN3

Il modulo **DS1307** è un preciso orologio, che restituisce ora, data e giorno della settimana. Funziona con il protocollo I2C

Nome del programma:	**DS1307_1**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	Modulo Ds1307, RTC (orologio)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale la data odierna, il giorno della settimana e ore, minuti, secondi
Note:	nessuna
Librerie richieste:	Wire.h; RTClib.h
Link:	nessuno

Nome del programma:	**DS1307_2**
Porte:	**Modulo principale:**
A4 – SDA;	**Modulo Ds1307, RTC (orologio)**
A5 - SCK (analogico)	
Porte:	**Comp. Accessori:**
D2/./D7 (digitali)	Lcd Display 1602, 16x2 characters su DY2
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Mostra sul monitor seriale la data odierna e ore, minuti, secondi, che vengono mostrati anche sul dispaly LCD, trasformandolo in un orologio a tutti gli effetti.
Note:	nessuna
Librerie richieste:	LiquidCrystal.h; Wire.h; RTClib.h; DS3231.h
Link:	nessuno

Nome del programma:	**DS1307_3**
Porte:	**Modulo principale:**
A4 – SDA;	**Modulo Ds1307, RTC (orologio)**
A5 - SCK (analogico)	
Porte:	**Comp. Accessori:**
D3/./D7 (digitali)	Dispaly ST7735, TFT 1,7” color
A1 (analogica)	Sens. Temeratura e umidità DHT11 su AN1
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Mostra sul monitor seriale la data odierna, il giorno della settimana; ore, minuti, secondi e temperatura e umidità. Arduino diventa un completo orologio digitale
Note:	nessuna
Librerie richieste:	Wire.h; RTClib.h; Adafruit_GFX.h; Adafruit_ST7735.h; SPI.h; dht.h
Link:	https://www.lombardoandrea.com/interfacciare-modulo-ds1307-con-arduino/

L’orologio che si ottiene caricando il programma DS1307_3 e utilizzando il display ST7735

Programmi per display LCD 1602 con adattatore I2C - AN3

vista posteriore del display LCD 1602 con adattatore I2C

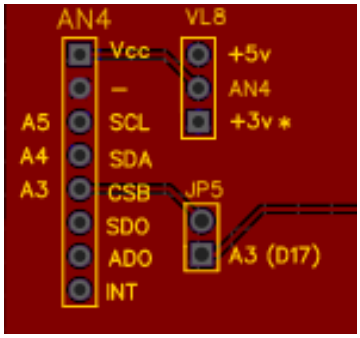
Il display LCD 1602 nativamente utilizza ben 6 porte digitali (sulla *bs* da D2 a D7), per cui non sempre è di facile collocazione, specie con progetti complessi, con vari sensori. Applicando invece il piccolo adattatore I2C, utilizza solamente le porte A4 – SDA e A5 SCL, che può condividere anche con altri moduli, sempre che abbiano indirizzi diversi. Effettuando alcuni ponticelli, è possibile variare l'indirizzo interno del display, per evitare conflitti (**vedi tabella**).

Nome del programma:	**LCD_i2c_1**
Porte: A4 – SDA; A5 - SCK (analogico)	**Modulo principale:** **Dispaly LCD 1602 con adattatore I2C**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale una serie di scritte
Note:	Usare solo le prime quattro porte in alto dello zoccolo AN3
Librerie richieste:	LiquidCrystal_I2C.h
Link:	https://win.adrirobot.it/display_lcd/display-lcd-i2c-16x2-con-retroilumminazione-blu.htm

Nome del programma:	**LCD_i2c_2**
Porte: A4 – SDA; A5 - SCK (analogico)	**Modulo principale:** **Dispaly LCD 1602 con adattatore I2C**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Un secondo programma didattico. Mostra sul monitor seriale una serie di scritte scorrevoli.
Note:	Usare solo le prime quattro porte in alto dello zoccolo AN3
Librerie richieste:	LiquidCrystal_I2C.h
Link:	https://www.meccanismocomplesso.org/lcd1602-utilizzare-un-display-a-cristalli-liquidi-lcd-con-arduino-tramite-i2c/

Nome del programma:	**LCD_i2c_3**
Porte: A4 – SDA; A5 - SCK (analogico)	**Modulo principale:** **Dispaly LCD 1602 con adattatore I2C su AN3**
Monitor Seriale: sì **(**9600 baud)	**Plotter seriale: no**
Porte: RST = D10 DAT = D9 CLK = D7	**Comp. Accessori:** Rtc clock DS1302 su zoccolo DG11
Scopo del programma:	Usando un RTC clock DS1302 e il display LCD, si trasforma Arduino Nano in un perfetto orologio da tavolo.
Note:	Usare solo le prime quattro porte in alto dello zoccolo AN3
Librerie richieste:	LiquidCrystal_I2C.h; DS1302.h; Wire.h
Link:	https://www.meccanismocomplesso.org/lcd1602-utilizzare-un-display-a-cristalli-liquidi-lcd-con-arduino-tramite-i2c/

Lo zoccolo AN4



Lo zoccolo AN4 si trova a sinistra, immediatamente sotto a AN3, e utilizza le stesse porte, ovvero A4 e A5.

Può ospitare alcuni moduli, porre molta attenzione all'orientamento del sensore e alla corretta tensione di alimentazione.

AN4. Su questo zoccolo si collegano diversi moduli interessanti. Come per AN3, esso si connette alle porte A4 e A5, potendo in linea di massima funzionare contemporaneamente ad altri moduli che si connettono sulle stesse porte, a patto che abbiano indirizzi diversi. Nella lista dei programmi ne è presente uno particolare, che serve per verificare l'indirizzo di ogni sensore. Si è inserito anche questo zoccolo sulla *bs*, perché la piedinatura è leggermente diversa. Questo zoccolo possiede addirittura otto connettori, per poter alloggiare i vari moduli, ma abitualmente sono utilizzati solamente i primi quattro. Il jumper JP5 permette di attivare il collegamento della porta A3/D17 sul quinto piedino, marcato "SDO", nel caso si dovesse collegare a questo versatile zoccolo un modulo che richieda tre porte analogiche, o addirittura 3 porte digitali (D19, D18, D17) o un mix delle possibili combinazioni. La porta A3 viene attivata attraverso il jumper JP8, per evitare che questo porta quando non utilizzata possa andare in conflitto con altri moduli, presenti per esempio su AN2 o AD2.

Su questo zoccolo si può inserire un **BMP280**, in grado di trasformare Arduino in una piccola stazione meteorologica, fornendo la temperatura, la pressione atmosferica e una stima dell'altezza; **Max30102** misura le pulsazioni del cuore, l'ossigenazione del sangue e la temperatura del corpo. *Questi due moduli funzionano a +3,3v!* ***GY521*** è un accelerometro/giroscopio a 3 assi, un sensore di temperatura e anche altro.

Ponticellando correttamente VL8, si possono alimentare i vari moduli alla giusta tensione.

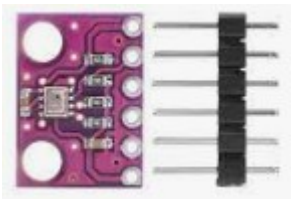
I sensori BMP280 e GY521 sono collegati direttamente sullo zoccolo; BMP280 usa solamente i primi sei piedini partendo dall'alto; GY521 li usa tutti. Fare attenzione all'orientamento; i piedini "vcc" vanno connessi al connettore "+" dello zoccolo. Il sensore MAX30102 invece richiede di essere connesso a AN4 con un connettore a quattro cavi. Ricordarsi di porre il jumper di alimentazione su +3,3v

Moduli che si collegano su AN4

Socket AN4 (analogic)			1	2	3	4	5	6	7	8
Description	Code	Note	Vcc	GND	SCL	SDA	CSB	SDO	/	/
Weather station	BMP280		+ 5v	-	a5	a4	/	/	/	/
Accelerometer/Giroscope	GY-521 - MPU-6050		VCC	GND	SCL	SDA	XDA	XCL	ADO	INT
			+ 3,3 v	-	a5	a4	/	/	/	/
Pulse – O_2 sensor	Max30102	Usare connettore – diversa piedinatura Alimentazione 3,3v	+ 3,3 v	-	a5	a4	/	/	/	/
Light sensor	GY-302 – BH1750		+ 5v	-	a5	a4	/	/	/	/
GPS NEO7			+ 5v	-	(a5) d19	(a4) d18				

Nota: in caso di necessità, utilizzando un connettore a quattro cavetti, gli stessi moduli possono essere collegati anche sullo zoccolo AN3, senza effettuare cambiamenti nei programmi.

Immagini dei moduli che si connettono su AN4



BMP280 – weather station

Max30102 – heart pulse sensor

GY521 - accelorometer/gyroscope

GY302 – light sensor

GPS NEO 7M

Note:

- BMP280 monta nativamente sullo zoccolo AN4, ma si connette solo i primi sei connettori dello zoccolo, di cui usa effettivamente solo i primi quattro!
- Max 30102 ha una piedinatura diversa e necessita di essere collegato con un connettore a 4 cavi, sui primi 4 piedini in alto.
- GY521 alloggia nativamente sullo zoccolo, e si inserisce su tutti i connettori dello zoccolo, ma usa solamente i primi quattro.
- Se il modulo GPS Neo 7M viene inserito su questo zoccolo, occupa le porte A4 (D18) e A5 (D19), che non possono più essere utilizzate per i moduli con protocollo I2C. Questo è sicuramente uno svantaggio, pero si liberano le porte D9 e D10, che possono essere utilizzate per collegare contestualmente un modulo wi-fi o bluetooth.

Valgono le stesse osservazioni relative ad AN3. Tutti i seguenti moduli funzionano con il protocollo I2C e oltre all'alimentazione usano solo due piedini per i dati, connessi su A4 e A5. Se hanno un indirizzo interno diverso, possono essere anche utilizzati contemporaneamente ai moduli inseriti su AN3 e DY3.

Programmi per AN4

Scanner per trovare l'indirizzo del modulo I2C

Nome del programma:	**I2C_scanner**
Porte: A4 – SDA; A5 - SCK (analogico)	**Modulo principale:** **Moduli IC2**. Modulo che restituisce temperatura, pressione e fa un calcolo approssimativo dell'altitudine.
Porte:	**Comp. Accessori:**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale l'indirizzo del modulo. Sulle porte SDA/SCK (IC2) possono essere collegati più moduli, sempre che abbiano indirizzi diversi. Adatto anche per altri moduli IC2.
Librerie richieste:	Wire.h
Note:	nessuna
Link:	nessuno

Il programma di controllo dell'indirizzo del modulo è lo stesso presentato nell'introduzione di AN3.

Programmi per modulo BMP280 – stazione meteorologica - AN4

Il sensore **BMP280** è veramente minuscolo, eppure riesce al monitorare la temperatura, la pressione atmosferica e in modo estremamente approssimativo, l'altitudine del luogo in cui ci si trova. Con pochi altri componenti può diventare una piccola stazione meteorologica!

BMP280 funziona a 3,3 volt!

Nome del programma:	**BMP280_1**
Porte: A4 – SDA; A5 - SCK (analogico)	**Modulo principale:** BMP280. Modulo che restituisce temperatura, pressione e fa un calcolo approssimativo dell'altitudine
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale la temperatura, la pressione dell'aria e l'altezza approssimativa
Note:	Meglio non fidarsi eccessivamente sul calcolo dell'altitudine! Richiede alimentazione a 3,3 volt
Librerie richieste:	Wire.h; Adafruit_Sensor.h; Adafruit_BMP280.h
Link:	https://iotprojectsideas.com/interface-bmp280-sensor-with-arduino/

Nome del programma:	**BMP280_2**
Porte:	**Modulo principale:**
A4 – SDA;	BMP280. Modulo che restituisce temperatura, pressione e fa un
A5 - SCK (analogico)	calcolo approssimativo dell'altitudine
Porte:	**Comp. Accessori:**
D2/./D7 (digitali)	Lcd Display 1602, 16x2 characters su DY2
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Mostra sul monitor seriale la temperatura, la pressione dell'aria e l'altezza approssimativa. Sul dispaly LCD mostra temperatura e pressione atmosferica.
Librerie richieste:	Wire.h; Adafruit_Sensor.h; Adafruit_BMP280.h; LiquidCrystal.h
Note:	Meglio non fidarsi eccessivamente sul calcolo dell'altitudine! Richiede alimentazione a 3,3 volt
Link:	nessuno

Nome del programma:	**BMP280_3**
Porte:	**Modulo principale:**
A4 – SDA;	BMP280. Modulo che restituisce temperatura, pressione e fa un
A5 - SCK (analogico)	calcolo approssimativo dell'altitudine.
Porte:	**Comp. Accessori:**
D3/./D7 (digitali)	Dispaly TFT 1,77" 160(RGB)x128 su DY1
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Mostra sul monitor seriale la temperatura, la pressione dell'aria e l'altezza approssimativa, come pure sul dispaly TFT.
Librerie richieste:	Wire.h; Adafruit_Sensor.h; Adafruit_BMP280.h; Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Meglio non fidarsi eccessivamente sul calcolo dell'altitudine! Richiede alimentazione a 3,3 volt
Link:	nessuno

Nome del programma:	**BMP280_4**
Porte:	**Modulo principale:**
A4 – SDA;	BMP280. Modulo che restituisce temperatura, pressione e fa un
A5 - SCK (analogico)	calcolo approssimativo dell'altitudine
Porte:	**Comp. Accessori:**
A1	DHT11
D3/./D7 (digitali)	Dispaly TFT 1,77" 160(RGB)x128
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Mostra sul monitor seriale la temperatura, la pressione dell'aria e l'altezza approssimativa. Sul dispaly TFT mostra temperatura, pressione atmosferica e altezza stimata. Utilizzando il DHT11, mostra anche l'umidità e la temperatura più precisa. La grafica è più accurata.

Librerie richieste:	Wire.h; Adafruit_Sensor.h; Adafruit_BMP280.h; Adafruit_GFX.h; Adafruit_ST7735.h ; dht.h
Note:	Meglio non fidarsi eccessivamente sul calcolo dell'altitudine! Richiede alimentazione a 3,3 volt
Link:	nessuno

Nome del programma:	**BMP280_5**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	BMP280. Modulo che restituisce temperatura, pressione e fa un calcolo approssimativo dell'altitudine su AN4
A4 – SDA; A5 - SCK (analogico)	Si aggiunge, sulle stesse porte, anche il modulo DS1307 (RTC) su AN3
Porte:	**Comp. Accessori:**
A1	DHT11 su AN1
D3/./D7 (digitali)	Dispaly TFT 1,77" ST7735 su DY1
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**

Weather station

Scopo del programma:	Mostra sul monitor seriale la temperatura, la pressione dell'aria e l'altezza approssimativa. Sul dispaly TFT mostra temperatura, pressione atmosferica e altezza stimata. Utilizzando il DHT11, mostra anche l'umidità e la temperatura più precisa. La grafica è più accurata. Inserendo il modulo RTC, appare anche la data e l'ora.
Librerie richieste:	Wire.h; Adafruit_Sensor.h; Adafruit_BMP280.h; Adafruit_GFX.h; Adafruit_ST7735.h ; dht.h; RTClib.h; DS3231.h
Note:	Meglio non fidarsi eccessivamente sul calcolo dell'altitudine! Richiede alimentazione a 3,3 volt
Link:	nessuno

Programmi per modulo GY-521 (accelerometro – giroscopio) – AN4

Il modulo **GY-521** per quanto minuscolo, è ricco di funzioni, come indicato nella descrizione del modulo. Esso contiene un accelerometro e un giroscopio MEMS. E' molto accurato e contiene un convertitore analogico/digitale a 16 bit per ogni canale, e quindi catturare le informazioni dei canali x, y e z contemporaneamente. Inoltre contiene anche un sensore di temperatura.

Nome del programma: **GY521_1**

Porte:
A4 – SDA;
A5 - SCK (analogico)

Modulo principale:
Il modulo **GY521** è molto versatile: accelerometro/giroscopio a 3 assi; sensore di temperatura; oscillatore di clock interno; un processore di movimento digitale.

Monitor Seriale: sì (9600 baud)
Plotter seriale: no

Scopo del programma: Mostra sul monitor seriale l'accelerazione sui 3assi; la posizione girscopica, sempre sui tre assi, e anche la temperatura.

Librerie richieste: Wire.h

Note: Verificare con lo scanner I2C l'indirizzo del sensore e se necessario cambiare l'indirizzo nel programma.

Link: nessuno

Nome del programma: **GY521_2**

Porte:
A4 – SDA;
A5 - SCK (analogico)

Modulo principale:
Il modulo **GY521** è molto versatile: accelerometro/giroscopio a 3 assi; sensore di temperatura; oscillatore di clock interno; un processore di movimento digitale.

Porte:
D3/./D7 (digitali)

Comp. Accessori:
Dispaly TFT 1,77" 160(RGB)x128 su DY1

Monitor Seriale: sì (9600 baud)
Plotter seriale: no

Scopo del programma: Mostra sul monitor seriale l'accelerazione sui 3assi; la posizione girscopica, sempre sui tre assi, e anche la temperatura. Fornisce le stesse informazioni sul dispaly TFT.

Librerie richieste: Wire.h; Adafruit_GFX.h; Adafruit_ST7735.h; SPI.h; Adafruit_SPITFT.h

Note: nessuna

Link: nessuno

Nome del programma: **sismografo**

Porte:
A4 – SDA;
A5 - SCK (analogico)

Modulo principale:
Il modulo **GY521**

Monitor Seriale: no
Plotter seriale: si (9600 baud)

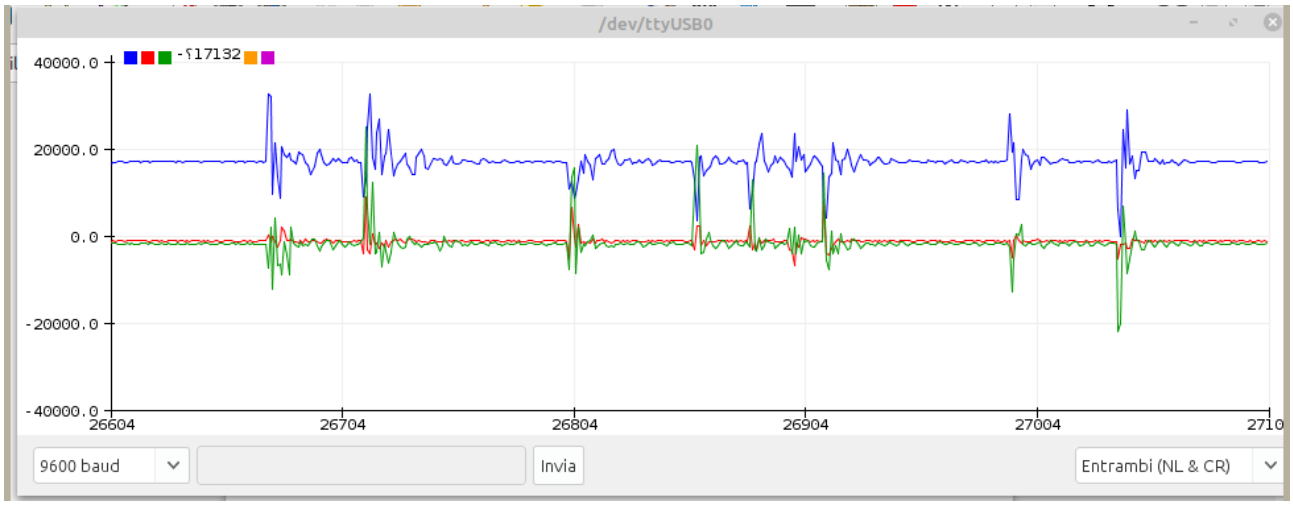
Scopo del programma: Il modulo **GY521** trasforma la bs in un sensibile sismografo

Librerie richieste: Wire.h;

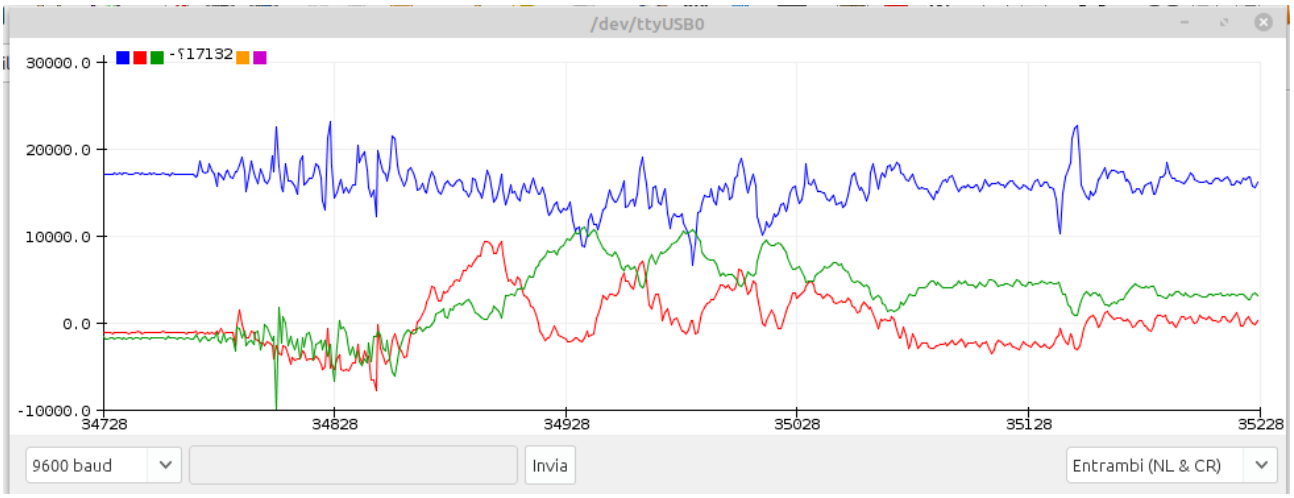
Note: nessuna

Link: https://www.youtube.com/watch?v=JUXODUxZ1PQ

Probabilmente questo “sismografo” non verrà utilizzato nei laboratori ufficiali, però è abbastanza sensibile e molto scenografico; il plotter seriale mostra tre linee separate, una per ogni asse, che si muovono indipendentemente.



Percuotendo il tavolo…



… e muovendo la bs sui tre assi.

Programmi per sensore Max30102 (pulsazioni/ossigenazione del sangue) - AN4

MAX30102 è un sensore con notevoli possibilità: permette di misurare la delle pulsazioni del cuore, la quantità di ossigeno (O_2) e la temperatura corporea. Il modulo funziona sia a 3,3 v quanto a 5v. Questo è un sensore per un apparecchio sperimentale, e non uno strumento elettro-medicale. Quindi per valutazione dello stato della propria salute, rivolgersi a un medico!

Nome del programma:	**HeartRate**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	Il modulo **Max30102** ha molte funzioni: misura le pulsazioni, la saturazione di ossigeno nel sangue; la temperatura
Monitor Seriale: sì (115200 baud)	**Plotter seriale: sì** (115200 baud). Leggi nota
Scopo del programma:	Mostra sul plotter o sul monitor seriale le pulsazioni del cuore.
Note:	La piedinatura è leggermente diversa sia per i connettori A3 che A4 (lavorano entrambi su A4 SDA; A5 SCK). Quindi è necessario usare un connettore e collegare con molta attenzione i cavi al connettore prescelto. Bisogna tenere qualche tempo il dito sul sensore prima che appaiano i dati, e il dito deve essere tenuto il più fermo possibile. Nel programma si consiglia di bloccarlo con un elastico. Bisogna commentare alcune righe del programma, nel caso che si preferisca usare il monitor o il plotter seriale. La seriale è settata a115200 baud. Se non si imposta la velocità corretta, si ottengono caratteri senza senso
Librerie necessarie	MAX30105.h; heartRate.h; Wire.h
Link:	https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/

Nome del programma:	**Temperature_Sense**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	Il modulo **Max30102** ha molte funzioni: misura le pulsazioni, la saturazione di ossigeno nel sangue; la temperatura
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì (9600 baud). Leggi nota**
Scopo del programma:	Mostra sul plotter o sul monitor seriale la temperatura corporea. Bisogna attendere 2/3 minuti per misurare la reale temperatura.
Librerie necessarie:	Wire.h; MAX30105.h
Note:	La piedinatura è leggermente diversa sia per i connettori A3 che A4 (lavorano entrambi su A4 SDA; A5 SCK). Quindi è necessario usare un connettore e collegare con molta attenzione i cavi al connettore prescelto. Bisogna tenere qualche tempo il dito sul sensore prima che appaiano i dati, e il dito deve essere tenuto il più fermo possibile. Nel programma si consiglia di bloccarlo con un elastico. Bisogna commentare alcune righe del programma, nel caso che si preferisca usare il monitor o il plotter seriale. La seriale è settata a115200 baud. Se non si imposta la velocità corretta, si ottengono caratteri senza senso.
Link:	https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/

Nome del programma:	**Oled_HeartRate**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	Il modulo **Max30102** ha molte funzioni: misura le pulsazioni, la saturazione di ossigeno nel sangue; la temperatura
A4 – SDA; A5 - SCK (analogico)	Dispaly Oled SSD1306 su DY3
D13 (digitale)	KY-006 (KY-012) Buzzer su BZ1
Monitor Seriale: sì (115200 baud)	**Plotter seriale:** sì (115200 baud). Leggi nota
Scopo del programma:	Mostra sul plotter o sul monitor seriale le pulsazioni del cuore. Le stesse informazioni sono mostrate in modo grafico sul piccolo dispaly Oled. Quando il sistema ottiene una misurazione, sul dispaly "batte" il cuore e si vedono i battiti. Viene emesso un "bip" molto simile a quello degli apparecchi medici. Un programma veramente carino.
Note:	La piedinatura è leggermente diversa sia per i connettori A3 che A4 (lavorano entrambi su A4 SDA; A5 SCK). Quindi è necessario usare un connettore e collegare con molta attenzione i cavi al connettore prescelto. Bisogna tenere qualche tempo il dito sul sensore prima che appaiano i dati, e il dito deve essere tenuto il più fermo possibile. Nel programma si consiglia di bloccarlo con un elastico. Bisogna commentare alcune righe del programma, nel caso che si preferisca usare il monitor o il plotter seriale. La seriale è settata a115200 baud. Se non si imposta la velocità corretta, si ottengono caratteri senza senso.
Librerie necessarie:	Adafruit_GFX.h; Adafruit_SSD1306.h; Adafruit_SSD1306.h; MAX30105.h; heartRate.h
Link:	https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/

Nome del programma:	**SPO2**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	Il modulo **Max30102** ha molte funzioni: misura le pulsazioni, la saturazione di ossigeno nel sangue; la temperatura
Monitor Seriale: sì (9600 baud)	**Plotter seriale: sì (9600 baud). Leggi nota**
Scopo del programma:	Mostra sul plotter o sul monitor seriale la saturazione di ossigeno nel sangue. Quando si è pronti è necessario premere un tasto.
Note:	La piedinatura è leggermente diversa sia per i connettori A3 che A4 (lavorano entrambi su A4 SDA; A5 SCK). Quindi è necessario usare un connettore e collegare con molta attenzione i cavi al connettore prescelto. Bisogna tenere qualche tempo il dito sul sensore prima che appaiano i dati, e il dito deve essere tenuto il più fermo possibile. Nel programma si consiglia di bloccarlo con un elastico. Bisogna commentare alcune righe del programma, nel caso che si preferisca usare il monitor o il plotter seriale. La seriale è settata a115200 baud. Se non si imposta la velocità corretta, si ottengono caratteri senza senso
Librerie necessarie:	Wire.h; MAX30105.h; spo2_algorithm.h
Note:	nessuna
Link:	https://lastminuteengineers.com/max30102-pulse-oximeter-heart-rate-sensor-arduino-tutorial/

Programmi per GY302 – light sensor

Il sensore di luce è un modulo semplice, usa le porte analogiche A4 e A5 e utilizza il protocollo I2C, per cui è possibile collegarlo insieme ad altri moduli che utilizzano lo stesso protocollo, come per esempio il display OLED 128 x 32 oppure 128 x 64.

Nome del programma:	**GY302_1**
Porte:	**Modulo principale:**
A4, A5 (analogica)	GY302
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Sul monitor seriale appaiono i valori della luce.
Note:	Nessuna.
Link:	https://www.adrirobot.it/bh1750-gy-302-sensore-di-luce/

Nome del programma:	**GY302_2**
Porte:	**Modulo principale:**
A4, A5 (analogica)	GY302
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Porte:	**Comp. Accessori:**
A4, A5 (analogica)	Display OLED 128 x 64 su DY3
Scopo del programma:	Sul monitor seriale appaiono i valori della luce, e la stessa informazione è anche presente sul display OLED 128 x 64
Librerie necessarie:	Wire.h; BH1750.h
Link:	https://www.adrirobot.it/bh1750-gy-302-sensore-di-luce/

Nome del programma:	**GY302_3**
Porte:	**Modulo principale:**
A4, A5 (analogica)	GY302
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Porte:	**Comp. Accessori:**
A4, A5 (analogica)	Display OLED 128 x 64 su DY3
D11 (digitale)	Led blu su LD1
D9 (digitale)	Led rosso su LD3
D9, D11 (digitale)	eventuale/i relay KY-019 in parallelo ai led
Scopo del programma:	Sul monitor seriale appaiono i valori della luce, e la stessa informazione è anche presente sul display OLED 128 x 64. Inserendo una soglia, i led segnalano quando si sono superate.
Note:	Wire.h; BH1750.h
Link:	https://www.adrirobot.it/bh1750-gy-302-sensore-di-luce/

Programmi per NEO 7M (GPS) - AN4

Neo 7M

Antenna per GPG

Il modulo Neo 7M è un ottimo GPS, il fratello maggiore del Neo 6MV2, di cui condivide i programmi, ovviamente cambiando le porte Rx/Tx per adattarle allo zoccolo AN4. Entrambi i moduli sono poco sensibili, e per riconoscere i satelliti devono essere messi vicino a una finestra o all'esterno.

Il Neo 7M ha anche un ingresso per un'antenna esterna, e questo migliora un poco la situazione, anche se non fa miracoli…

La prima volta che si collega il modulo alla bs, sembra che non funzioni (su Neo 7M il led rimane fisso durante la ricerca, lampeggia quando ha trovato un satellite, mentre sul Neo 6MV2 il led è spento durante la ricerca, lampeggia quando si è agganciato). E' necessario un po' di pazienza. Entrambi i moduli hanno una piccola pila di backup, per mantenere i dati salvati. Le volte successive, il satellite viene ritrovato con maggiore rapidità.

Purtroppo i due moduli hanno una diversa disposizione dei piedini, per cui il Neo 7M alloggia nativamente su AN4. Nulla toglie di collegarlo allo zoccolo DG6, come il suo fratello minore, ma in questo caso è necessario un connettore a quattro cavi, per rispettare la piedinatura.
Entrambi possono essere alimentati sia a +3,3v che a +5v.

Nome del programma:	**Neo7M_3**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**Neo-6MV2 – modulo GPS**
Monitor Seriale: si (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D3/./D7 (digitali)	TFT 1.77" ST7735
Scopo del programma:	Oltre alla la latitudine e la longitudine del luogo in cui si esegue la misurazione, mostra anche la data, l'ora, l'altezza, la velocità attuale e il satellite a cui si è agganciato sul display TFT. Ottima grafica
Librerie richieste:	SoftwareSerial.h; SPI.h; TinyGPS++.h; Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Eseguire il test vicino a una finestra, su di un balcone o all'aperto. Ci vuole qualche decina di secondi prima che agganci il satellite. In questo caso, il led sul modulo comincia a lampeggiare.
Link:	nessuno

Come modificare i programmi per lo zoccolo DG6:

Originale per DG6

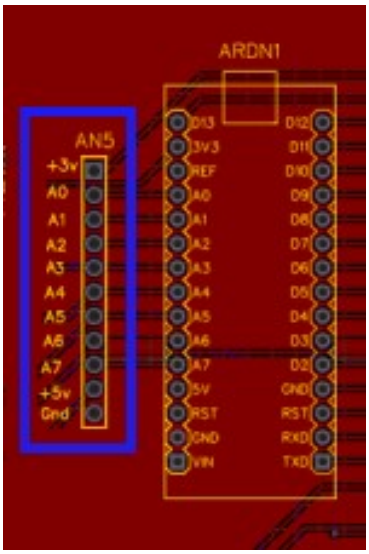
```
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS,
static const int RXPin = 9, TXPin = 10; //GPS
static const uint32_t GPSBaud = 9600;
```

Modificato per AN4

```
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS,
static const int RXPin = 19, TXPin = 18; //GPS
static const uint32_t GPSBaud = 9600;
```

Ricordo che "A4" presente sullo zoccolo AN4 trasformata in porta digitale diventa "18" e "A5" diventa "19". Si possono usare tutti i programmi per NEO6M su DG6, con queste modifiche.

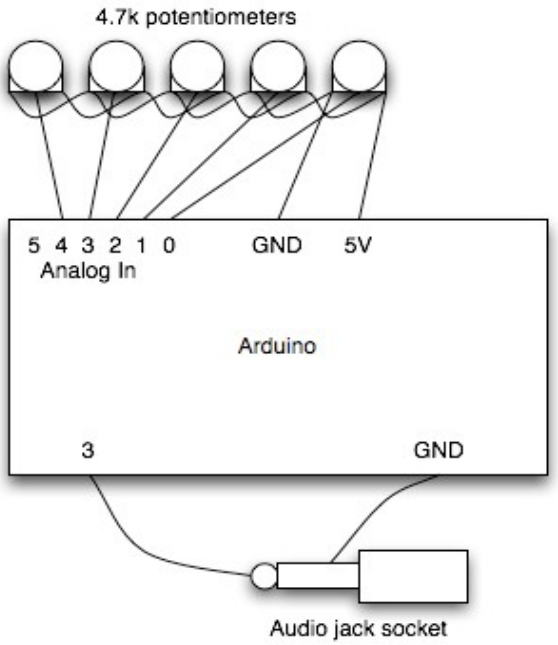
Lo zoccolo AN5



AN5 si trova alla sinistra di Arduino Nano.
È il gemello di D11: in sé raccoglie tutte le porte analogiche, oltre che le due alimentazioni.

A essa si possono collegare liberamente tutti i moduli che usano le porte analogiche, senza dimenticare che quelle comprese tra A0 e A5 possono essere facilmente trasformate in porte digitali.

AN5. AN5 è per i collegamenti analogici ciò che DG11 è per quelle digitali: uno zoccolo "universale", infatti su di esso sono presenti tutte le porte analogiche, da A0 ad A7, più le due alimentazione (+5v e +3,3v) e la massa. Perciò su questo connettore si possono collegare **tutti** i moduli analogici elencati per gli zoccoli da AN1 ad AN4 e volendo – trasformando le porte analogiche in digitali, come abbiamo visto relativamente a AD1 e AD2 – anche sensori misti o addirittura digitali, nel caso avessimo utilizzato tutte le porte digitali disponibili.
Come esempio, abbiamo utilizzato "*granular_synth*", che trasforma Arduino in un piccolo sintetizzatore analogico. Esso utilizza cinque potenziometri, collegati rispettivamente da A0 ad A4. In questo caso non ci sono zoccoli per cinque potenziometri; si potrebbe creare una struttura "volante, con tanti fili; ma seguendo la nostra filosofia, abbiamo creato una basetta apposta su cui alloggiare i potenziometri, che si può trovare nella sezione "**appendici**"



Questo è lo schema di collegamento dei potenziometri sul progetto originale del "granular synth".

Questo è il link su youtube:
https://www.youtube.com/watch?v=NTob27lOpcU&t=74s
Le informazioni sono un po' sintetiche. Nelle appendici si trova lo schema elettrico, leggermente modificato per adattarlo anche ad altri progetti.

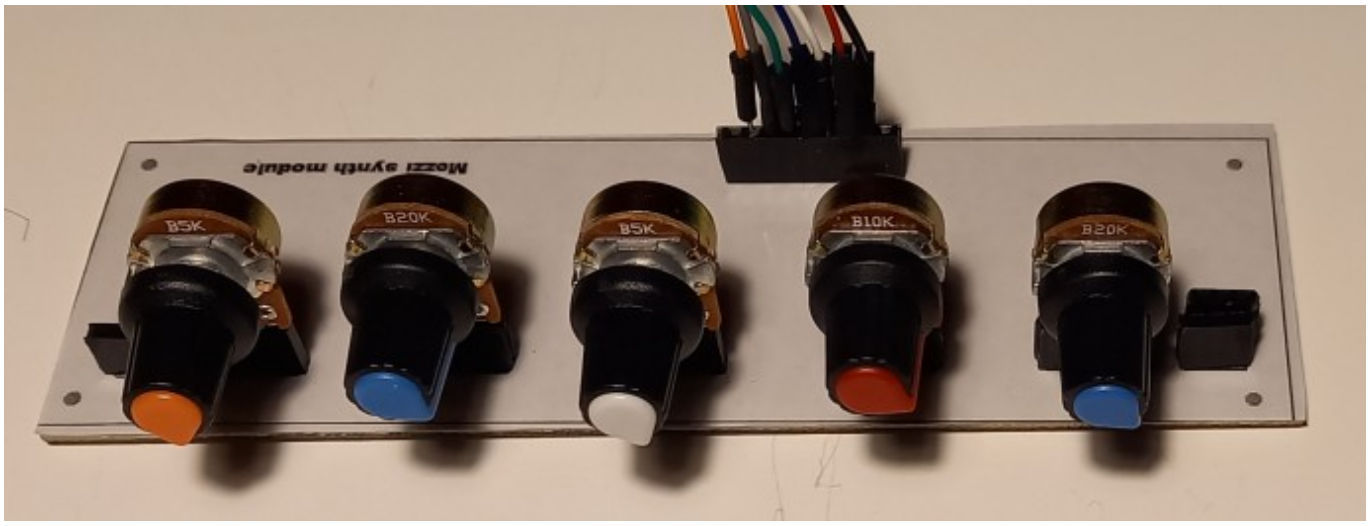
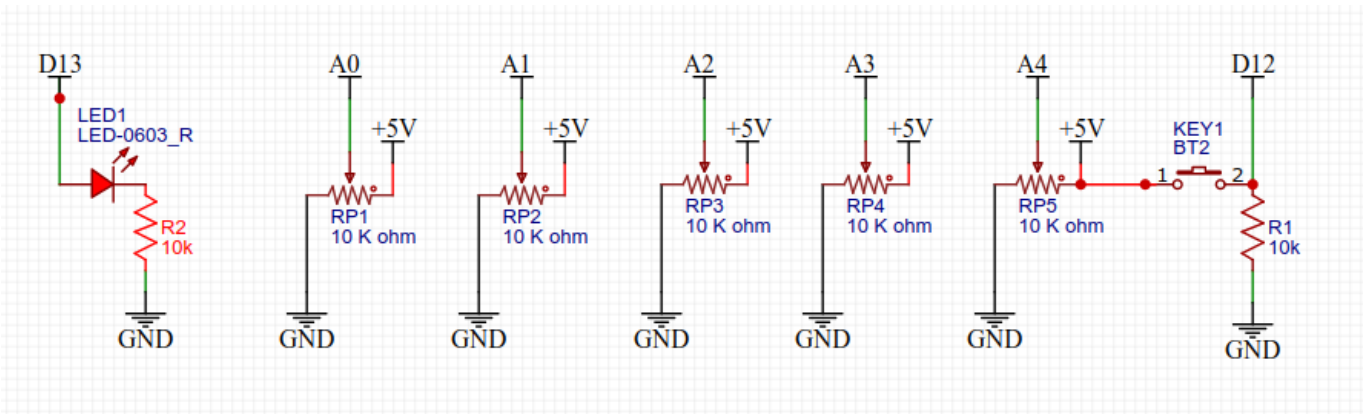


Foto della basetta autocostruita con i potenziometri per il granular_synth

Per il progetto, vai alle appendici

Il primo programma dà l'idea della versatilità di Arduino Nano: infatti lo trasforma in un mini-sintetizzatore!
Saranno necessari cinque potenziometri lineari, da 10 K , comunque il valore non è critico. Per tutti, vale lo stesso schema: il pin a sinistra va collegato a massa, quello di destra a +5 v, mentre i cinque centrali andranno collegati rispettivamente da A0 ad A4. Per non avere un numero eccessivo di cavi, è stata progettata e realizzata una basetta apposita (vedi foto). Ulteriori istruzioni particolareggiate saranno reperibili nella sezione "appendici" alla voce **"Granular synth"**.



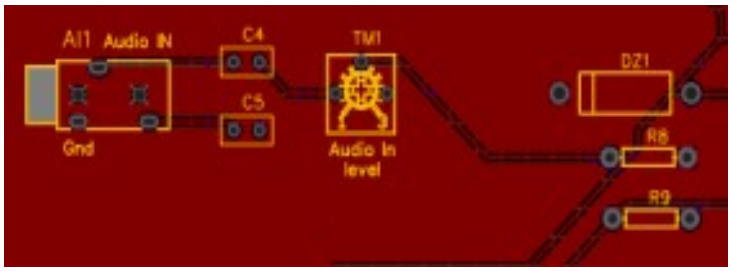
Schema dei potenziometri da collegare su AN5 per il programma "granular synth"

Programmi per sintetizzatore sonoro - AN5

Nome del programma:	**Granular_synth_1**
Porte:	**Comp. Aggiuntivi:**
A0 (analogico)	Potenziometro 10 K lineare – grain freq. control
A1 (analogico)	Potenziometro 10 K lineare – grain 2 decay control
A2 (analogico)	Potenziometro 10 K lineare – grain decay control
A3 (analogico)	Potenziometro 10 K lineare – grain 2 freq. control
A4 (analogico)	Potenziometro 10 K lineare – sync control
D3 (digitale)	Buzzer KY-006 (Ky012) oppure uscita audio
Scopo del programma:	Trasforma Arduino nano in un mini-sintetizzatore. Non richiede moduli aggiuntivi.
Note:	E' stata disegnata e costruita una basetta per accogliere i cinque potenziometri. ***Usare la porta D3, spostando il piccolo selettore denominato "buzzer" su D3***, altrimenti non percepirete alcun suono!
Link:	https://www.youtube.com/watch?v=NTob27lOpcU&list=RDCMUC97oxpI-kInN6trsvs6yLNg&index=1

Nome del programma:	**Granular_synth_2**
Porte:	**Comp. Aggiuntivi:**
A0 (analogico)	Potenziometro 10 K lineare – grain freq. control
A1 (analogico)	Potenziometro 10 K lineare – grain 2 decay control
A2 (analogico)	Potenziometro 10 K lineare – grain decay control
A3 (analogico)	Potenziometro 10 K lineare – grain 2 freq. control
A4 (analogico)	Potenziometro 10 K lineare – sync control
D12	Pulsante KY-004 su DG4
D3 (digitale)	Buzzer KY-006 (Ky012) oppure uscita audio
Scopo del programma:	Trasforma Arduino nano in un mini-sintetizzatore. Non richiede moduli aggiuntivi. L'aggiunta di un pulsante serve a cambiare il metodo di emissione delle note: in modo continuo, con scala coromatica o pentatonica.
Note:	E' stata disegnata e costruita una basetta per accogliere i cinque potenziometri. ***Usare la porta D3, spostando il piccolo selettore denominato "buzzer" su D3***, altrimenti non percepirete alcun suono!
Link:	https://www.youtube.com/watch?v=NTob27lOpcU&list=RDCMUC97oxpI-kInN6trsvs6yLNg&index=1

L’ingresso AI1 (Audio in – analogico)



La porta di ingresso audio AI1 si trova in basso a sinistra. E’ connessa alla porte analogica A6 e va in conflitto con eventuali moduli presenti sullo zoccolo AD2.

Richiede un segnale in ingresso di una certa ampiezza, ma che comunque non deve superare i 5 volt, per non danneggiare la porta di Arduino.

Audio in (AI1). Per la *bs* è stato integrato anche un ingresso audio, per processare i segnali provenienti da amplificatori, computer, ecc. Il livello del segnale è importante, perché deve essere compreso tra 0 e 5 volt. Perché sia rilevabile, è necessario che sia abbastanza elevato, e il segnale che si preleva dall’uscita della cuffia, non sempre è sufficiente per provocare delle variazioni significative nel monitor seriale, o su di un display; tuttavia non deve superare i 5 v, per non danneggiare l’ingresso analogico di Arduino. I condensatori C4 e C5 servono per disaccoppiare il segnale in ingresso. Il trimmer TM1 regolare l’ampiezza del segnale in ingresso. Sempre per evitare di superare i 5 v, è stato inserito un partitore resistivo, per mezzo delle resistenze R8 ed R9 e allo stesso scopo è presente il diodo zener DZ1, dal valore di 4,7 volt. Alcuni programmi consentono di avere una visione semplificata del segnale in ingresso, altri di costruire un analizzatore di spettro a barre del segnale audio. Niente male per un microcontroller così piccolo!

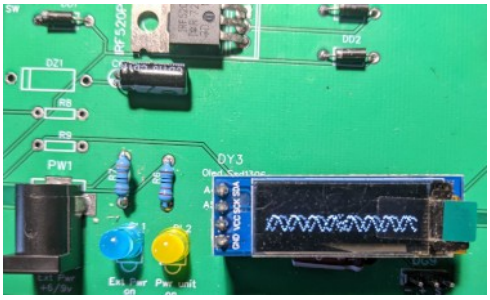
I programmi per l’ingresso audio AI1

Qui di seguito si trovano un paio di programmi per analizzare il suono in ingresso. Il primo mostra una linea che varia rozzamente in base alla forma d’onda del segnale di ingresso. Non ha alcuna pretesa né qualitativa né quantitativa, però è bella da vedere.

Il secondo è più raffinato. Richiede una libreria per le curve audio studiate da Fourier. Il display si trasforma in un piccolo analizzatore di spettro a 32 bande luminose.
I programmi sono stati duplicati, per poter essere usati sia con il display 128x32 che 128x64.
L’ingresso va sulla porta A6 di Arduino.

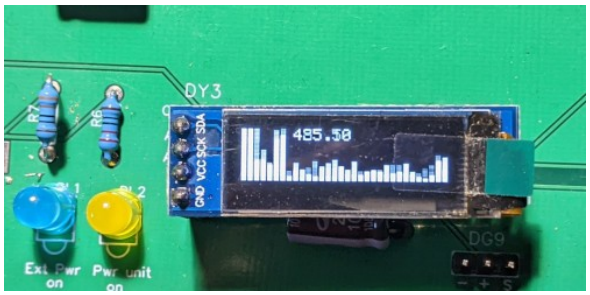
Programmi per SSD1306 128x32 - AI1

Nome del programma:	**Wave (128 x 32)**
Porte:	**Modulo principale:**
A6 (analogico)	Ingresso audio (audio in)
Porte:	**Comp. Accessori:**
A4 SDA (analogico)	**SSD1306** -Dispaly Oled 128x32 pixel
A5 SCK (analogico)	
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sullo schermo appaiono le variazioni di frequenza in sincrono con la musica, oppure le frequenze sonore ottenute con un generatore. Naturalmente non è un oscilloscopio… l'immagine è abbastanza grezza, ma piacevole.
Librerie necessarie:	Adafruit_GFX; Adafruit_SSD1306; Wire
Note:	Naturalmente le visualizzazioni sono semplicemente qualitative. Eseguendo alcune modifiche sul programma è possibile usare un dispay OLED 128x64, dando un immagine un po' più ampia. Può mostrare frequenze fino a qualche chiloHertz. Richiede un segnale abbastanza alto, ma comunque inferiore a 5 V, per non danneggiare l'ingresso di Arduino.
Link:	https://www.youtube.com/watch?v=TFatB-5mWlo



Una onda sinusoidale ottenuta con un generatore di segnale. Naturalmente la visualizzazione dà un'idea approssimativa del segnale, e non ha alcun valore qualitativo né quantitativo.

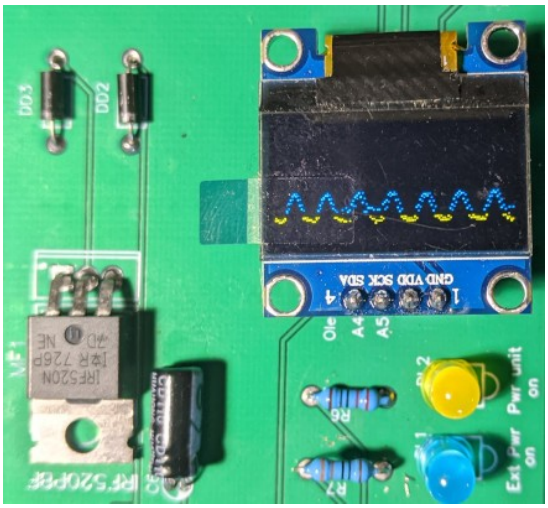
Nome del programma:	**Spectrum (128 x 32)**
Porte:	**Modulo principale:**
A6 (analogico)	Ingresso audio (audio in)
Porte:	**Comp. Accessori:**
A4 SDA (analogico)	**SSD1306** -Dispaly Oled 128x32 pixel
A5 SCK (analogico)	
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sullo schermo appare un analizzatore di spettro a 32 barre, che salgono e scendono in base alla frequenza analizzata dai 32 filtri. Naturalmente anche questo programma fornisce delle informazioni semplicemente qualitative; non è uno strumento professionale, però è piacevole da osservare
Librerie necessarie:	Adafruit_GFX.h; Adafruit_SSD1306.h; Wire.h; arduinoFFT.h
Note:	Può mostrare frequenze fino a qualche KHz. Richiede un segnale abbastanza alto, ma comunque inferiore a 5 V, per non danneggiare l'ingresso di Arduino.
Link:	https://www.youtube.com/watch?v=TFatB-5mWlo



In questa immagine si osserva un'immagine spettrale del suono a 32 barre.
Il grafico che si ottiene è abbastanza realistico, tuttavia è necessario un segnale audio abbastanza alto affinché le barre si muovano in maniera facilmente leggibile.

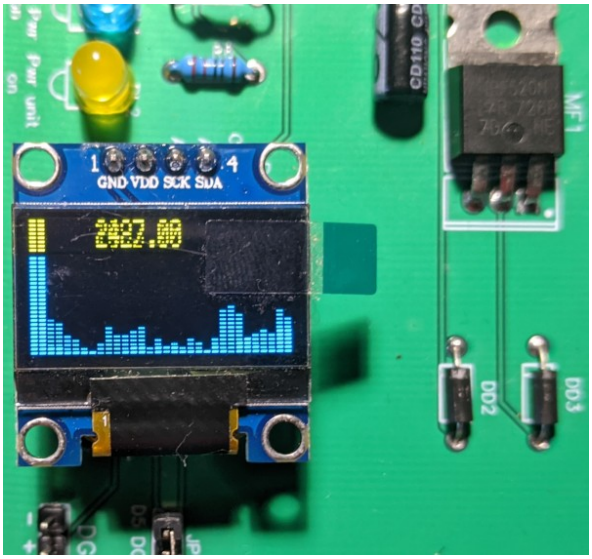
Programmi per SSD1306 128x64 - AI1

Nome del programma:	**Wave (128 x 64)**
Porte:	**Modulo principale:**
A6 (analogico)	Ingresso audio (audio in)
Porte:	**Comp. Accessori:**
A4 SDA A5 SCK (analogico)	**SSD1306** -Dispaly Oled 128x64 pixel
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sullo schermo appaiono le variazioni di frequenza in sincrono con la musica, oppure le frequenze sonore ottenute con un generatore. Naturalmente non è un oscilloscopio… l'immagine è abbastanza grezza, ma piacevole.
Librerie necessarie:	Adafruit_GFX.h; Adafruit_SSD1306.h; Wire.h
Note:	Naturalmente le visualizzazioni sono semplicemente qualitative. Eseguendo alcune modifiche sul programma è possibile usare un dispay OLED 128x64, dando un immagine un po' più ampia. Può mostrare frequenze fino a qualche KHz. Richiede un segnale abbastanza alto, ma comunque inferiore a 5 V, per non danneggiare l'ingresso di Arduino.
Link:	https://www.youtube.com/watch?v=TFatB-5mWlo



L'immagine di un onda sinusoidale ottenuta collegando all'ingresso audio un generatore di segnale.

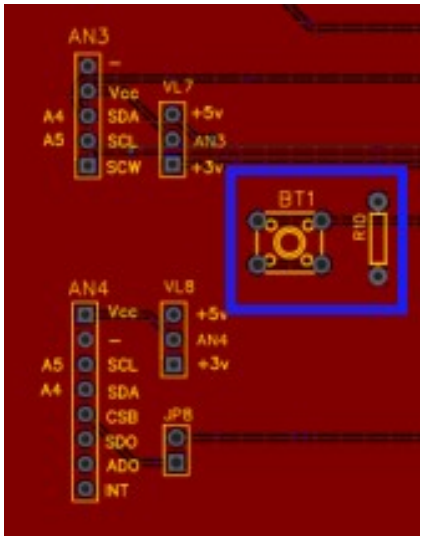
Nome del programma:	**Spectrum (128 x 64)**
Porte:	**Modulo principale:**
A4 SDA	**SSD1306** -Dispaly Oled 128x64 pixel
A5 SCK (analogico)	
Porte:	**Comp. Accessori:**
A6 (analogico)	Ingresso audio (audio in)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sullo schermo appare un analizzatore di spettro a 32 barre, che salgono e scendono in base alla frequenza analizzata dai 32 filtri. Naturalmente anche questo programma fornisce delle informazioni semplicemente qualitative, non è uno strumento professionale, però è piacevole da osservare
Librerie necessarie:	Adafruit_GFX.h; Adafruit_SSD1306.h; Wire.h; arduinoFFT.h
Note:	Può mostrare frequenze fino a qualche chiloHertz. Richiede un segnale abbastanza alto, ma comunque inferiore a 5 V, per non danneggiare l'ingresso di Arduino.
Link:	https://www.youtube.com/watch?v=TFatB-5mWlo



L'analisi dei livelli di frequenza a 32 barre ottenuta collegando un lettore CD all'ingresso audio della *bs*. Il livello di emissione della fonte audio deve essere sufficientemente elevata per permettere una visualizzazione chiaramente leggibile.

Attenzione comunque a non superare i 5V del segnale, per non danneggiare Arduino!

Il pulsante BT1



Il pulsante BT1 è stato inserito nativamente nella *bs*, in posizione in alto a sinistra, perché spesso serve un pulsante di test nei programmi, per accendere un led, attivare un relay, far suonare un cicalino, effettuare il reset di un allarme…

Usa la porta analogica A2, che può essere trasformata facilmente in digitale, diventando D16.

BT1. Questo pulsante è stato inserito nella *bs* per eseguire dei test, per esempio per accendere un led, eccitare un relay, produrre un suono con un cicalino, o anche per incrementare o decrementare un contatore, utile per esempio per aumentare o diminuire una soglia di temperatura, come si può vedere nei programmi ad esso dedicati.

E' stato inserito un partitore resistivo R10 tra la porta di Arduino e la massa (ground).
La porta utilizzata è A2, ma nei programmi, se necessario, può essere sostituita da D16.

Configurazione di BT1 (montato sulla basetta):

Socket BT1 (analogic)			1	2	3
Description	Code	Note	-	+ (+5v)	S
Button	on board	+ res. 10 KΩ	-	+	A2

Programmi per pulsante BT1

Nome del programma: BT1_1
Porte: **Modulo principale:**
A2 (analogica) BT1
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Porte: **Comp. Accessori:**
D13 Buzzer KY-006 (KY-012)
Scopo del programma: Sul monitor digitale appaiono i valori del pulsante (0 quando è aperto; abitualmente sopra il 1000 quando premuto. Quando si preme il pulsante, il buzzer emette un suono.
Note: Nessuna.
Link: nessuno

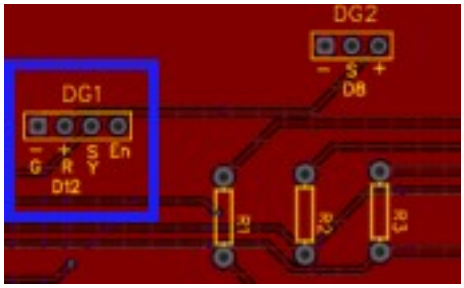
Nome del programma:	**BT1_2**
Porte:	**Modulo principale:**
D16 (digitale) porta fisica: A2	BT1
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D2/./D7 (digitali)	Display LCD 1602 su DY2 (16 caratteri x 2 linee)
D10 (digitale)	Led (abitualmente blu) su LD2
D11 (digitale)	Led (abitualmente verde) su LD1
Scopo del programma:	Il valore iniziale è zero, il led verde è acceso. Se si preme il pulsante, i valori incrementano fino a 20 e il led blu si accende. Se si preme ancora, il valore torna a "0" e il ciclo riprende. Il monitor seriale riporta i valori. Questo programma potrebbe essere la base per impostare una temperatura limite.
Note:	Si trasforma la porta analogica A2 in digitale D16. **Vedi paragrafo con le spiegazioni.**
Link:	nessuno

Nome del programma:	**BT1_3**
Porte:	**Modulo principale:**
D16 (digitale) porta fisica: A2	BT1
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D2/./D7 (digitali)	Display LCD 1602 su DY2 (16 caratteri x 2 linee)
D10 (digitale)	Led (abitualmente blu) su LD2
D9 (digitale)	Led (abitualmente rosso) su LD3
D9 (digitale)	Relay KY-019 su RL2
D12 (digitale)	Pulsante KY-004 su DG1
D11 (digitale)	KY-001 sens. temperatura 18B20 su RL1
Scopo del programma:	In questo programma un sensore di temperatura misura la temperatura ambiente. All'accensione la temperatura di soglia è di 25 gradi. Se la temperatura è superiore a quella di soglia, si accende il led rosso; altrimenti quello blu. Con i pulsanti si può variare la temperatura di soglia, tra 15 e 35 gradi. Un relay collegato al led rosso (LD3) può pilotare, per esempio, un condizionatore o una caldaia.
Note:	Si trasforma la porta analogica A2 in digitale D16. **Vedi paragrafo con le spiegazioni.**
Link:	nessuno

Elenco degli zoccoli collegati alle porte digitali.

Sulla bs ci sono ben tredici zoccoli per gestire le porte digitali, perché molti moduli usano le porte digitali, in cui i valori possono essere solo “1” (+5 volt) oppure “0” (nessuna tensione)

Lo zoccolo DG1



Guardando la *bs* dall’alto, lo zoccolo DG1 (riquadro blu), si trova in alto e alla destra di Arduino Nano.

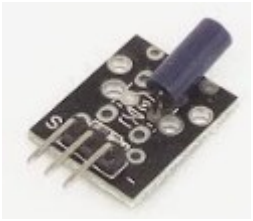
La porta utilizzata è D12

DG1. Come tutti gli zoccoli che iniziano per DG, DG1 è collegato alle porte digitali; come già accennato nel riquadro, si connette alla porta D12 di Arduino. Perciò tutti i programmi che fanno capo ad esso, si riferiranno a questa porta. La maggior parte dei moduli che si collegano su questo zoccolo, hanno solo tre connessioni; l’unico che ne ha quattro (di cui l’ultimo non è comunque gestito) è KY-032, “Avoiance”. Il connettore connesso a “EN” non viene utilizzato.

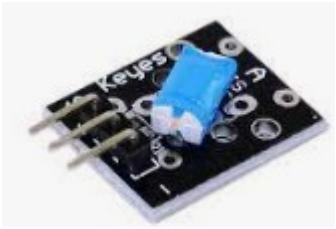
Ecco la lista dei numerosi moduli che si possono collegare direttamente a questo zoccolo:

Socket DG1 (digital, D12)			1	2	3	4
Description	Code	Note	-	VCC (+5v)	S	EN
Tap Module	KY-031 – HW500					
Shock sensor	KY-002 – HW513					
Tilt switch	KY-020 – HW501					
IR emission	KY-005 - HW489					
Relay	KY-019 - HW482					
Temperature Humidity	KY-015 – DHT11 – HW507					
Button	KY-004 - HW-483					
Laser emit	KY-008 - HW493					
433 MHz transmitter (TX)			-	+	D12	/
Tracking	KY-033 - HW511	piedinatura ruotata				
Digital Temperature	KY-001 – 18B20 - HW506	piedinatura ruotata				
Photo interrupt	KY-010 – HW487	piedinatura ruotata				
IR Receiver	KY-022 - HW490	piedinatura ruotata				
Mercury tilt module	KY-017 -HW505	piedinatura ruotata				
Hall magnetic sensor	KY-035 – HW492	piedinatura ruotata				
Reed switch sensor	KY-021 – HW497	piedinatura ruotata				
Avoiance	KY-032 - HW201	piedinatura ruotata				

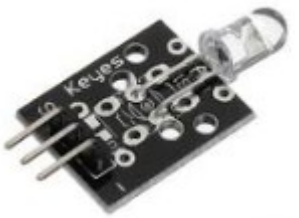
KY-031 Tap module



KY-002 shock sensor

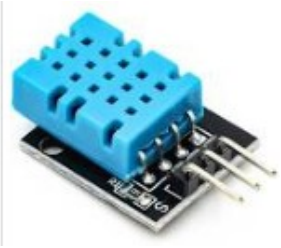


KY-020 tilt switch



KY-005 IR trasmission

KY-019 relay



KY-015 (DHT11) temperature & humidity

KY-004 button

KY-008 laser emit

433 MHz transmitter (TX)

Questi moduli, guardandoli frontalmente, hanno i piedini in questa sequenza, da sinistra a destra:

“S”, “+”, “-”.

Quelli seguenti, che verranno mostrati nei prossimi riquadri, hanno i piedini in sequenza inversa, ovvero **“-”, “+”, “S”**. Pertanto vanno inseriti sullo zoccolo ruotati di 180°.

Nota: comunque controllare con attenzione, la piedinatura può essere diversa in base ai vari costruttori. Controllare attentamente, per evitare di danneggiare il modulo!

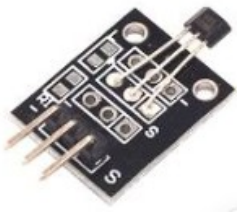
KY-033 - tracking

KY-0011 DS18B20 digital temperature

KY-010 photo interrupt

KY-022 IR receiver

KY-017 mercury tilt sens.



KY-035 hall sensor

KY-021 reed sw. sensor

KY-033 Obstacle avoiance

Programmi per KY-032 Obstacle Avoiance module - DG1

"Obstacle avoiance" serve per evitare ostacoli, particolarmente utile per robot o moduli semoventi. La sua portata è piuttosto ridotta, nell'ordine di qualche centimetro. E' stato usato anche per verificare la chiusura di una finestra, quindi come antifurto.
Ha due trimmer: quello che si vede in alto, regola la sensibilità, mentre quello inferiore vicino al connettore EN) regola la frequenza del segnale, che deve essere di 38 Khz. **Non modificare l'impostazione di fabbrica!**

Nome del programma:	**Avoiance_1**
Porte:	**Modulo principale:**
D13 (digitale)	**Avoiance module - KY-032**
Monitor Seriale: sì	**Plotter seriale: no**
(9600) baud)	
Porte:	**Comp. Accessori:**
D9 (digitale)	**Led L3 - rosso**
Scopo del programma:	Il modulo Avoiance permette di evitare gli ostacoli. Se si pone un ostacolo davanti al sensore, si accende il led e sul monitor seriale appare la segnalazione .
Note:	nessuna
Link:	nessuno

Nome del programma:	**Avoiance_2**
Porte:	**Modulo principale:**
D12 (digitale)	Avoiance module - KY-032
Monitor Seriale: sì (9600) baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 - rosso
D2/./D7 (digitali)	Dispaly LCD 1602 (16 caratteri x 2 linee)
D13 (digitale)	Buzzer KY-006 (o KY-012)
Scopo del programma:	Il modulo Avoiance permette di evitare gli ostacoli. Se si pone un ostacolo davanti al sensore, si accende il led e sul monitor seriale appare la segnalazione . Sul display a cristalli liquidi appare la segnalazione di presenza/assenza di ostacoli. Quando c'è un ostacolo, il buzzer emette una serie di note.
Note:	nessuna
Link:	nessuno

Programmi per KY-004 Button - DG1

Il pulsante ha una funzione molto semplice: può essere solo o premuto (0) o non premuto (1). Quindi è utile in tutti i casi in cui un evento accade solo se il pulsante viene premuto (o non premuto), in base al programma.

Nome del programma:	**Button_1**
Porte:	**Modulo principale:**
D12 (digitale)	Button KY-004
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 - rosso
D11 (digitale)	Led L1 - verde
Scopo del programma:	Se il pulsante non è premuto, il led verde è acceso, mentre quello rosso è spento. Quando si preme il pulsante, lo stato dei led si inverte. Lo stato del pulsante appare sul monitor seriale.
Note:	nessuna
Link:	nessuno

Nome del programma:	**Button_2**
Porte:	**Modulo principale:**
D12 (digitale)	Button KY-004
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 - rosso
D11 (digitale)	Led L1 - verde
D13 (digitale)	Buzzer KY-006 (o KY-012)
Scopo del programma:	Se il pulsante non è premuto (stato High), il led verde è acceso, mentre quello rosso è spento. Quando si preme il pulsante (stato Low), lo stato dei led si inverte. Lo stato del pulsante appare sul monitor seriale. Quando si preme il pulsante, il buzzer emette una serie di note.
Note:	nessuna
Link:	nessuno

Nome del programma:	**Button_3**
Porte:	**Modulo principale:**
D12 (digitale)	Button KY-004
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 – rosso su LD3
D10 (digitale)	Led L2 – blu su LD2
D11 (digitale)	Led L1 – verde su LD1
Scopo del programma:	Se si preme il pulsante, ciclicamente si accende un led diverso, in un ciclo di tre casi, poi si ripete.
Note:	nessuna
Link:	nessuno

Programmi per DHT11 Temperature & humidity sensor - DG1

DHT11 misura la temperatura e l'umidità dell'ambiente in tempo reale. Si usa per monitorare questi parametri, utile per esempio per accendere una caldaia o un condizionatore, ecc.

Nome del programma:	**DHT_1**
Porte:	**Modulo principale:**
D12 (digitale)	DHT-11 -KY-015. Questo sensore rileva temperatura e umidità
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sul monitor seriale appare la temperatura (gradi Celsius) e umidità in percentuale.
Librerie necessarie:	EEPROM.h; dht_nonblocking.h
Note:	nessuna
Link:	nessuno

Nome del programma:	**DHT_2**
Porte:	**Modulo principale:**
D12 (digitale)	DHT-11 -KY-015. Questo sensore rileva temperatura e umidità
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led L3 - blu
D10 (digitale)	Led L2 - verde
D11 (digitale)	Led L1 - rosso
D9 (digitale)	Relay KY-019 su porta RL2 (opzionale)
D11 (digitale)	Relay KY-019 su porta RL1 (opzionale)
Scopo del programma:	Sul monitor seriale appare la temperatura (gradi Celsius) e umidità in percentuale. Con due variabili, si può stabilire un intervallo tra una temperatura minima e una massima. Se la temperatura è in questo intervallo, si accende il led verde, e non succede nulla. Se la temperatura è inferiore alla minima, si accende il led blu (freddo) e si attiva il relay RL2, che può accendere il riscaldamento. Se la temperatura è superiore alla massima, si accende il led rosso (caldo) e si attiva il relay RL1, che può accendere il condizionatore.
Librerie necessarie:	EEPROM.h; dht_nonblocking.h
Note:	Per poter abbinare i led ai relay, è stato necessario cambiare la normale sequenza dei led. Per cui se si usasse il led a tre colori sul connettore L4, i colori non corrisponderebbero al reale stato del sistema. Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almenno 500 mA
Link:	nessuno

Nome del programma:	**DHT_3**
Porte:	**Modulo principale:**
D12 (digitale)	DHT-11 -KY-015. Questo sensore rileva temperatura e umidità
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D2/./D7	Dispaly LCD 1602 (16 caratteri x 2 linee)
D9 (digitale)	Led L3 - blu
D10 (digitale)	Led L2 - verde
D11 (digitale)	Led L1 - rosso
D9 (digitale)	Relay KY-019 su porta RL2 (opzionale)

D11 (digitale)	Relay KY-019 su porta RL1 (opzionale)
Scopo del programma:	Sul monitor seriale appare la temperatura (gradi Celsius) e umidità in percentuale. Con due variabili, si può stabilire un intervallo tra una temperatura minima e una massima. Se la temperatura è in questo intervallo, si accende il led verde, e non succede nulla. Se la temperatura è inferiore alla minima, si accende il led blu (freddo) e si attiva il relay RL2, che può accendere il riscaldamento. Se la temperatura è superiore alla massima, si accende il led rosso (caldo) e si attiva il relay RL1, che può accendere il condizionatore. La temperatura e l'umidità appare anche sul display a cristalli liquidi.
Librerie necessarie:	EEPROM.h; dht_nonblocking.h; LiquidCrystal.h
Note:	Per poter abbinare i led ai relay, è stato necessario cambiare la normale sequenza dei led. Per cui se si usasse il led a tre colori sul connettore L4, i colori non corrisponderebbero al reale stato del sistema. Nel caso si usi il relay, è necessario collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA
Link:	nessuno

Nome del programma:	**DHT_4**
Porte:	**Modulo principale:**
D12 (digitale)	DHT-11 -KY-015. Questo sensore rileva temperatura e umidità
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D3/./D7	Dispaly TFT 1,77" - ST7735
D9 (digitale)	Led L3 - blu
D10 (digitale)	Led L2 - verde
D11 (digitale)	Led L1 - rosso
D9 (digitale)	Relay KY-019 su porta RL2 (opzionale)
D11 (digitale)	Relay KY-019 su porta RL1 (opzionale)
Scopo del programma:	Sul monitor seriale appare la temperatura (gradi Celsius) e umidità in percentuale. Con due variabili, si può stabilire un intervallo tra una temperatura minima e una massima. Se la temperatura è in questo intervallo, si accende il led verde, e non succede nulla. Se la temperatura è inferiore alla minima, si accende il led blu (freddo) e si attiva il relay RL2, che può accendere il riscaldamento. Se la temperatura è superiore alla massima, si accende il led rosso (caldo) e si attiva il relay RL1, che può accendere il condizionatore. Il display TFT permette di vedere oltre alla temperatura e all'umidità, anche le attività richieste.
Librerie necessarie:	EEPROM.h; dht_nonblocking.h; Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Per poter abbinare i led ai relay, è stato necessario cambiare la normale sequenza dei led. Per cui se si usasse il led a tre colori sul connettore L4, i colori non corrisponderebbero al reale stato del sistema. Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA
Link:	nessuno

Programmi per DS18B20 Digital temperature sensor - DG1

Il sensore DS18B20 è molto preciso nel rilevare la temperatura, ed è anche programmabile. Un'altra sua particolarità è di avere un indirizzo al suo interno, così sarà possibile, connettere due sensori sulla stessa porta. Il primo programma serve a mostrare l'indirizzo interno del sensore.

Nome del programma:	**DS18B20_1**
Porte:	**Modulo principale:**
D12 (digitale)	DS18B20 – KY-001 Sensore di temperatura
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Dopo aver verificato la presenza del sensore, mostra a distanza di un secondo tra una misurazione e l'altra la temperatura sia in gradi Celsius che Kevin.
Note:	nessuna
Link:	nessuno

Programmi per KY-008   Laser led module

Il modulo led laser, emette una luce che si dice "coerente", ovvero di una sola lunghezza d'onda. Il fascio di luce del nostro laser è di colore rosso.

Attenzione! NON fissare MAI direttamente il fascio di luce prodotto dal diodo laser! Può danneggiare irrimediabilmente la retina!

Nome del programma:	**LASER_1**
Porte:	**Modulo principale:**
D12 (digitale)	Diodo Laser - KY008
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
A2 (analogica)	Pulsante su “BT1”
Scopo del programma:	Premendo il pulsante “BT1”presente sulla scheda sperimentale, si accende il diodo laser.
Note:	Attenzione! NON fissare MAI direttamente il fascio di luce prodotto dal diodo laser! Può danneggiare irrimediabilmente la retina!
Link:	nessuno

Programmi per KY-022 Infrared receiver module - DG1

Il sensore (ricevitore) a infrarosso può dare molte soddisfazioni, perché se abbinato a un qualsiasi vecchio telecomando, trasforma Arduino in un perfetto maggiordomo domotico: può accendere e spegnere luci, elettrodomestici, aprire o bloccare porte. Il limite è solo la nostra fantasia.
L’unico neo è che ci vuole un po’ di pazienza per configurare il proprio telecomando con Arduino: si è notato che diversi telecomandi generano stringhe di numeri diverse abbinate ai vari tasti.
Configurato questo aspetto, i programmi funzionano a meraviglia.

Nome del programma:	**Infrared_1**
Porte:	**Modulo principale:**
D12 (digitale)	Infrared receiver - KY022
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Questo programma permette di scoprire i codici che corrispondono ai tasti del proprio telecomando, che appaiono sul monitor seriale. Purtroppo ogni telecomando invia i propri codici, quindi è necessario aver pazienza, segnarseli e inserirli in uno dei prossimi programmi.
Librerie necessarie:	IRremote.h
Note:	nessuna
Link:	Nessuno

Nome del programma:	**Infrared_1a**
Porte:	**Modulo principale:**
D12 (digitale)	Infrared receiver - KY022
Porte:	**Comp. Accessori:**
D52/./D7 (digitale)	Dispaly LCD 602 su DY2
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Questo programma permette di scoprire i codici che corrispondono ai tasti del proprio telecomando, che appaiono sul monitor seriale. Purtroppo ogni telecomando invia i propri codici, quindi è necessario aver pazienza, segnarseli e inserirli in uno dei prossimi programmi. Essi possono essere visualizzati anche sul display LCD.
Librerie necessarie:	Irremote.h, LiquidCrystal.h
Note:	nessuna
Link:	Nessuno

Nome del programma:	**Infrared_2**
Porte:	**Modulo principale:**
D12 (digitale)	Infrared receiver - KY022
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Questo è un programma semplicemente dimostrativo: sono stati pazientemente inseriti i codici relativo a un vecchio telecomando Audiola, e sarebbe un caso veramente fortunato se si adattassero perfettamente al vostro; in questo modo, sul monitor seriale appaiono decodificati i codici corrispondenti ai vari tasti, invece che una lunga serie di numeri. Questo passo è utile per poter utilizzare il proprio telecomando per ottenere risultati più interessanti, come accendere o spegnere la luce in casa, avviare qualche elettrodomestico, ecc.
Librerie necessarie:	IRremote.h
Note:	Personalizzare i codici in base al proprio telecomando. Vedi immagine successiva.
Link:	nessuno

```
switch (IrReceiver.decodedIRData.command)

{
  case 11: videoir(); Serial.println(" Tasto: Power");
  case 93: videoir(); Serial.println(" Tasto: Mute");
  case 94: videoir(); Serial.println(" Tasto: Recall")
  case 3: videoir(); Serial.println(" Tasto: Menu");
  case 27: videoir(); Serial.println(" Tasto: CH +");
  case 31: videoir(); Serial.println(" Tasto: Exit");
  case 2: videoir(); Serial.println(" Tasto: Vol -");
  case 26: videoir(); Serial.println(" Tasto: OK");
  case 30: videoir(); Serial.println(" Tasto: Vol +");
  case 1: videoir(); Serial.println(" Tasto: Pg Up");
  case 25: videoir(); Serial.println(" Tasto: CH -");
  case 29: videoir(); Serial.println(" Tasto: Pg Dn");
  case 92: videoir(); Serial.println(" Tasto: 0");
```

In questa immagine si vedono i codici che sono stati desunti con il programma “Infrared_1”, dal nostro telecomando, per esempio “11” corrisponde al tasto “Power”. Verificare i codici emessi dal proprio telecomando e visualizzati con il programma “Infrared_1”. Sostituire i valori trovati nel programma “Infrared_2” o “Infrared_3” e inserirli nella funzione “case” e se necessario.

modificare anche il nome del tasto, inserendolo dopo ***Serial.println***. Attenzione a inserire le parenti e i doppi apici. Nel programma Infrared_2a, che utilizza anche il dispaly LCD 1602, correggere il dato corrispondente a ***lcd.print.***

Nel caso ci fossero delle righe inutilizzate, perché il vostro telecomando ha meno funzioni, cancellare quelli eccedenti; viceversa aggiungerli nel caso ne abbia di più.

Nel programma Infrared_3, ***che va comunque personalizzato in base al vostro telecomando***, vedremo di inserire qualche funzione maggiormente utile.

Nome del programma:	**Infrared_2a**
Porte:	**Modulo principale:**
D12 (digitale)	Infrared receiver - KY022
Porte:	**Comp. Accessori:**
D52/./D7 (digitale)	Dispaly LCD 602 su DY2
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Questo è un programma semplicemente dimostrativo: sono stati pazientemente inseriti i codici relativo a un vecchio telecomando Audiola, e sarebbe un caso veramente fortunato se si adattassero perfettamente al vostro; in questo modo, sul monitor seriale appaiono decodificati i codici corrispondenti ai vari tasti, invece che una lunga serie di numeri. Essi possono essere visualizzati anche sul display LCD. Questo passo è utile per poter utilizzare il proprio telecomando per ottenere risultati più interessanti, come accendere o spegnere la luce in casa, avviare qualche elettrodomestico, ecc.
Librerie necessarie:	IRremote.h, LiquidCrystal.h
Note:	Personalizzare i codici in base al proprio telecomando. Vedi immagine precedente.
Link:	nessuno

Nome del programma: **Infrared_3**

Porte:	**Modulo principale:**
D12 (digitale)	Infrared receiver - KY022
Porte:	**Comp. Accessori:**
D5 (digitale)	Servomotore MG90S
D9 (digitale)	Led rosso su LD3
D9 (digitale)	Relay KY-019 su RL2 (opzionale)
D10 (digitale)	Led blu su LD2
D11 (digitale)	Led verde su LD1
D11 (digitale)	Relay KY-019 su RL1 (opzionale)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**

Scopo del programma: Questo programma è un'estensione di quello precedente. Dopo aver configurato nel progetto i tasti del telecomando, sono stati aggiunti alcuni moduli di output al programma: i led, i relay e il servomotore. La configurazione dei tasti:

- 1 accende Led1 (ed eventualmente il relay RL1, se collegato);
- 2 accende Led 2
- 3 accende Led3 (ed eventualmente il relay RL2, se collegato);
- 4 spegne led 1 (e disattiva RL1 se collegato);
- 5 spegne Led2
- 6 spegne led 3 (e disattiva RL2 se collegato);
- >> ruota di 90° l'asse del servomotore;
- << riporta a 0° l'asse del servomotore.

Chiaramente i relays possono attivare degli apparecchi; mentre il servomotore può bloccare/sbloccare una chiusura, per esempio di una porta. Le applicazioni sono molte, e modificando facilmente questo programma si possono comandare varie periferiche.

Librerie necessarie: IRremote.h

Note: Personalizzare i codici in base al proprio telecomando.

Link: nessuno

Nome del programma: **Infrared_multi**

Porte:	**Modulo principale:**
D12 (digitale)	Infrared receiver - KY022
Porte:	**Comp. Accessori:**
D2/./D7	Multiplexer CD74HC4067 su DY3
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**

Scopo del programma: Con il telecomando (inserito nel programma i codici corretti), si avranno a disposizione 16 calali in uscita sul multiplexer (vedi multiplexer), che potremo utilizzare collegando per esempio led o relays.

Librerie necessarie: Irremote.h; CD74HC4067.h

Note: nessuna

Link: Nessuno

Programmi per KY-010 Photo interrupt module – DG1

Questo modulo è molto utile quando si deve sapere con precisione se qualche ostacolo interrompe il flusso di luce che colpisce il sensore. Per esempio, è usato in alcune stampanti di etichette se la testina è chiusa o aperta, ecc., e al limite come antifurto, per sapere se una porta o finestra è chiusa o aperta, ponendo per esempio una linguetta tra le pareti del sensore, per indicare quando la finestra è chiusa.

Nome del programma:	**PhotoInt_1**
Porte:	**Modulo principale:**
D12 (digitale)	Photo Interrupt KY-010. Questo sensore rileva se nella sua scanalatura è stato inserito un oggetto.
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led rosso su LD3
D11 (digitale)	Led verde su LD1
Scopo del programma:	Quando si inserisce un oggetto nella scanalatura del sensore, il led rosso (LD3) si accende; contemporaneamente si spegne quello verde. In situazione di riposo, lo stato dei led si inverte.
Note:	nessuna
Link:	nessuno

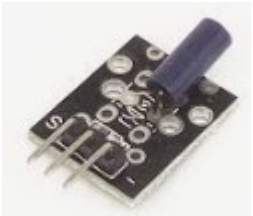
Nome del programma:	**PhotoInt_2**
Porte:	**Modulo principale:**
D12 (digitale)	Photo Interrupt KY-010. Questo sensore rileva se nella sua scanalatura è stato inserito un oggetto.
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led rosso su LD3
D11 (digitale)	Led verde su LD1
D13 (digitale)	Buzzer KY-006 (KY-012) su BZ1
Scopo del programma:	Quando si inserisce un oggetto nella scanalatura del sensore, il led rosso (LD3) si accende; contemporaneamente si spegne quello verde e il buzzer emette una serie di note per segnalate il cambiamento di stato. In situazione di riposo, lo stato dei led si inverte.
Note:	nessuna
Link:	nessuno

Programmi per KY-019 Relay - DG1

Molti dei programmi inseriti in questa sezione usano (o potrebbero usare) un relay, perché permette di interfacciarsi con periferiche esterne ad Arduino, che possono essere anche collegate alla rete elettrica, per esempio caldaie, condizionatori, elettrodomestici, ecc. Ricordarsi sempre di usare direttamente solo apparecchi che funzionano a bassa tensione e senza manomettere i circuiti interni. **Se si dovesse utilizzare la tensione di rete, rivolgersi sempre a un tecnico specializzato. La tensione di rete è pericolosa, in alcuni casi addirittura mortale!**

Nome del programma:	Relay_1
Porte:	**Modulo principale:**
D12 (digitale)	Relay KY-019
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Porte:	**Comp. Accessori:**
A2 (analogica)	Pulsante su “BT1”
Scopo del programma:	Premendo il pulsante “BT1”presente sulla scheda sperimentale, si attiva il relay;contemporaneamente si accende il led rosso e si spegne quello verde. Quando il pulsante non è premuto, il relay è in riposo, resta acceso il led verde e spento quello rosso.
Note:	Nessuna
Link:	nessuno

Programmi per KY-002, KY-017, KY-020, KY-031 Shock/Tap/Tilt modules - DG1



KY-002 shock module

KY-031 tap module

KY-020 tilt module

KY-017 Mercury tilt

Questi quattro moduli: Shock Module KY-002, Tap Module Ky-031, Tilt switch KY-017, anche se con sfumature diverse, lavorano sullo stesso principio, e hanno i piedini nella stessa disposizione, per cui usano lo stesso programma. Di base, se si verifica un movimento, una percussione, ecc. cambiano di stato e che viene intercettato dal programma comune per i vari sensori. Esso può essere personalizzato o inglobato in programmi più complessi.

Nome del programma:	**Shock_Tap_Tilt_Mercury_1**
Porte:	**Modulo principale:**
D12 (digitale)	**Shock KY-002), oppure Tap (KY-031), oppure Tilt (KY-020) module, oppure mercury tilt (KY-017)**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD1 – led verde
D11 (digitale)	LD2 – led rosso
Scopo del programma:	Questi moduli, quando sono spostati sugli assi, oppure percossi, cambiano di stato: se il segnale è “0”, è acceso il led verde e spento il led rosso; quando il segnale passa a “1”, i due led cambiano di stato.
Note:	Nessuna
Link:	nessuno

Nome del programma:	**Shock_Tap_Tilt_Mercury_2**
Porte:	**Modulo principale:**
D12 (digitale)	**Shock KY-002), oppure Tap (KY-031), oppure Tilt (KY-020) module, oppure mercury tilt (KY-017)**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD1 – led verde
D11 (digitale)	LD2 – led rosso
D13 (digitale)	Buzzer KY-006 oppure KY-012
Scopo del programma:	Questi moduli, quando sono spostati sugli assi, oppure percossi, cambiano di stato: se il segnale è “0”, è acceso il led verde e spento il led rosso; quando il segnale passa a “1”, i due led cambiano di stato e il buzzer emette una serie di note.
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA
Link:	nessuno

Programmi per KY-033 Tracking module - DG1

Per quanto riguarda il sensore vero e proprio, esso è costituito da una trasmittente (T) e da una ricevente (R) e, proprio come con il sensore KY-032, viene emesso un segnale IR che, una volta riflesso da una superficie, viene rilevato dalla ricevente e trasmesso sotto forma di segnale digitale (0 se il segnale viene ricevuto, 1 nel caso contrario).

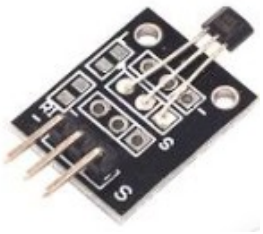
Questo particolare sensore, però, non riesce a **"**vedere il colore nero"; spiegato in termini più tecnici, il segnale IR emesso non ritorna alla ricevente perché assorbito dalle superfici nere. Per questa peculiarità, spesso vengono adoperati gruppi di questi sensori su sistemi semoventi "smart" come carrelli elettrici automatizzati) per seguire le linee nere disegnate per terra e, da ciò, questo modulo prende il nome di Line Tracking Sensor.

Nome del programma:	**Tracking_1**
Porte:	**Modulo principale:**
D12 (digitale)	**Tracking module, KY-033.** Questo modulo ha una coppia di trasmettitore/ricevitore infrarosso. Utile per robot o quando è necessario avere una sensazione precisa degli ostacoli. La sensibilità è regolabile tra 2 e 40 cm. circa.
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD3 – led rosso
D9 (digitale)	Relay KY-019 (opzionale) su RL2
Scopo del programma:	Quando si incontra un ostacolo, la porta digitale passa da "HIGH" a "LOW" e il led si accende. Se è collegato anche un relay su RL2, questi si attiva.
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA
Link:	nessuno

Nome del programma:	**Tracking_2**
Porte:	**Modulo principale:**
D12 (digitale)	**Tracking module, KY-033.** Questo modulo ha una coppia di trasmettitore/ricevitore infrarosso. Utile per robot o quando è necessario avere una sensazione precisa degli ostacoli. La sensibilità è regolabile tra 2 e 40 cm. circa.
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD3 – led rosso
D9 (digitale)	Relay KY-019 (opzionale) su RL2
D11 (digitale)	LD1 – led verde
D13 (digitale)	Buzzer KY-006 (KY-012) su BZ1
Scopo del programma:	Quando si incontra un ostacolo, la porta digitale passa da “HIGH” a “LOW”; si spegne il led verde su LD1 e si accende il led rosso su LD1.e il led si accende. Se è collegato anche un relay su RL2, questi si attiva. Il buzzer emette una serie di note di allarme.
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA
Link:	nessuno

Nome del programma:	**Tracking_3**
Porte:	**Modulo principale:**
D12 (digitale)	**Tracking module, KY-033.** Questo modulo ha una coppia di trasmettitore/ricevitore infrarosso. Utile per robot o quando è necessario avere una sensazione precisa degli ostacoli. La sensibilità è regolabile tra 2 e 40 cm. circa.
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD3 – led rosso
D9 (digitale)	Relay KY-019 (opzionale) su RL2
D11 (digitale)	LD1 – led verde
D13 (digitale)	Buzzer KY-006 (KY-012)
D2/./D7 (digitali)	Dispaly LCD 1602 (16 caratteri x 2 righe) su DY2
Scopo del programma:	Quando si incontra un ostacolo, la porta digitale passa da “HIGH” a “LOW”; si spegne il led verde su LD1 e si accende il led rosso su LD1 e il led si accende, mentre il buzzer emette una nota di allerta. Se è collegato anche un relay su RL2, questi si attiva. Le informazioni vengono mostrate anche sul display LCD.
Librerie richieste:	LiquidCrystal.h
Note:	Nel caso si usi il relay, è nbene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA

Programmi per KY-021 e KY-035 Reed switch e Hall effect module - DG1



KY-035 hall effect sensor

KY-021 reed switch sensor

Trattiamo insieme questi due sensori, perché sebbene usino tecnologie diverse , sono entrambi sensori digitali sensibili ai campi magnetici. Usano lo stesso zoccolo e hanno i piedini disposti in modo identico, quindi sono praticamente intercambiabili.
I programmi sono gli stessi degli altri switch digitali: shock, tilt, tap, mercury tilt.

Nome del programma:	**Hall_reed_1**
Porte:	**Modulo principale:**
D12 (digitale)	**Hall effect KY-035, Reed switch KY-021**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD1 – led verde
D11 (digitale)	LD2 – led rosso
Scopo del programma:	Questi moduli, quando sono prossimi a un campo magnetico, cambiano di stato: se il segnale è “0”, è acceso il led verde e spento il led rosso; quando il segnale passa a “1”, i due led cambiano di stato.
Note:	Nessuna
Link:	nessuno

Nome del programma:	**Hall_reed_2**
Porte:	**Modulo principale:**
D12 (digitale)	**Hall effect KY-035, Reed switch KY-021**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD1 – led verde
D11 (digitale)	LD2 – led rosso
D13 (digitale)	Buzzer KY-006 oppure KY-012
Scopo del programma:	Questi moduli, quando sono prossimi a un campo magnetico, cambiano di stato: se il segnale è “0”, è acceso il led verde e spento il led rosso; quando il segnale passa a “1”, i due led cambiano di stato e il buzzer emette una serie di note.
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA
Link:	nessuno

I programmi per il trasmettitore a 433 MHz (Tx) - DG1

A differenza di tutti gli altri sensori utilizzati in questi programmi, il trasmettitore a 433 MHz non funziona da solo (o meglio, funziona, ma non vengono raccolti i dati inviati), ma richiede un ricevitore a 433 MHz. Per cui, a ogni programma che trasmette qualche dato, ce ne sarà uno gemello che li riceve e li rende usufruibili.
Naturalmente perché il sistema funzioni, sono necessari due Arduino, uno connesso al trasmettitore e uno al ricevitore

Ecco i programmi relativi al trasmettitore:

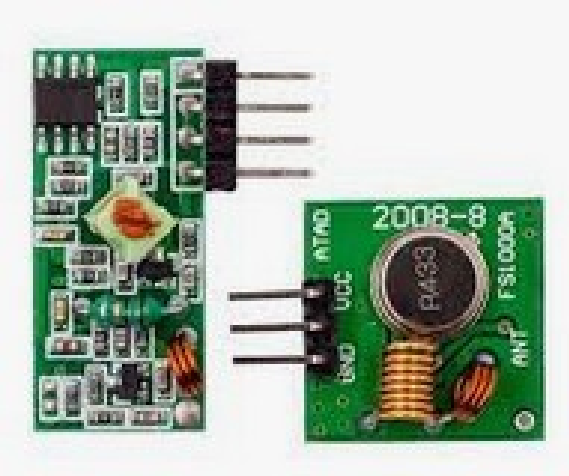
Nome del programma:	**433_hello_tx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	Questo programma trasmette il solito messaggio “hello world”, che verrà ricevuto dal corrispondente ***433_hello_rx***. In sé non ha grande utilità, se non quello di verificare che il trasmettitore e il ricevitore comunichino.
Librerie richieste:	RH_ASK.h
Note:	nessuno
Link:	https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/

Nome del programma:	**433_led_tx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: no	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led rosso su LD3
Scopo del programma:	Questo programma è anch’esso semplicemente didattico. Il trasmettitore, invia il segnale per far lampeggiare all’unisono un led sia sul trasmettitore che sul ricevitore Programma: ***433_led_rx***. In sé non ha grande utilità, se non quello di verificare che il trasmettitore e il ricevitore comunichino.
Librerie richieste:	VirtualWire.h
Note:	nessuno
Link:	https://techatronic.com/rf-transmitter-and-receiver-with-arduino/

Nome del programma:	**433_temp_tx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
A1 (analogico)	DHT11
Scopo del programma:	Il sensore DHT11, che legge la temperatura e l'umidità ambientale, è collegato al trasmettitore, che invia queste informazioni al ricevitore, attraverso al programma 433_temp_rx.
Librerie richieste:	VirtualWire.h; DHT.h
Note:	nessuno
Link:	https://www.electronics-lab.com/project/using-433mhz-rf-transmitter-receiver-arduino/

Nome del programma:	**433_button_tx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9	Led rosso su LD3
D16 (fisico: A2)	Pulsante su BT1
Scopo del programma:	Premendo il pulsante BT1, si accende il led sul trasmettitore,che invia il segnale al ricevitore, attraverso al programma 433_button_rx, che fa accendere in sincrono il led presente sul modulo rx.
Librerie richieste:	VirtualWire.h
Note:	nessuno
Link:	http://www.brescianet.com/appunti/Elettronica/Arduino/corso/Esempio_RF315-433MHZ.htm

Nome del programma:	**433_3button_tx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9	Led rosso su LD3
A2 (analogico)	Pulsante su BT1
A1 (analogico)	Pulsante KY-004 su AN1
A3 (analogico)	Pulsante KY-004 su AN2
Scopo del programma:	Questo programma diventa un un telecomando a tre canali, che farà accendere/spegnere indipendentemente 3 led sul ricevitore. Programma: 433_3button_rx.
Librerie richieste:	RH_ASK.h; SPI.h
Note:	nessuno
Link:	https://mariodenichilo.altervista.org/arduino-trasmissione-senza-fili-a-433-mhz-radiocomando-a-3-canali



La coppia trasmettitore/ricevitore
a 433 MHz

Nome del programma:	**433_servo_tx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: sì	**Plotter seriale: no**
(9600 baud)	
Porte:	**Comp. Accessori:**
D9	Led rosso su LD3
A3 (analogico)	Potenziometro su AN2
Scopo del programma:	In questo semplice programma è collegato un potenziometro. Il ricevitore, con programma 433_servo_rx, riceve il segnale e permette all'asse di un servomotore di ruotare proporzionalmente alla rotazione del potenziometro.
Librerie richieste:	RCSwitch.h
Note:	nessuno
Link:	https://srituhobby.com/433mhz-rf-transmitter-and-receiver-module-with-arduino/

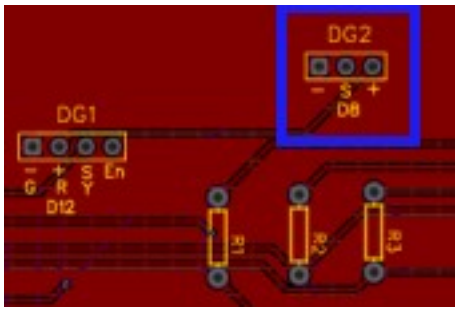
Clicca qui per accedere ai seguenti programmi del ricevitore, collegato su DG7:

- 433_hello_rx
- 433_led_rx
- 433_button_rx
- 433_3button_rx
- 433_temp_rx

Clicca qui per accedere al programma del ricevitore, collegato su DG11:

- 433_servo_rx

Lo zoccolo DG2



Lo zoccolo DG2 si trova a destra di DG1, leggermente più in alto, affinché il sensore PIR non entri in contatto con altri eventuali moduli inseriti sugli zoccoli DG1 o LD4.

La porta utilizzata è D8, quindi non può essere utilizzato contemporaneamente a un modulo presente sullo zoccolo DG7, pena conflitti.

Nel caso sia necessario, il modulo PIR può essere collegato sullo zoccolo AN2, naturalmente variando la porta di comunicazione.

DG2. Su questo zoccolo alloggia il modulo HC-SR501, "pir motion sense", ovvero in grado di intercettare il movimento di corpi che emettono calore nel raggio di qualche metro. Presenta un paio di trimmer, per regolare la sensibilità e la durata del segnale. Il connettore centrale si collega alla porta D8 di Arduino, come si può vedere dall'immagine panoramica della *bs*, presente all'inizio di questa sezione.

Anche il "laser detector module ", che come dice il nome, rileva con un sensore particolare la presenza di un fascio di luce emesso da un led laser. Utile per verificare l'interruzione del raggio, come un contapezzi o un antifurto. Il led laser è sufficiente che sia collegato ai + 5 volt, quindi il sistema può funzionare usando anche con un solo Arduino.

Ecco la lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG2 (digital)			1	2	3
Description	Code	Note	-	S	VCC (+5v)
PIR – presence	HC-SR501 – KY-007		-	D8	+
Laser detector		richiese diodo laser KY-008			

Il modulo pir

Su questo zoccolo si connettono *nativamente* solo il modulo di rilevazione presenza "PIR" e quello che rileva il raggio laser.
Tuttavia questo accade perché la disposizione del modulo PIR, che rileva movimenti nel raggio di 1 o 2 metri ha una disposizione di piedini diversa da quella dello zoccolo DG1.

Laser detector module

Per cui nulla vieta, usando un connettore a tre fili, di inserire su questo zoccolo uno qualsiasi dei moduli che montano su DG1, naturalmente facendo attenzione alla loro sequenza, che su DG2 segue il seguente ordine: **"-", "S", "+"**, dove per "S" si intende il piedino che porta il segnale.

Programmi per HC-SR501 PIR sensor module - DG2

Nome del programma:	**Pir_1**
Porte:	**Modulo principale:**
D8 (digitale)	**HC-SR501,** sensore pir di movimento, sensibile ai raggi infrarossi.
Monitor Seriale: no	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	LD3 – led rosso
D13 (digitale)	Buzzer – KY-006 (KY-012)
Scopo del programma:	Il sensore PIR segnala un movimento in prossimità (max. 2/3 metri), producendo un segnale "High", ovvero 1, quando percepisce un movimento. La sensibilità si regola con un trimmer; la durata del segnale con un secondo. Quando si verifica un movimento, lampeggia il led rosso e il buzzer emette una serie di note di allarme, che può essere trasmesso attraverso la linea "audio out" a un amplificatore esterno.
Note:	nessuna
Link:	nessuno

Nome del programma:	**Pir_2**
Porte:	**Modulo principale:**
D8 (digitale)	**HC-SR501,** sensore pir di movimento, sensibile ai raggi infrarossi.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	I led rosso e verde lampeggiano a ritmo alternato.
Porte:	**Comp. Accessori:**
D9 (digitale)	Led LD3, rosso
D9 (digitale)	Relay KY-019
D11 (digitale)	Led LD1, giallo
D13 (digitale)	Buzzer – KY-006 (KY-012)
A2 (analogico)	Punsante sulla scheda (BT1)
Scopo del programma:	Il sensore PIR segnala un movimento in prossimità (max. 2/3 metri), producendo un segnale "High", ovvero 1, quando percepisce un movimento. La sensibilità si regola con un trimmer; la durata del segnale con un secondo. Quando si verifica un movimento, lampeggia il led rosso e il buzzer emette una serie di note di allarme, che può essere trasmesso attraverso la linea "audio out" a un amplificatore esterno. In questa versione del programma è stato aggiunto un secondo led (giallo) che rimane attivo, anche quando il movimento è cessato. Esso viene disattivato solamente premendo il tasto di reset "BT1". Insieme al led giallo può essere collegato un relay (RL2, che può attivare un sistema remoto di segnalazione o di allarme.

Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almeno 500 mA
Link:	nessuno

Nome del programma:	**Pir_3**
Porte:	**Modulo principale:**
D8 (digitale)	**HC-SR501,** sensore pir di movimento, sensibile ai raggi infrarossi.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	I led rosso e verde lampeggiano a ritmo alternato.
Porte:	**Comp. Accessori:**
D9 (digitale)	Led 3, rosso
D9 (digitale)	Relay KY-019
D11 (digitale)	Led 1, giallo
D13 (digitale)	Buzzer
D2/./D7 (digitale)	Display a cristalli liquidi 1602 (16 caratteri, 2 linee)
A2 (analogico)	Pulsante sulla scheda (BT1)
Scopo del programma:	Il sensore PIR segnala un movimento in prossimità (max. 2/3 metri), producendo un segnale “High”, ovvero 1, quando percepisce un movimento. La sensibilità si regola con un trimmer; la durata del segnale con un secondo. Quando si verifica un movimento, lampeggia il led rosso e il buzzer emette una serie di note di allarme, che può essere trasmesso attraverso la linea “audio out” a un amplificatore esterno. In questa versione del programma è stato aggiunto un secondo led (giallo) che rimane attivo, anche quando il movimento è cessato. Esso viene disattivato solamente premendo il tasto di reset “BT1”. Insieme al led giallo può essere collegato un relay (RL2), che può attivare un sistema remoto di segnalazione o di allarme. E’ stato aggiunto un dispaly a cristalli liquidi per visualizzare le informazioni.
Librerie richieste:	LiquidCrystal.h
Note:	Nel caso si usi il relay, è bene collegare a ext. Power un alimentatore con una tensione in cc di 6/./9 volts er almenno 500 mA
Link:	nessuno

Nota: questo sensore, con piccole modifiche di programma, può essere inserito anche sullo zoccolo AN2.

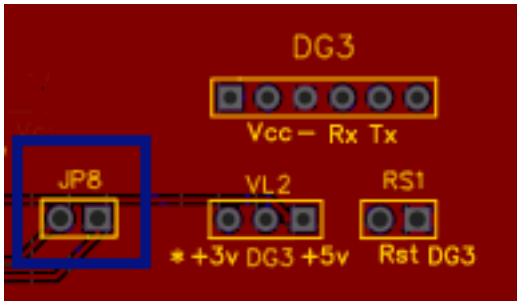
Programmi per il laser detector module - DG2.

il laser + il ricevitore

Questo programma è molto semplice, se il raggio laser colpisce il sensore, il led LD1 (ed eventualmente il relay su RL1) sono attivi; altrimenti si accende il led LD3 e il relay RL2.
Il modulo ricevitore del fascio laser (diodo KY-008) ha una funzione molto semplice, di switch: high se il fascio laser è interrotto, low se viene colpito. In questo modo può essere utilizzato come antifurto, per segnalare un passaggio, ecc.
Il laser (KY-008) è semplicemente collegato al +5v e al ground di Arduino, in modo che emetta continuamente il raggio laser, che viene intercettato dal ricevitore.

Nome del programma:	**Laser_detect**
Porte:	**Modulo principale:**
D8 (digitale)	**Laser detect sensor**
Monitor Seriale: no	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D8 (digitale)	Led Laser (KY-008)
D9 (digitale)	LD3 – led rosso (+ Relay su RL2)
D11 (digitale)	LD1 – led verde (+ Relay su RL1)
Scopo del programma:	Se il fascio non è interrotto, attivo il led LD1 (+ il relay su RL1); altrimenti si attiva il led LD3 (e il relay su RL2)
Note:	Il led lase si attiva semplicemente alimentandolo con i +5 v prelevati da Arduino stesso.
Link:	nessuno

Lo zoccolo DG3



Lo zoccolo DG3 si trova all'estrema destra in alto della bs.

L'alimentazione può essere scelta tra 3,3v 3 5v.
I moduli Bluetooth devono essere tassativamente alimentati a 3,3v!!!

Le porte utilizzate sono D9 e D10, per cui non può essere utilizzato in concomitanza di moduli presenti sullo zoccolo DG4, pena conflitti.
Ci potrebbero essere conflitti anche con DG5 e DG6. Leggi il testo.

DG3. Questo zoccolo ospita attualmente tre moduli diversi, tutti adatti per le connessioni bluetooth. HC-06 , HC-05 e HM-19. Questi due ultimi hanno sei piedini, ma quelli realmente usati sono solamente i quattro centrali. HC-06 ha solo quattro piedini, che però corrispondono a quelli centrali degli altri due moduli, quindi lo zoccolo è adeguato per tutte le schede indicate. Con questi moduli si possono effettuare interessanti automatismi, quando il campo di azione si limita ad alcuni metri, come per esempio l'apertura automatica di un garage, l'accensione di una luce, ecc. Utilizza le porte D9 e D10, quindi va in conflitto con gli zoccoli DG4. DG5 e DG6 usano abitualmente le porte D9 e D10, per cui in questo caso andrebbero in conflitto; ***tuttavia su questa nuova versione della bs possono usare in alternativa anche le porte D3 e D4; in questo caso non ci sarebbe conflitto***. Naturalmente ci sono sempre delle alternative, per esempio se ci fosse la necessità di collegare sia moduli bluetooth che wi-fi o gps. contemporaneamente, sarà sufficiente collegare il secondo sullo zoccolo DG7, usando un connettore a quattro fili per ottenere la giusta sequenza dei connettori e modificare i programmi in base alle porte utilizzate, oppure usare il connettore "universale" DG11, scegliendo delle porte digitali inutilizzate, modificando contestualmente il programma.
Il jumper **RS1** permette il reset hardware del modulo wi-fi. Nell'immagine è stato evidenziato il jumper **JP8**. Nelle precedenti versioni della bs, i piedini "Tx" dei moduli presenti su DG3 e DG4 erano direttamente connessi alla porta digitale D9 tramite un partitore resistivo. Sia i moduli wi-fi che ESP8266, devono essere tassativamente alimentati a + 3,3v, pena la loro distruzione. Questo partitore, composto da due resistenze in serie per un totale di 3 Kohm, erano sempre collegati alla porta D9, anche quando i moduli non erano presenti. Nei test si è dimostrato che quando si usava D9 per altri scopi, il suo comportamento era spesso anomalo. Perciò si è inserito il jumper JP8, per collegare il partitore ***solo*** quando sono presenti i suddetti moduli. Ricordarsi di ponticellarlo quando necessario, altrimenti i moduli presenti su DG3 e DG4 non risponderanno!

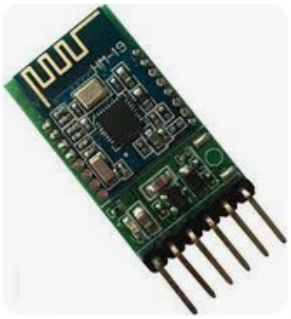
Lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG3 (digital)			1	2	3	4	5	6
Description	Code	Note	EN	Vcc (+3 v)	GND	TX	RX	STATE
Bluetooth	HC-06		/	+	-	d9	d10	/
Bluetooth	HM-19		EN					STATE
Bluetooth	HC-05		EN					STATE
GPS NEO7			/					/

Immagini dei moduli che alloggiano su DG3:

HC-06

HC-05



HM-19

Neo 7M (Gps)

Programmi per lo zoccolo DG3

Su questo connettore si possono inserire i moduli wi-fi HC-06 e HC-05, per cui c'è un programma di test, e il modulo HM-19, che funziona con i comandi AT, per cui non abbiamo ancora testato dei programmi.
Per utilizzare intanto HC-06, è necessario caricare una app adeguata sullo smartphone; è stato testato con successo "***arduino bluetooth control***, che può essere scaricato gratuitamente da Play store per Android. Ci sarà sicuramente una app simile per OS di Apple.

Quando si è caricata la app, inserire il modulo HC-06 nello zoccolo, facendo attenzione alla corrispondenza dei piedini. ***Alimentarlo esclusivamente a 3,3 v, pena il danneggiamento irreversibile del modulo stesso***! Dopo aver caricato il programma su Arduino, il led su HC-06 lampeggia. Attivare il bluetooth su Smartphone, si dovrebbe trovare tra i dispositivi nel raggio di azione anche HC-06. Connetterlo e scegliere sul cellulare l'opzione "terminale". Se tutto è andato a buon fine, dopo pochi secondi il led sul modulo resterà acceso fisso, indice che si è stabilito il collegamento.

Inviando (lentamente) i comandi, si otterrà:

- digitando "a" (invio), il buzzer emetterà una nota;
- digitando "b" (invio), il buzzer si silenzierà;
- digitando "c" (invio), si accenderà il led presente su LD1;
- digitando "d" (invio), il led si spegnerà.

Programmi per HC-06 bluetooth module - DG3

Nome del programma: **HC06_1**

Porte:
D9 Digitale) RX
D10 (digitale) TX

Modulo principale:
HC-06 modulo wi-fi

Porte:
D13 (digitale)
D11 (digitale)
D11 (digitale)

Comp. Accessori:
Buzzer KY-06 (KY-012)
Led LD1 - verde
Relay KY-019 su RL1 (opzionale)

Monitor Seriale: sì (9600 bit)

Plotter seriale: no

Scopo del programma:
Fornendo i comandi dal monitor del cellulare, si otterranno i seguenti risultati:

- digitando "a" (invio), il buzzer emetterà una nota;
- digitando "b" (invio), il buzzer si silenzierà;
- digitando "c" (invio), si accenderà il led presente su LD1;
- digitando "d" (invio), il led si spegnerà.

Librerie richieste: SoftwareSerial.h

Note: Alimentarlo esclusivamente a 3,3 v, pena il danneggiamento irreversibile del modulo stesso!

Link: nessuno

Nome del programma: **HC06_2**

Porte:
D9 Digitale) RX
D10 (digitale) TX

Modulo principale:
HC-06 modulo wi-fi

Porte:
D13 (digitale)
D11 (digitale)
D11 (digitale)
D6 (digitale)

Comp. Accessori:
Buzzer KY-06 (KY-012) su BZ1
Led LD1 - verde
Relay KY-019 su RL1 (opzionale)
Motore elettrico su DG8

Monitor Seriale: sì (9600 bit)

Plotter seriale: no

Scopo del programma:
Fornendo i comandi dal monitor del cellulare, si otterranno i seguenti risultati:

- digitando "a" (invio), il buzzer emetterà una nota;
- digitando "b" (invio), il buzzer si silenzierà;
- digitando "c" (invio), si accenderà il led presente su LD1;
- digitando "d" (invio), il led si spegnerà.
- digitando "e" (invio), si accenderà il motore
- digitando "f" (invio), si fermerà il motore

Librerie richieste: SoftwareSerial.h

Note: Alimentarlo esclusivamente a 3,3 v, pena il danneggiamento irreversibile del modulo stesso! Se si collega il motore, è necessario che sia collegata l'alimentazione esterna, altrimenti non si avvierà.

Link: nessuno

Programmi per Neo 7M (Gps)

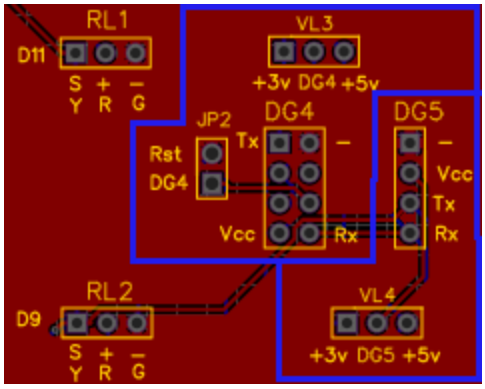
Neo 7M è simile a Neo 6Mv2, in quanto sono entrambi moduli GPS, infatti utilizzano gli stessi programmi. La differenza è che Neo 7M è una versione più aggiornata, che integra anche l'antenna e inoltre permette di collegare un'antenna esterna per migliorarne la sensibilità. Il contro è che purtroppo non hanno lo stesso numero di piedini (4 per Neo 6Mv2 e 5 per Neo 7M e inoltre non seguono lo stesso ordine. Questo fa sentire ancora di più la mancanza di uno standard.
In alternativa, Neo 7M può essere collegato sullo zoccolo AN4, modificando naturalmente il programma per le porte utilizzate.

Per illustrare le funzionalità del modulo, è stato inserito un solo programma, identico a quello per Neo 6Mv2; semplicemente è stato invertito l'ordine di Tx e Rx, per adeguarlo a questo zoccolo. Anche gli altri programmi possono essere usati per questo modulo, avendo la stessa accortezza.

Nome del programma:	**Neo7M_3**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**Neo-6MV2 – modulo GPS**
Monitor Seriale: si	**Plotter seriale: no**
(9600 baud)	
Porte:	**Comp. Accessori:**
D3/./D7 (digitali)	**TFT 1.77" ST7735**
Scopo del programma:	Oltre alla la latitudine e la longitudine del luogo in cui si esegue la misurazione, mostra anche la data, l'ora, l'altezza, la velocità attuale e il satellite a cui si è agganciato sul display TFT. Ottima grafica
Librerie richieste:	SoftwareSerial.h; SPI.h; TinyGPS++.h; Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Eseguire il test vicino a una finestra, su di un balcone o all'aperto. Ci vuole qualche decina di secondi prima che agganci il satellite. In questo caso, il led sul modulo comincia a lampeggiare.
Link:	nessuno

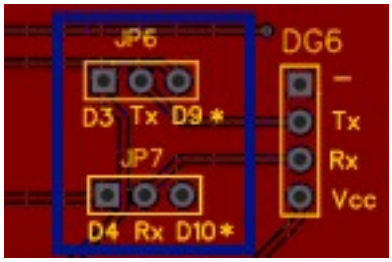
Nota: sebbene il satellite si colleghi (il led lampeggia), a volte non si attiva il programma, probabilmente per il partitore resistivo presente sul piedino Tx che riduce la quantità di segnale presente sulla porta digitale D9. Nel caso si verifichi, collegare il modulo sullo zoccolo AN4.
Clicca qui per il collegamento.

Gli zoccoli DG4 - DG5



Gli zoccoli DG4 e DG5 sono un caso un po' speciale: entrambi servono per lo stesso modulo wi-fi, il potente ESP8266 che può addirittura funzionare autonomamente!
Lo zoccolo DG4 è quello "standard: accoglie il modulo con otto piedini, e deve essere tassativamente alimentato a 3,3v, pena la sua distruzione!
Mentre lo zoccolo DG5 accoglie lo ESP8266 montato su di un modulo che regola automaticamente la tensione, anche per i piedini Tx-Rx e quindi non rischia di danneggiarlo.
Per i problemi di conflitti, vedi lo specchietto successivo.

Perché è stato inserito un riquadro con lo zoccolo DG6?
Lateralmente ad esso sono presenti due jumper, JP6 che permette la scelta delle porte D3 o D9 (default) per Tx, e JP7 che offre la possibilità di scegliere D4 o D10 (default) per Rx.
Ma queste scelte sono in comune per DG5 e DG6, quindi non possono essere usati contemporaneamente i moduli su entrambi gli zoccoli.
I jumper sono stati inseriti in questa versione della bs per evitare conflitti con gli zoccoli DG3 e DG4, che utilizzano esclusivamente le porte D9 e D10. Questo accorgimento amplia la flessibilità della bs.



Questi due zoccoli sono stati trattati insieme, perché sono relativi ad uno stesso modulo. Può sembrare uno spreco, e forse lo è. Proverò a spiegare questa scelta. Il modulo in oggetto è ESP8266, che permette una connessione wi-fi. Esso è talmente potente che può essere utilizzato anche indipendentemente da Arduino: possiede un paio di porte digitali, che possono pilotare in uscita per esempio un relay, o in ingresso essere collegati, sempre come esempio, a un sensore di temperatura come DHT11 o a tantissimi altri sensori. Collegandolo ad Arduino Nano, che nella versione base non ha la possibilità di agganciare la rete internet (al contrario del più evoluto Arduino Nano 33 IOT), e quindi ad aprirlo al mondo e poter utilizzalo anche per qualche semplice attività di domotica.

L'ESP8266 "nudo"...

… e montato su ESP01

Questo modulo non è di utilizzo immediato, è necessario avere pazienza e non scoraggiarsi!
Per poterlo connettere ad Arduino è necessario scaricare alcune librerie ed effettuare delle scelte sulla IDE, ma per questo ci si può riferire a ottime guide on-line:
https://www.lutritech.it/come-programmare-esp8266-con-lide-di-arduino/ (testo, italiano)
https://www.youtube.com/watch?v=sVbr0rfCFMY video
https://www.youtube.com/watch?v=AkdzqjkeqwM&t=54s video

https://www.youtube.com/watch?v=gbfTxQ_xgBw&t=221s video

ESP8266 ***deve essere assolutamente alimentato a 3,3v,*** pena la distruzione; su alcuni progetti è collegato direttamente al piedino Tx di Arduino, in altri attraverso ad alcune resistenze per ridurre la tensione. Lo zoccolo DG4 permette di collegarlo direttamente, settando la corretta tensione di alimentazione e usa un partitore resistivo collegato al piedino Tx. Vedi zoccolo DG3. Per evitare qualsiasi problema di interfacciamento, è stato inserito lo zoccolo DG5, che accogliendo un ESP8266 con adattatore ESP-01 (che utilizza solo quattro piedini e costa pochi euro, facilmente reperibile su internet) , si evitano tutti i problemi di interfacciamento e alimentazione.

Lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG4 (digital – D9, D10)			1	2	3	4	5	6	7	8
Description	Code	Note	'-	GPC0	GPO2	RX	TX	CHPD	RST	3,3+v
Wi-Fi	ESP8266	x	-	/	/	d9	d10	/	/	3,3+v

Socket DG5 (digital – D9, D10)			1	2	3	4
Description	Code	Note	'-	+5v	RX	TX
Wi-Fi	ESP8266	x	-	+5v	d9	d10

Nota: i connettori DG3, DG4, DG5, DG6 usano di default le stesse porte di Arduino, Quindi non possono funzionare Insieme. Tuttavia, come si è visto nel riquadro relativo, sia DG5 che DG6 possono usare in alternativa anche le porte D3 e D4, modificando i programmi. Naturalmente se si usano altre porte, utilizzando il connettore "Universale" DG11 oppure DG10 (vedi informazioni relative a questi zoccoli) si potranno scegliere ulteriori porte libere, e i vari moduli potranno essere collegati anche insieme contemporaneamente, cambiando nei programmi le porte corrispondenti a TX e Rx.

Programmi per ESP8266 – Wi-Fi module

Nome del programma: **ScanNetworks**

Porte: D9, Rx; D10, Tx (digitale)

Modulo principale: **ESP8266,** modulo wi-fi

Monitor Seriale: si (115200 baud)

Plotter seriale: no

Scopo del programma: Esegue un test di collegamento tra Arduino e il modulo in oggetto; in caso di esito positivo mostra sul monitor seriale tutte le reti WI-FI presenti nelle vicinanze.

Librerie richieste: WiFiEsp.h; SoftwareSerial.h

Note: Vedi immagine di collegamento qui di seguito. Il programma è stato tratto tra quelli presenti nela libreria WiFiEsp. Dopo aver caricato questa libreria, cliccare su File > Esempi > WIFIEsp > ScanNetworks. Naturalmente nel programma è stato adattato Rx alla porta digitale D9 e Tx alla porta D10

Link: nessuno

Elenco delle reti wi-fi trovate dopo la scansione con ScanNetworks

Nome del programma:	**ScanNetworks_a**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**ESP8266,** modulo wi-fi
Porte:	**Comp. Accessori:**
D3/./D7 (digitale)	Dispaly TFT 1.7” ST7735 su DY1
Monitor Seriale: si (115200 baud)	**Plotter seriale: no**
Scopo del programma:	Esegue un test di collegamento tra Arduino e il modulo in oggetto; in caso di esito positivo mostra sul monitor seriale tutte le reti WI-FI presenti nelle vicinanze. Le stesse informazioni appaiono sul visore TFT
Librerie richieste:	WiFiEsp.h; SoftwareSerial.h; Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Vedi immagine di collegamento qui di seguito. Il programma è stato tratto tra quelli presenti nela libreria WiFiEsp. Dopo aver caricato questa libreria, cliccare su File > Esempi > WIFIEsp > ScanNetworks. Naturalmente nel programma è stato adattato Rx alla porta digitale D9 e Tx alla porta D10
Link:	nessuno

Nome del programma:	**ConnectWPA**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**ESP8266,** modulo wi-fi
Monitor Seriale: si (115200 baud)	**Plotter seriale: no**
Scopo del programma:	E’ necessario inserire nelle prime righe del programma il nome della

	propria rete Wi-Fi e la password di collegamento. Esegue un test di collegamento tra Arduino e il modulo in oggetto; in caso di esito positivo e in presenza dei nomi corretti della rete e password, si collega alla propria rete Wi-Fi (vedi immagine)
Librerie richieste:	WiFiEsp.h; SoftwareSerial.h
Note:	Vedi immagine di collegamento qui di seguito. Il programma è stato tratto tra quelli presenti nela libreria WiFiEsp. Dopo aver caricato questa libreria, cliccare su File > Esempi > WIFIEsp > ScanNetworks. Naturalmente nel programma è stato adattato Rx alla porta digitale D9 e Tx alla porta D10
Link:	nessuno

Dati da inserire nel

ConnectWPA

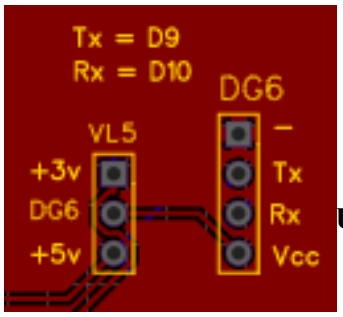
```
/*
 WiFiEsp example: ConnectWPA

 This example connects to an encrypted WiFi network using an ESP8266 module.
 Then it prints the  MAC address of the WiFi shield, the IP address obtained
 and other network details.

 For more details see: http://yaab-arduino.blogspot.com/p/wifiesp-example-connect.html
*/

#include "WiFiEsp.h"

// Emulate Serial1 on pins 9/10 if not present
#ifndef HAVE_HWSERIAL1
#include "SoftwareSerial.h"
SoftwareSerial Serial1(9, 10); // RX, TX
#endif

char ssid[] = "nome della rete wi-fi";            // your network SSID (name)
char pass[] = "password di rete";                 // your network password
int status = WL_IDLE_STATUS;                      // the Wifi radio's status
```

programma per effettuare la connessione:

Messaggi che appaiono sul monitor seriale:

Connessione effettuata. Complimenti!

Lo zoccolo DG6



Lo zoccolo DG6 si trova in basso a destra della bs.
Il modulo GPS Neo-6MV2 può essere alimentato sia a 3,3 che 5v.

Usa le porte D9 e D10 (o in alternativa D3 e D4), quindi va in conflitto con i moduli eventualmente presenti sugli zoccoli DG3, DG4 e DG5.

DG6. A questo zoccolo si connette un altro modulo molto interessante: NEO6Mv2, ovvero un sensore che trasforma Arduino in un preciso GPS! Caricando i programmi relato ad esso, si può ottenere con grande precisione data, ora, latitudine, longitudine, altezza, velocità e numero del satellite a cui si collega. Il modulo viene fornito con un'antenna non particolarmente sensibile, per cui è bene effettuare i test vicino a una finestra, in modo che si possa collegare facilmente. Per agganciare un satellite, è necessario attendere qualche decina di secondi. Quando si verifica il collegamento, il led presente a bordo del modulo comincia a lampeggiare.

Il modulo GPS

Anche in questo caso, le porte utilizzate sono D9-D10, quindi non si può utilizzare il GPS *direttamente* insieme a un modulo bluetooth o wi-fi. Naturalmente se ciò fosse indispensabile, si potrebbe collegare uno dei due moduli allo zoccolo "universale " DG11, modificando le porte che si intende usare, utilizzando un connettore a quattro fili. A questo proposito, leggere quanto indicato per lo zoccolo DG11.

Neo 6Mv2

Neo 7M

Lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG6 (digital – D9, D10)			1	2	3	4	5
Description	Code	Note	vcc (+5v) *	RX	TX	GND	///
GPS	NE06MV2	* Piedino 4 anche 3,3 v (alternativo a 5v)	+	d10	d9	-	/
			vcc (+5v) *	GND	TX	RX	PPS
GPS	NE0-7M	Usare connettore – diversa piedinatura	+	-	d9	d10	/

Il modulo NEO-6M è molto interessante, perché in pochi centimetri quadrati nasconde un vero GPS, in grado di fornire molte informazioni.
Dato le sue dimensioni, molto contenute, è possibile costruire con un display un piccolo apparecchio GPS portatile.

Su questo zoccolo si può anche collegare il modulo Neo-7M, utilizzando gli stessi programmi. Poiché la disposizione è – purtroppo – diversa, è necessario usare un cavetto a quattro pin per utilizzare le porte corrette. Il modulo Neo 7M può essere collegato direttamente su DG3 oppure AN4, variando naturalmente le porte di comunicazione.

Nota 1: L'antenna fornita insieme al modulo non è molto sensibile, e si potrebbe pensare che il modulo o il programma non funzionino. E sufficiente fare i propri test vicino a una finestra, o meglio ancora all'aperto, oppure acquistare separatamente un'antenna più sensibile o amplificata. Su internet se ne trovano vari modelli.

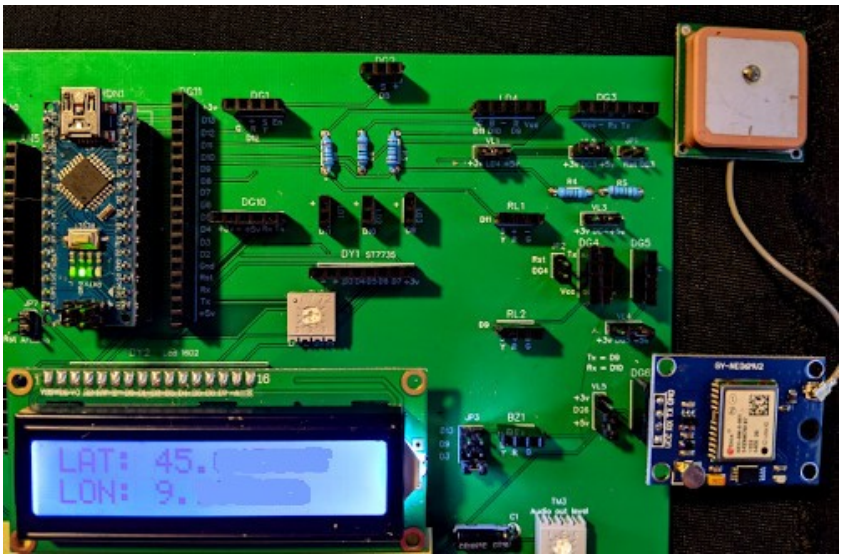
Nota 2: gli zoccoli DG3, DG4, DG5, DG6 usano le stesse porte di Arduino, Quindi non possono funzionare Insieme. Naturalmente se si usano altre porte, utilizzando il connettore "Universale" DG11 (vedi informazioni relative a questi zoccoli) In questo modo si eviteranno i conflitti, e i vari moduli potranno essere collegati anche insieme e contemporaneamente, cambiando nei programmi le porte corrispondenti a TX e Rx.
DG5 e DG6 permettono di usare in alternativa le porte D3 e D4.

Programmi per Neo6MV2 - GPS

Nome del programma:	**Neo6_1**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**Neo-6MV2 – modulo GPS**
Monitor Seriale: si	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Mostra sul monitor seriale la latitudine e la longitudine del luogo in cui si esegue la misurazione
Librerie richieste:	SoftwareSerial.h; SPI.h; TinyGPS++.h
Note:	Eseguire il test vicino a una finestra, su di un balcone o all'aperto. Ci vuole qualche decina di secondi prima che agganci il satellite. In questo caso, il led sul modulo comincia a lampeggiare.
Link:	https://lastminuteengineers.com/neo6m-gps-arduino-tutorial/

Nome del programma:	**Neo6_1a**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**Neo-6MV2 – modulo GPS**
Monitor Seriale: si	**Plotter seriale: no**
(9600 baud)	
Scopo del programma:	Identico al precedente. Oltre alla la latitudine e la longitudine del luogo in cui si esegue la misurazione, mostra anche la data, l'ora, l'altezza e il satellite a cui si è agganciato.
Librerie richieste:	SoftwareSerial.h; SPI.h; TinyGPS++.h
Note:	Eseguire il test vicino a una finestra, su di un balcone o all'aperto. Ci vuole qualche decina di secondi prima che agganci il satellite. In questo caso, il led sul modulo comincia a lampeggiare.
Link:	https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/

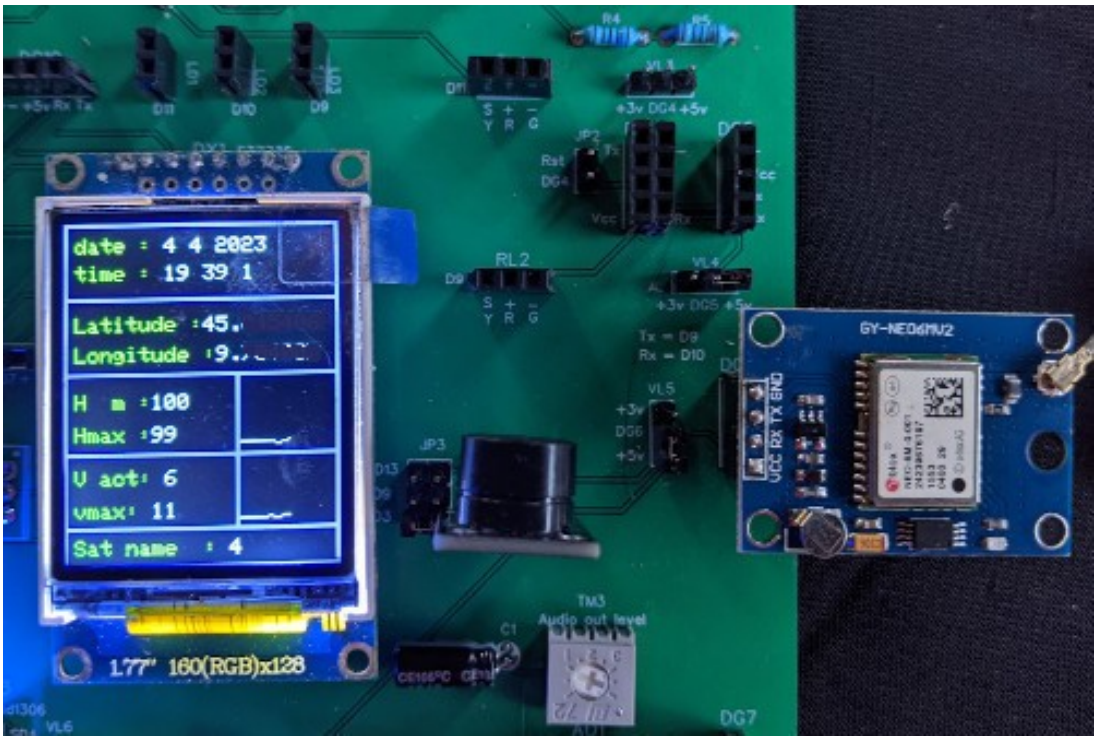
Nome del programma:	**Neo6_2**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**Neo-6MV2 – modulo GPS**
Monitor Seriale: si (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D2/./D7 (digitali)	Display LCD 1602 (16 caratteri x 2 linee)
Scopo del programma:	Mostra la latitudine e la longitudine del luogo in cui si esegue la misurazione sia sul monitor seriale che sul display LCD.
Librerie richieste:	SoftwareSerial.h; SPI.h; TinyGPS++.h; LiquidCrystal.h
Note:	Eseguire il test vicino a una finestra, su di un balcone o all'aperto. Ci vuole qualche decina di secondi prima che agganci il satellite. In questo caso, il led sul modulo comincia a lampeggiare.
Link:	https://www.instructables.com/How-to-Communicate-Neo-6M-GPS-to-Arduino/



Ecco apparire sul display LCD la latitudine e la longitudine del luogo in cui effettuiamo la rilevazione.

P.s.: sono stati cancellati i cinque digitali.

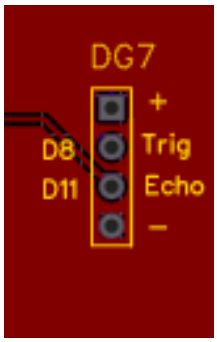
Nome del programma:	**Neo6_3**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**Neo-6MV2 – modulo GPS**
Monitor Seriale: si (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D3/./D7 (digitali)	**TFT 1.77" ST7735**
Scopo del programma:	Oltre alla la latitudine e la longitudine del luogo in cui si esegue la misurazione, mostra anche la data, l'ora, l'altezza, la velocità attuale e il satellite a cui si è agganciato sul display TFT. Ottima grafica
Librerie richieste:	SoftwareSerial.h; SPI.h; TinyGPS++.h; Adafruit_GFX.h; Adafruit_ST7735.h
Note:	Eseguire il test vicino a una finestra, su di un balcone o all'aperto. Ci vuole qualche decina di secondi prima che agganci il satellite. In questo caso, il led sul modulo comincia a lampeggiare.
Link:	nessuno



Con questo programma, otteniamo non solo la latitudine e la longitudine, ma anche la data, l'ora, l'altezza sul livello del mare e l'eventuale velocità, oltre che il satellite a cui siamo agganciati. Potrebbe essere la base per un piccolo rilevatore portatile, magari per le escursioni a piedi o in bicicletta.

P.s.: latitudine e longitudine mostrano ben cinque decimali, che semplicemente sono stati oscurati.

Lo zoccolo DG7



Lo zoccolo DG7 è posto sull'estrema destra in basso della bs.

Usa le porte D8 e D11, quindi va in conflitto con eventuali moduli contestualmente presenti e utilizzati sugli zoccoli DG2, LD1, LD4, RL1

DG7. Lo zoccolo DG7 ospita nativamente un sensore ultrasonico, HC-SR04, che può percepire degli oggetti o degli ostacoli a distanza di alcuni metri (nei miei test non ho superato i due metri), utile per robot, antifurti, ecc. Un emettitore presente sul modulo lancia un segnale ultrasonico, mentre un sensore lo riceve. In base al ritardo di tempo, esso misura la distanza dell'oggetto.

A questo stesso zoccolo può essere collegato con un cavetto, perché la piedinatura non è la stessa, una cella di carico con un amplificatore HX711. Arduino diventa una precisa bilancia digitale in grado di leggere pesi fino a 30 chili. La cella di carico è costituita da un materiale deformabile, che emette una bassissima tensione, proporzionale alla flessione dovuta al peso sostenuto; l'amplificatore porta la tensione a un livello che Arduino la possa leggere e interpretare. Le porte utilizzate sono D8 e D11; non può essere utilizzato contemporaneamente ai moduli presenti sugli zoccoli DG2, LD1, LD4, RL1; usare lo zoccolo DG11 con altre porte oppure ut(vedi zoccolo DG11) al sensore pir, connesso allo zoccolo DG2.

Lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG7 (digital – D11, D8)			1	2	3	4
Description	Code	Note	vcc (+5v)	TRIG	ECHO	GND
Ultrasonic sensor	KY-050 HC-SR04		+	d11	d8	–
Weight module	Load cell + HX711	Usare connettore piedinatura diversa	vcc (+5v)	SCK	DT	GND
			+	d11	d8	–
433 MhHz receiver (RX)			Vcc	Data		GND
			+	d11	/	–

Immagine dei moduli che possono essere connessi a questo zoccolo:

Il sensore ultrasonico HC-SR4

Il ricevitore (RX) a 433 MHz

La cella di carico + amplificatore HX711 (deve essere collegata con un cavetto – piedinatura differente)

Programmi per HC-SR04 (DG7)

Su questo connettore viene collegato direttamente il sensore ad ultrasuoni HC-SR04, che ha un raggio di azione di un paio di metri. Emette un'onda ad ultrasuoni, che colpito un oggetto emette un eco. Un sensore lo capta, e in base al ritardo stabilisce la distanza dell'oggetto.

Nome del programma:	**Ultra_1**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**HC-SR04** , sensore ad ultrasuoni
Monitor Seriale: si (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale la distanza dell'oggetto rilevato
Note:	Le caratteristiche ufficiali forniscono una gamma di misurazione da 2 cm a 4 metri, ma con il programma attuale non si riesce a superare 1,5 m.
Link:	https://projecthub.arduino.cc/Isaac100/7cabe1ec-70a7-4bf8-a239-325b49b53cd4

Nome del programma:	**Ultra_2**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**HC-SR04** , sensore ad ultrasuoni
Porte:	**Comp. Accessori:**
D9 (digitale)	Led 3 rosso su LD3
D10 (digitale)	Led 2 blu su LD2
D13 (digitale)	Buzzer KY-006 (KY-012)
Monitor Seriale: si (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale la distanza dell'oggetto rilevato. Il led blu e una nota alta sono emesse se l'oggetto è oltre 50 centimetri; si accende il led rosso e si sente una nota media se l'oggetto è tra 50 e 20 centimetri; si accendono i due led contemporaneamente e viene emessa una nota bassa se la distanza è minore di 20 cm.
Note:	Le caratteristiche ufficiali forniscono una gamma di misurazione da 2 cm a 4 metri, ma con il programma attuale non si riesce a superare 1,5 m.
Link:	nessuno

Nome del programma:	**Ultra_3**
Porte:	**Modulo principale:**
D9, Rx; D10, Tx (digitale)	**HC-SR04** , sensore ad ultrasuoni
Porte:	**Comp. Accessori:**
D2/./D7 (digitali)	Display LCD 1602 (16 caratteri x 2 righe) su DY2
D9 (digitale)	Led 3 rosso su LD3
D10 (digitale)	Led 2 blu su LD2
D13 (digitale)	Buzzer KY-006 (KY-012) su BZ1
Monitor Seriale: si (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale la distanza dell'oggetto rilevato. Il led blu e una nota alta sono emesse se l'oggetto è oltre 50 centimetri; si accende il led rosso e si sente una nota media se l'oggetto è tra 50 e 20 centimetri; si accendono i due led contemporaneamente e viene emessa una nota bassa se la distanza è minore di 20 cm. La distanza viene mostrata sul display LCD, oltre che al tempo dell'eco in ms.
Librerie necessarie:	LiquidCrystal.h
Note:	Le caratteristiche ufficiali forniscono una gamma di misurazione da 2 cm a 4 metri, ma con il programma attuale non si riesce a superare 1,5 m.
Link:	nessuno

Programmi per cella di carico + HX711 (DG7)

Su DG7 è possibile collegare anche la cella di carico + l'amplificatore di segnale HX711, che permettono di trasformare Arduino in una perfetta bilancia digitale, in grado di pesare fino a 30 kg.

HX711 usa due porte digitali come HC-SR04 (D8 e D11), ma ha una piedinatura diversa, perciò è necessario collegarlo con un connettore a quattro fili.

La cella di carico l'amplificatore di segnale HX711 trasformano Arduino in una pesa digitale capace di pesare fino a 30 Kg.
La cella di carico restituisce alla pressione una tensione di pochi millivolt, che non sono in grado di alimentare gli ingressi digitali di Arduino. Perciò il segnale viene passato in un amplificatore composto ad un integrato HX711 che eleva la tensione al giusto livello. Il questa tabella sono indicati i collegamenti corretti.

Da cella verso	→	ingresso dell'ampl.	Da uscita dell'ampli.	→	verso Arduino DG7
filo rosso	→	E+	GND	→	–
filo nero	→	E-	DT	→	Echo (data)
filo bianco	→	A-	SCK	→	Trig (clock)
filo verde	→	A+	VCC	→	+5 v

Ogni cella di carico ha delle tolleranze di produzione, per cui va calibrata prima di pesare correttamente. Un primo programma (calib_grezza) dà una prima informazione più grossolana; una seconda (calib_fine) fornisce un'informazione più precisa. L'ultimo programma è una pesa digitale, che presenta le informazioni sul display LCD.

Nome del programma: **Calib_grezza**
Porte: **Modulo principale:**
D8 - clk (digitale) Cella di carico + amplificatore HX711
D11 - data (digitale)
Monitor Seriale: si **Plotter seriale: no**
(9600 baud)
Scopo del programma: Inserire sulla cella di carico un oggetto con un peso noto (nel programma è di 150 gr.; modificarlo in base alle proprie necessità). Iniziare digitando un tasto nel monitor seriale e segnarsi il valore ottenuto.
Librerie necessarie: HX711.h
Note: nessuna
Link: https://www.youtube.com/watch?v=zdPSOnwjCOM

Nome del programma: **Calib_fine**
Porte: **Modulo principale:**
D8 - clk (digitale) Cella di carico + amplificatore HX711
D11 - data (digitale)
Monitor Seriale: si **Plotter seriale: no**
(9600 baud)
Scopo del programma: Inserire in "float calibration_factor" il valore rilevato dal programma precedente, poi lanciare il programma (vedi immagine). Pesare l'oggetto usato in precedenza, e controllare sul visore il peso rilevato. Con "+" o "a" si può aumentare il fattore; con "-" o "z" si può diminuire. Agire dino a quando si è trovato il valore corretto e segnarlo.
Note: nessuna
Librerie necessarie: HX711.h
Link: https://www.youtube.com/watch?v=jqLhLg5WhmQ
https://randomnerdtutorials.com/arduino-load-cell-hx711/

calib_fine

```
#include "HX711.h"
const int LOADCELL_DOUT_PIN = 11;
const int LOADCELL_SCK_PIN = 8;
HX711 scale;

float calibration_factor = 1.82; // this calibration factor is adjusted according to my load cel
```

L'informazione che si ricava da "calib_fine",
e che va inserita nel programma definitivo, chiamato "pesa"

Nome del programma:	**Pesa**
Porte:	**Modulo principale:**
D8 - clk (digitale) D11 - data (digitale)	Cella di carico + amplificatore HX711
Porte:	**Comp. Accessori:**
D2/./D7 (digitali)	Display LCD 1602 (16 caratteri x 2 linee) su DY2
Monitor Seriale: si (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Inserire in "float calibration_factor" il valore rilevato dal programma precedente, poi lanciare il programma. Pesare l'oggetto usato in precedenza, e controllare sul visore il peso rilevato. Se non ci fosse ancora una perfetta corrispondenza, modificare ancora leggermente il valore. Pesare oggetti con massa diversa, per verificare la precisione dello strumento. Per azzerare la tara, premere il tasto "t" o "T". Il peso apparirà sul display a cristalli liquidi.
Librerie necessarie:	HX711.h; LiquidCrystal.h
Note:	https://www.youtube.com/watch?v=izTqR7onqpI&list=PL9_01HM23dGEDNNfR6BtlDWD8DDoAcLOT&index=217
Link:	nessuno

Il risultato ottenuto con il programma "pesa"; naturalmente si può perfezionare migliorando la calibratura fine del programma

Immagine della cella di carico con la base e il piatto eseguiti con una stampante 3D

Clicca qui per scaricare i file della base inferiore, quella superiore e la vaschetta della pesa per la stampante 3D

I programmi per il ricevitore a 433 MHz (Rx) -DG7

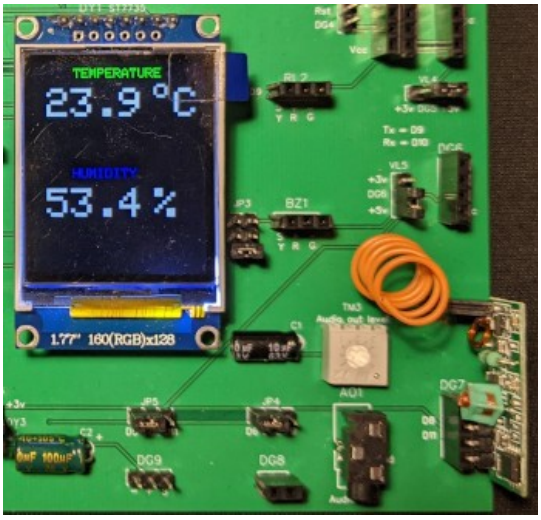
A differenza di tutti gli altri sensori utilizzati in questi programmi, il ricevitore a 433 MHz non funziona da solo (o meglio, funziona, ma non riceve nulla), ma richiede di essere accoppiato a trasmettitore a 433 MHz. Per cui, a ogni programma che trasmette qualche dato, ce ne sarà uno gemello che li riceve e li rende usufruibili.
Naturalmente perché il sistema funzioni, sono necessari due Arduino, uno connesso al trasmettitore e uno al ricevitore

Ecco i programmi relativi al ricevitore, presenti su DG7:

Nome del programma:	**433_hello_rx**
Porte:	**Modulo principale:**
D11 (digitale)	**Rx 433 MHz**
Monitor Seriale: sì (9600 bps)	**Plotter seriale: no**
Scopo del programma:	Questo programma riceve il solito messaggio "hello world", che verrà trasmesso dal corrispondente ***433_hello_tx***. In sé non ha grande utilità, se non quello di verificare che il trasmettitore e il ricevitore comunichino.
Librerie richieste:	RH_ASK.h
Note:	nessuno
Link:	https://lastminuteengineers.com/433mhz-rf-wireless-arduino-tutorial/

Nome del programma:	**433_led_rx**
Porte:	**Modulo principale:**
D11 (digitale)	**Tx 433 MHz**
Monitor Seriale: no	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9 (digitale)	Led rosso su LD3
Scopo del programma:	Questo programma è anch'esso semplicemente didattico. Il trasmettitore, invia il segnale per far lampeggiare all'unisono un led sia sul trasmettitore che sul ricevitore Programma: ***433_led_tx***. In sé non ha grande utilità, se non quello di verificare che il trasmettitore e il ricevitore comunichino.
Librerie richieste:	VirtualWire.h
Note:	nessuno
Link:	https://techatronic.com/rf-transmitter-and-receiver-with-arduino/

Nome del programma:	**433_temp_rx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D3/./D7 (digitali)	Display TFT 1.77" ST7735 su DY1
Scopo del programma:	Il sensore DHT11, che legge la temperatura e l'umidità ambientale, è collegato al trasmettitore, che invia queste informazioni al ricevitore, attraverso al programma ***433_temp_tx***. Il ricevitore mostra questi dati su di un display TFT a colori.
Librerie richieste:	VirtualWire.h; Adafruit_ST7735.h; Adafruit_GFX.h
Note:	nessuno
Link:	https://www.electronics-lab.com/project/using-433mhz-rf-transmitter-receiver-arduino/



La temperatura e l'umidità mostrati sul display dal programma 433_temp_rx/433_temp_tx. Sulla destra, in basso, si vede il modulo del ricevitore a 433 MHz e la sua antenna.

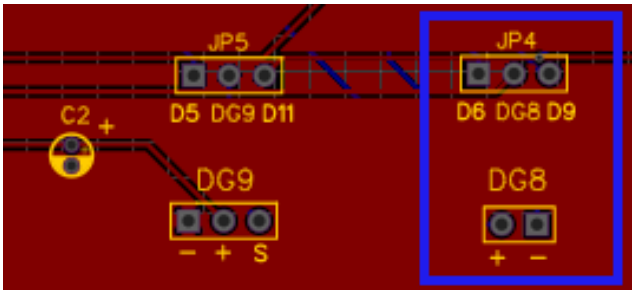
Nome del programma:	**433_button_rx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9	Led rosso su LD3
D16 (fisico: A2)	Pulsante su BT1
Scopo del programma:	Premendo il pulsante BT1, si accende il led sul trasmettitore,che invia il segnale al ricevitore, attraverso al programma ***433_button_tx***, che fa accendere in sincrono il led presente sul modulo rx.
Librerie richieste:	VirtualWire.h
Note:	nessuno
Link:	http://www.brescianet.com/appunti/Elettronica/Arduino/corso/Esempio_RF315-433MHZ.htm

Nome del programma:	**433_3button_rx**
Porte:	**Modulo principale:**
D12 (digitale)	**Tx 433 MHz**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D9	Led rosso su LD3
D10	Led blu su LD2
D7	Led verde sul connettore DY1. Collegare il led attraverso una resistenza da 220 Ohm.
A1 (analogico)	Pulsante KY-004 su AN1
Scopo del programma:	Questo programma diventa un un telecomando a tre canali, che farà accendere/spegnere indipendentemente 3 led sul ricevitore. Un buzzer in base ai bip emesse, indica quanti led sono attivi. Programma: ***433_3button_tx.***
Librerie richieste:	RH_ASK.h; SPI.h
Note:	nessuno
Link:	https://mariodenichilo.altervista.org/arduino-trasmissione-senza-fili-a-433-mhz-radiocomando-a-3-canali

Clicca qui per accedere ai seguenti programmi del ricevitore, collegato su DG1:

- 433_hello_rx
- 433_led_rx
- 433_button_rx
- 433_3button_rx
- 433_temp_rx

Lo zoccolo DG8



Lo zoccolo DG8 è posto in basso a destra della *bs*.
E' controllato da un transistor Mos-fet (MF1) e protetto dai diodi DD2 e DD3.
Affinché funzioni, è necessario che Arduino sia alimentato esternamente (non da USB).
Questo zoccolo usa la porta D6 di default, quindi non è compatibile con i display DY1 e DY2, ma può essere gestito anche dalla porta digitale D9, modificando opportunamente il programma.

DG8. Questo zoccolo può alimentare un generico motore elettrico, che funzioni con una tensione compresa tra 6 e 9 v. Utilizzando la porta D6, che è una PWM, è possibile variare la velocità del motore stesso, per esempio collocando un potenziometro sullo zoccolo analogico A2. Seguono alcune informazioni relative al funzionamento di questo sistema, abbastanza particolare perché un motore elettrico assorbe una corrente superiore a quella che potrebbe fornire Arduino attraverso la porta USB. Perciò deve essere alimentato da un alimentatore esterno (vedi la sezione di **alimentazione di Arduino**, opzione 2 o 3). Il Mos-fet denominato MF1 fa da ponte tra il microcontroller e l'alimentazione del motore; D6 (e D9) sono porte PWM, quindi possono variare la tensione, e di conseguenza la velocità di rotazione dell'albero del motore. Il diodo D3 serve per evitare che una corrente inversa, prodotta dal motore quando ruota ancora qualche attimo dopo che sia stata tolta la tensione, danneggi MF1. Ricordarsi di collegare la bs ***a un alimentatore esterno***, in grado di fornire una tensione compresa tra 6 e 9v, altrimenti non arriverà la corrente al motore. Utilizzando la porta digitale D6, non è possibile utilizzare contestualmente i display collegati su DY1 e DY2, perché usano anche questa porta. Nel caso si desiderasse visualizzare alcune informazioni, si potrà utilizzare il dispaly Oled da collegare sullo zoccolo DY3, perché utilizza le porte analogiche A4 e A5.

Un motore elettrico

Nota: spostando il ponticello presente su JP4, può essere gestito dalla porta digitale D9 (anch'essa PWM), modificando opportunamente il programma. Quindi diventa compatibile con i display che alloggiano su DY1 e DY2.

Lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG8 (digital- D6)			1	2
Description	Code	Note	Signal	GND
Motor	Generic 6/./9 V motor	Richiede un'alimentazione esterna	d6 (d9)	-

Programmi per motore elettrico generico

Questo connettore è specifico per controllare dei motori elettrici che funzionano con una tensione compresa tra 6 e 9 volt. Poiché i motori elettrici richiedono una corrente che non è fornibile da Arduino, i motori funzioneranno solamente se alla basetta è collegato un alimentatore esterno o sull'ingresso "ext. Power" oppure alla "Power unit" siglata "Power MB V2".

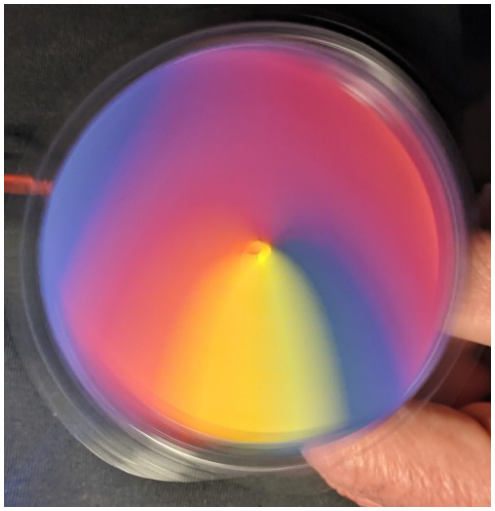
Nome del programma:	**Motor_1**
Porte:	**Modulo principale:**
D6	Motore elettrico
Porte:	**Comp. Accessori:**
A2 (analogico)	BT1 – pulsante presente sulla basetta
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Il motore si avvia quando il pulsante viene premuto.
Note:	I motori funzioneranno solamente se alla basetta è collegato un alimentatore esterno o sull'ingresso "ext. Power" oppure alla "Power unit" siglata "Power MB V2".
Link:	Nessuno

Nome del programma:	**Motor_2**
Porte:	**Modulo principale:**
D6	Motore elettrico
Porte:	**Comp. Accessori:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	I giri del motore aumentano progressivamente, in base alla rotazione della manopola del potenziometro.
Note:	I motori funzioneranno solamente se alla basetta è collegato un alimentatore esterno o sull'ingresso "ext. Power" oppure alla "Power unit" siglata "Power MB V2".
Link:	nessuno

Nota: **motor_2a** ha le stesse caratteristiche di Motor_2, semplicemente il potenziometro funziona in modo inverso.

Nome del programma:	**Motor_3**
Porte:	**Modulo principale:**
D6	Motore elettrico
Porte:	**Comp. Accessori:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
D11 (digitale)	Led L1 - verde
D10 (digitale)	Led L2 - blu
D9 (digitale)	Led L3 - rosso
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	I giri del motore aumentano progressivamente, in base alla rotazione della manopola del potenziometro. I led si accendono progressivamente, sempre relativamente alla rotazione della manopola del potenziometro.
Note:	I motori funzioneranno solamente se alla basetta è collegato un alimentatore esterno o sull'ingresso "ext. Power" oppure alla "Power unit" siglata "Power MB V2".
Link:	nessuno

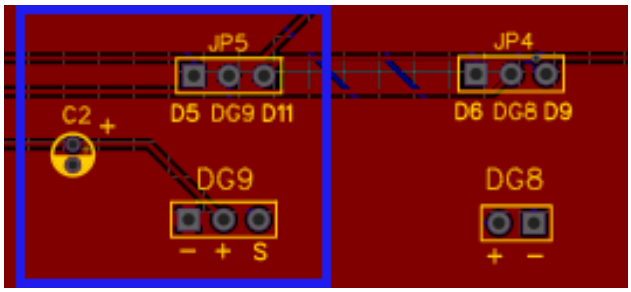
motore fermo



motore in movimento

Nome del programma:	**Motor_3a**
Porte:	**Modulo principale:**
D6	Motore elettrico
Porte:	**Comp. Accessori:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
D11 (digitale)	Led L1 - verde
D10 (digitale)	Led L2 - blu
D9 (digitale)	Led L3 - rosso
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Uguale al precedente. Reagisce al contrario alla rotazione del potenziometro.
Note:	Idem come il programma "Motor_3".
Link:	nessuno

Lo zoccolo DG9



Lo zoccolo DG9 può gestire un servomotore. E' l'unico zoccolo con connettori maschi, tutti gli altri sono femmine.
I servomotori, a differenza dei motori, assorbono bassissime correnti, quindi possono essere alimentati anche tramite la USB di Arduino.
Questo zoccolo usa la porta D5, quindi non è compatibile con i display DY1 e DY2; tuttavia può anche essere gestito dalla porta digitale D11, eliminando questa limitazione.

DG9. Si rimane nell'ambito degli attuatori, in questo caso di un servomotore. La differenza tra un semplice motore e un servomotore è che in quest'ultimo, il perno non gira continuamente, ma in genere ruota con un raggio di azione di 180° (mezzo giro), più raramente di 360° (un giro completo). Può essere controllato in modo molto preciso e puntuale, infatti i "servo" sono utilizzati dove sono necessari degli spostamenti angolari precisi, e questo è possibile perché la porta digitale D5 è PWM, e quindi può passare delle precise informazioni al servo per farlo ruotare dell'angolo richiesto. Rispetto ai motori, i servo assorbono una bassa corrente, per cui abitualmente possono essere alimentati direttamente da Arduino. DG9 ha una particolarità, infatti è l'unico zoccolo con connettori maschi, in quanto i servomotori che hanno tre fili (due di alimentazione, uno di segnale), terminano abitualmente con un connettore femmina.

Un servomotore.

Utilizzando la porta digitale D6, non è possibile utilizzare contestualmente i display collegati su DY1 e DY2, perché usano anche questa porta. Nel caso si desiderasse visualizzare alcune informazioni, si potrà utilizzare il dispaly Oled da collegare sullo zoccolo DY3, perché utilizza le porte analogiche A4 e A5.

Nota1: spostando il ponticello presente su JP5, può essere gestito dalla porta digitale D11 (anch'essa PWM), modificando opportunamente il programma. Quindi diventa compatibile con i display che alloggiano su DY1 e DY2.

Lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG9 (digital – D5)			1	2	3
Description	Code	Note	GND	vcc (+5v)	Signal
Servomotor	MG90S		-	+	d5 (d11)

Nota2: nel riquadro è stato inserito il codice del servomotore MG90S perché è quello che è stato testato, ma ne possono essere montati anche altri, purché abbiano le stesse caratteristiche di tensione e assorbimento di corrente. In questo caso, verificare che la disposizione dei piedini sia la stessa di quella mostrata, altrimenti usare un connettore a tre cavi, da collegare nella sequenza corretta.

Programmi per servomotore MG90S

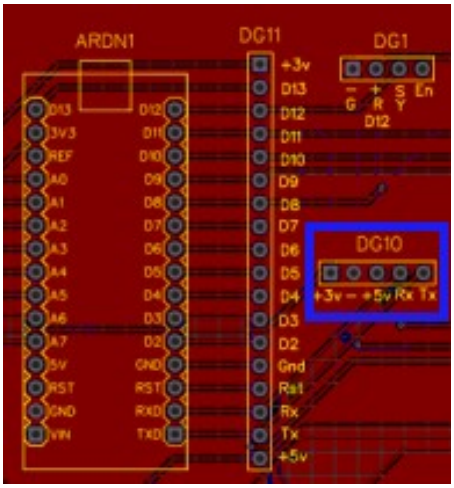
Questo connettore è stato esplicitamente progettato per controllare i servomotori a tre fili. A differenza dei motori i servo consumano poca corrente, quindi possono essere alimentati direttamente da Arduino, oppure, per un uso più intenso , attraverso la “Poewr Unit”. I servomotori, in genere, permettono una rotazione di circa 180°.
Utile per ruotare bracci o creare servomeccanismi, o anche per chiudere/aprire una porta o una serratura.

Nome del programma:	**Servomotor_1**
Porte:	**Modulo principale:**
D5	Servomotore di tipo MG90S
Porte:	**Comp. Accessori:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Girando la manopola in senso orario, l’asse del motore ruota da 0° a 180° (circa); ruotando la manopola in senso inverso, l’asse ruota da 180° a 0°.
Librerie richieste:	Servo.h
Note:	Attenzione a rispettare la polarità del connettore (in genere il cavo giallo è quello di segnale).
Link:	nessuno

Nota: **servomotor_1a** la differenza con servomotor_1 consiste semplicemente nel senso di rotazione del potenziometro.

Nome del programma:	**Servomotor_2**
Porte:	**Modulo principale:**
D5	Servomotore di tipo MG90S
Porte:	**Comp. Accessori:**
A2	BT1 – pulsante presente sulla basetta
D11 (digitale)	Led L1 - verde
D9 (digitale)	Led L3 - rosso
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Premendo il pulsante, l’asse del motore ruota da 0° a 180° (circa); il led verde si spegne e si accende quello rosso, e poi ritorna in posizione 0. Il led rosso si spegne e si riaccende quello blu.
Librerie richieste:	Servo.h
Note:	Attenzione a rispettare la polarità del connettore (in genere il cavo giallo è quello di segnale).
Link:	nessuno

Lo zoccolo DG10



DG10 si trova sulla destra di DG11 e di Arduino Nano. E' un connettore particolare, un po' difficile da programmare. Quindi lo si usa solo in caso di stretta necessità.

Infatti usa le porte Rx e Tx di Arduino, le stesse usate da Arduino per comunicare con il computer, limitando l'uso di alcune funzioni molto utili, quali il monitor seriale durante il test dei programmi.

Il modulo dovrebbe essere inserito nel connettore solo dopo aver caricato lo sketch, altrimenti si riceverà un messaggio di errore.

DG10. DG10 è un connettore particolare, che si trova a destra di Arduino, di lato a DG11.perché è connesso ai pin digitali TX1 e RX0, ovvero quelli che Arduino usa per effettuare le connessioni seriali da e per il computer, per cui non si può connettere il modulo allo zoccolo quando si deve passare il programma al microcontroller, pena un errore di comunicazione. Perciò *prima* si carica il programma, *poi* si connette il modulo.
E' sempre buona norma collegare i moduli alla bs quando quest'ultima non è alimentata.
Questo comporta due aspetti particolari:

- poiché bisogna scollegare il computer, Arduino deve essere alimentato esternamente;
- non si può usare il monitor seriale. Questa è una limitazione non da poco, in quanto attraverso il monitor si hanno molte informazioni sull'andamento del programma; potrebbe quindi essere necessario configurare un display per poter ottenere le informazioni necessarie.

Queste riflessioni, oltre alla difficoltà intrinseca nell'utilizzare le porte relative allo zoccolo DG10, consigliano di usarlo solo in caso di vera necessità.

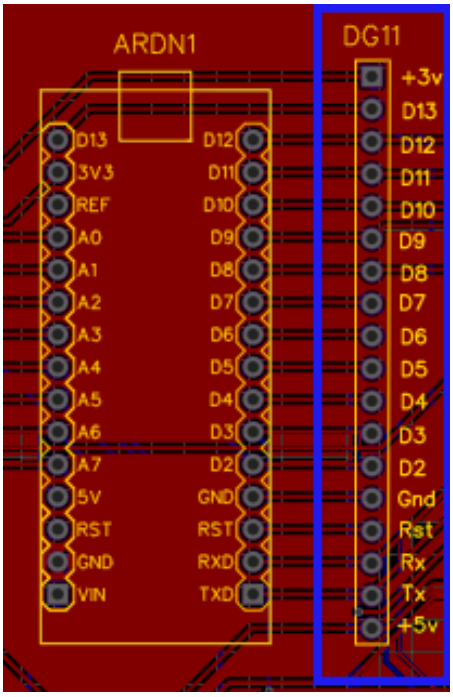
Lista dei moduli che si possono collegare direttamente a questo zoccolo:

Socket DG10 (digital) multipurpose			1	2	3	4	5
Description	Code	Note	Vcc (+3,3v)	vcc (+5v)	GND	RX (d1)	TX (d0)
		Inserire il sensore solo dopo aver caricato il programma. Usare un connettore.					

Al momento non ci sono programmi disponibili.

Nota: la lista dei moduli che possono utilizzare questo zoccolo sono in teoria tutti quelli che usano una o due porte digitali; tuttavia non avendo fatto dei test approfonditi, per il momento la tabella è stata intenzionalmente lasciata vuota.

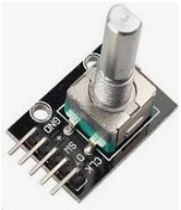
Lo zoccolo DG11



Il connettore DG11 è un connettore "universale", in quanto riassume in sé i contatti di tutte le porte digitali.

E' posto immediatamente a destra di Arduino.

Ovviamente non si possono usare per altri zoccoli le porte utilizzate dai moduli collegati direttamente ad esso.



Encoder KY-040

DG11. DG11 è un connettore "universale", per quel che riguarda le porte digitali. Infatti raccoglie, in ordine, tutte le porte digitali da "Tx" a "D13", oltre che le tensioni "+5v", "+3,3v" e la massa (GND). Può essere utilizzata per tutti i moduli che utilizzano porte digitaliche non sono stati previsti nella *bs*, qualora non si possa "adattare" uno zoccolo esistente, o quando lo zoccolo prescelto fosse già utilizzato da un altro modulo, oppure infine quando si tratta di moduli complessi , che usino contemporaneamente più di due porte digitali, come un encoder (KY-040), oppure di un sensore RFID, o ancora il display a quattro cifre 5641AS, a cui è stata dedicata una basetta personalizzata, che con l'uso di un integrato permette di usare solo tre porte digitali invece che otto!

RFID

Moduli che usano il connettore DG11:

Socket DG11 (digital) multipurpose			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Description	Code	Note	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	-	RST	RX0	TX1	+3v	+5v
Keypad module 4x4 *		Usare connettore – diversa piedinatura		☑	☑	☑		☑												
Keypad module 4x3 *		Usare connettore – diversa piedinatura		☑	☑	☑		☑												
Lettore Rfid	MF522	Usare connettore – diversa piedinatura	☑	☑	☑	☑				☑					☑				☑	
Four digit display	5641S + 7HC595	Usare connettore – diversa piedinatura		☑	☑	☑									☑				☑	
433 MHz receiver (RX) **														☑						
Encoder	KY-040 – HW040	Usare connettore – diversa piedinatura										☑	☑	☑	☑					☑
Stepper motor + ULC2003		Usare connettore – diversa piedinatura			☑	☑	☑	☑												

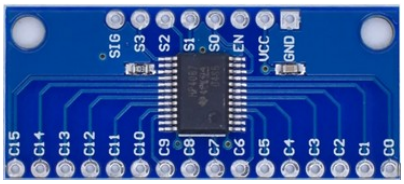
* **Nota :** La tastiera 4x4 usa otto porte di cui 4 digitali, mentre quella 4x3 ne usa sette. In questa tabella abbiamo utilizzato anche delle porte analogiche, per ottimizzare l'utilizzo delle porte stesse. Naturalmente nulla vieta di modificare i programmi, utilizzando solo porte digitali.

** **Nota 2**: Il ricevitore a 433 MHz in genere usa la porta D11, ma per un progetto la libreria utilizzata chiede espressamente la porta D2.

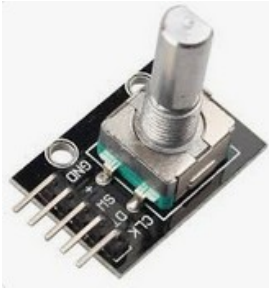
Immagini dei moduli *particolari* che alloggiano si DG11:

433 Mhz receiver (RX)

4 digit mod. 5641AS



Multiplexer hw-178



Encoder KY-040

Keypad 3x4

Keypad 4x4

Stepper motor

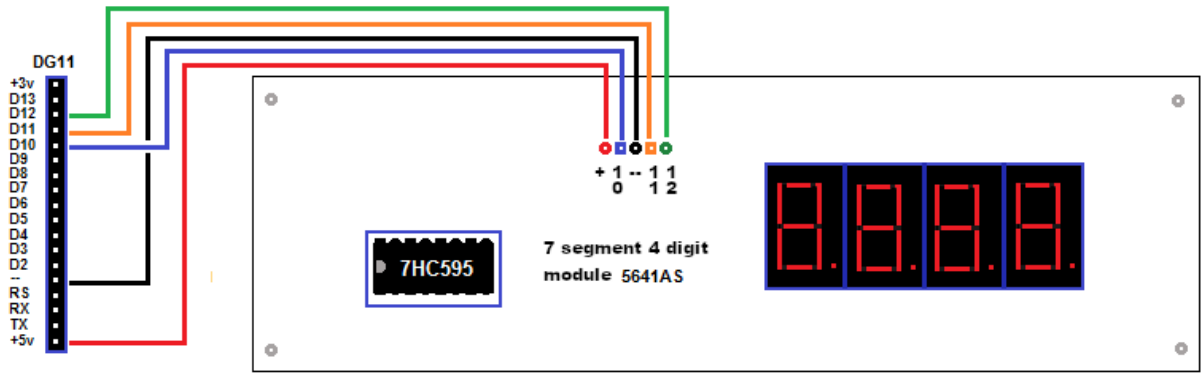
SD card reader

Approfondimento relativo ai moduli che alloggiano su DG11:

Display a 4 cifre 5641AS con integrato 7HC595

Il display a quattro cifre 5641AS restituisce dei caratteri alfanumerici di grandi dimensioni in colore rosso con una buona luminosità, quindi adatto, per esempio, a un orologio digitale, o in apparecchi che richiedano un numero limitato di caratteri ma ben visibili.
Questo display presenta un problema: richiede ben otto porte digitali per essere pilotato, quindi lascia poco spazio per altri moduli. Pertanto in questo progetto è stato gestito per mezzo di un integrato operazionale 74HC595, che riduce il numero di porte digitali necessarie e tre: le porte D10, D11 e D12. Lo schema di questo interessante progetto si trova nella sezione delle **appendici.**

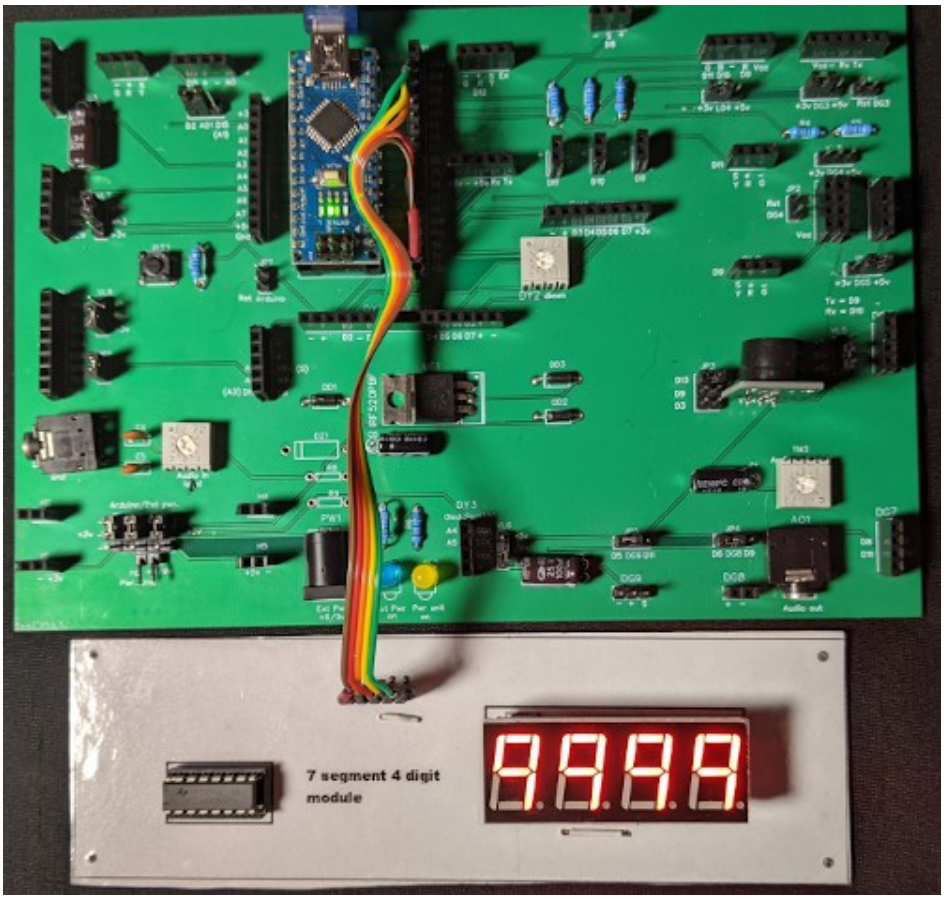
Modificando opportunamente il programma, utilizzando per esempio le porte (D15), D(14), (D)17, potremmo collegarlo anche alle prime tre porte (originariamente analogiche) di AD3



Schema del collegamento del display a 4 cifre 5641AS + ic 7HC595

Clicca qui per andare alla pagina del progetto.

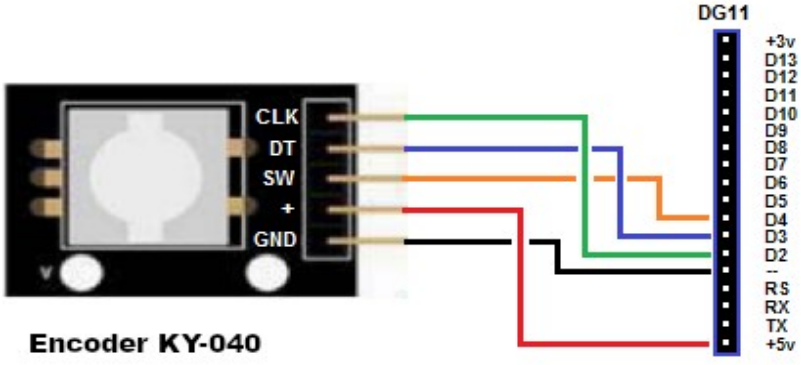
Nome del programma:	**5641_1**
Porte:	**Modulo principale:**
D10, D11, D12	Dislay a quattro cifre 5641AS + IC 74HC595
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	Questo programma dimostrativo mostra in sequenza i numeri in esadecimale, da “1” a “F”
Note	Attenzione a rispettare la polarità del connettore
Link:	nessuno



Il modulo a quattro cifre collegato ad Arduino con un integrato 74HC595 che permette di usare solo tre porte digitali, invece delle otto che sarebbero necessarie collegandolo direttamente.

Encoder KY-040

L'encoder KY-040 permette di misurare con molta precisione uno spostamento, per cui potrebbe essere utile nel misurare lo spostamento di un carrello, per esempio come quello utilizzato sull'asse X, Y o Z di una stampante 3D autocostruita, un robot, ecc. Questo encoder usa 3 porte digitali: D2, D3, D4, pertanto non è compatibile con i display TFT e LCD, che usano rispettivamente gli zoccoli DY1 e DY2. Si potrà usare per mostrare le informazioni i display OLED 128x32 o 128x64 che si collegano allo zoccolo DY3, perché usano delle porte analogiche.
Naturalmente nulla vieta di usare un'altra serie di porte digitali, modificando il programma per renderlo compatibile con i display eventualmente presenti su DY1 o DY2.



Encoder KY-040

Ecco lo schema di collegamento dell'encoder:

Programmi per Encoder KY-40

Nome del programma:	**Encoder_1**
Porte:	**Modulo principale:**
D2, D3. D4 (digitali)	**Rotary encoder KY-040**
Monitor Seriale: sì	**Plotter seriale: no**
(9600 bit)	
Scopo del programma:	Girando in senso orario il perno dell'encoder, si vedranno i valori incrementare sul monitor seriale. Ruotando in senso antiorario, diminuiscono. Premendo sul perno, i valori si azzerano.
Note:	nessuna
Link:	**nessuno**

Nome del programma:	**Encoder_2**
Porte:	**Modulo principale:**
D2, D3. D4 (digitali)	**Rotary encoder KY-040**
Porte:	**Comp. Accessori:**
D9	Led L3 rosso su LD3
D10	Led L2 blu su LD2
Monitor Seriale: sì	**Plotter seriale: no**
(9600 bit)	
Scopo del programma:	Girando n senso orario il perno dell'encoder, si vedranno i valori incrementare sul monitor seriale e il led blu aumenta progressivamente di luminosità. Ruotando in senso antiorario, i valori diminuiscono e la luce del led blu si affievolisce. Se i valori vanno in negativo, progressivamente si accende il led rosso; se tornano verso lo zero, si affievolisce. Premendo sul perno, i valori si azzerano.
Note:	nessuna
Link:	Nessuno

Nome del programma:	**Encoder_3**
Porte:	**Modulo principale:**
D D2, D3. D4 (digitali)	**Rotary encoder KY-040**
Porte:	**Comp. Accessori:**
A4 (analogiche)	Dispaly LCD 1602 + adattatore I2C su zoccolo AN3
A5	
D5	Servomotore MG90S su zoccolo DG9
Monitor Seriale: sì	**Plotter seriale: no**
(9600 bit)	
Scopo del programma:	IL servomotore all'avvio si posiziona con l'asse a 90°. Girando n senso orario il perno dell'encoder, si incrementa l'angolo di rotazione del servo, fino a 180°. Ruotando in senso antiorario, si decrementa l'angolo di rotazione del servomotore fino a 0°. Sul display LCD appaiono i dati angolari.
Librerie richieste:	LiquidCrystal_I2C.h; Servo.h
Note:	nessuna
Link:	https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

Una tastiera flessibile da 12 o 16 tasti

Può essere molto utile collegare una tastiera ad Arduino, per esempio per aprire una porta con una combinazione, magari abbinandola, per aumentare la sicurezza, alla lettura di una scheda RFID.

La tastiera da 12 tasti usa una matrice di 4x3 (quattro righe, tre linee orizzontali), per un totale di 7 cavi di uscita, da collegare ad altrettante porte digitali (o in parte digitali e altre analogiche), mentre la tastiera da 16 tasti usa una matrice di 4x4 (quattro righe, quattro linee orizzontali), per un totale di 8 cavi di uscita, da collegare ad altrettante porte digitali, o anche in questo caso, utilizzando in parte alcune porte digitali e altre analogiche).

Le tastiere flessibili digitali da 12 o da 16 tasti, usano rispettivamente una matrice di 4x3 o di 4x4 ed usano 7 cavi (12 tasti) o 8 cavi (16 tasti).

Perciò sono piuttosto voraci di porte digitali e bisogna ottimizzarle al meglio per poterli inserire in progetti che abbiano una certa utilità, e non solo didattici.
Si possono collegare con dei cavetti (o meglio ancora, una piattina) allo zoccolo DG11, ma con alcuni accorgimenti per bilanciare i carichi, possono essere utilizzate anche alcune porte del lo zoccolo AN5.

Con la nuova scheda, alcuni progetti sono stati spostati su di uno zoccolo dedicato, AD3
Clicca qui per visualizzarli.

Programmi per keypad

Anche le keypad sono “voraci” di porte: 7 per la tastiera a 12 tasti e 8 per quella a 16 tasti. Per non occupare troppe porte digitali, ne sono state utilizzate 4 digitali e 3 o 4 (in base al tipo di tastiera) analogiche.
Per cui per questi programmi si è usato sia lo zoccolo DG11 che AN5.

Alcuni programmi per le tastiere sono stati spostati su di uno zoccolo dedicato. AD3.
Clicca qui per visualizzarli.

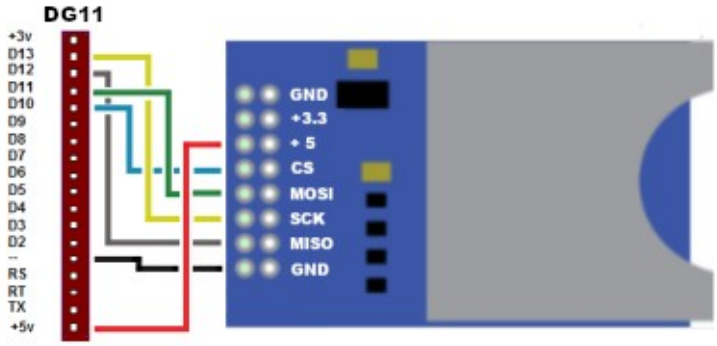
Attenzione: questa configurazione delle porte analogiche e digitali è da usare SOLO per il progetto Key_16_2a, che utilizza il display LCD 1602 posto sullo zoccolo DY2, che utilizza 16 pin invece dei 4 necessari per il display con protocollo I2C, e quindi richiede le porte digitali da D2 a D7.

Le differenze sono indicate in rosso scuro.

Nome del programma:	**Key16_2a**
Porte:	**Modulo principale:**
D8, D10, D11, D12 (digitali) A0, A1, A2, A3 (analogiche)	**Tastiera a 16 cifre** (8 connettori)
Porte:	**Comp. Accessori:**
D2/./D7	Display LCD1602 su DY2
Monitor Seriale: sì	**Plotter seriale: no**
Scopo del programma:	Questo programma dimostrativo mostra sul monitor seriale e sul display LCD 1602 con protocollo I2C il valore dei tasti premuti, che va da “0” a “9”, più “*”, “#”, e “A”, “B”, “C”, “D”.
Librerie richieste:	Keypad.h, LiquidCrystal.h
Note:	Attenzione a rispettare la corretta sequenza delle connessioni su Arduino Questo programma usa porte leggermente diverse dagli altri, perché adotta il display LCD 1602 senza l’adattatore I2C, quindi usa per sé ben sei porte digitali.
Link:	nessuno

Una SD card reader per Arduino

Un lettore di schede SD può essere utile; con alcuni programmi è possibile verificare che la scheda SD sia compatibile con il lettore, avere una lista dei file salvati su di essa e di poter scrivere e/o leggere direttamente un file. Nell’immagine qui di seguito, sono indicati i pin di collegamento.



Schema di collegamento della SD reader

Schema di collegamento

Mosi	Miso	SCK	CS
11	12	13	10

Il lettore di schede SD può essere alimentato indifferentemente a 5v o a 3,3v.
Attenzione però di collegarlo al pin corretto!

Il lettore di SD card (ormai è quasi d'obbligo dotarlo di un adattatore per le micro SD card), può essere utile per sapere se una scheda e compatibile con Arduino, e in un secondo tempo per leggere/scrivere delle informazioni.

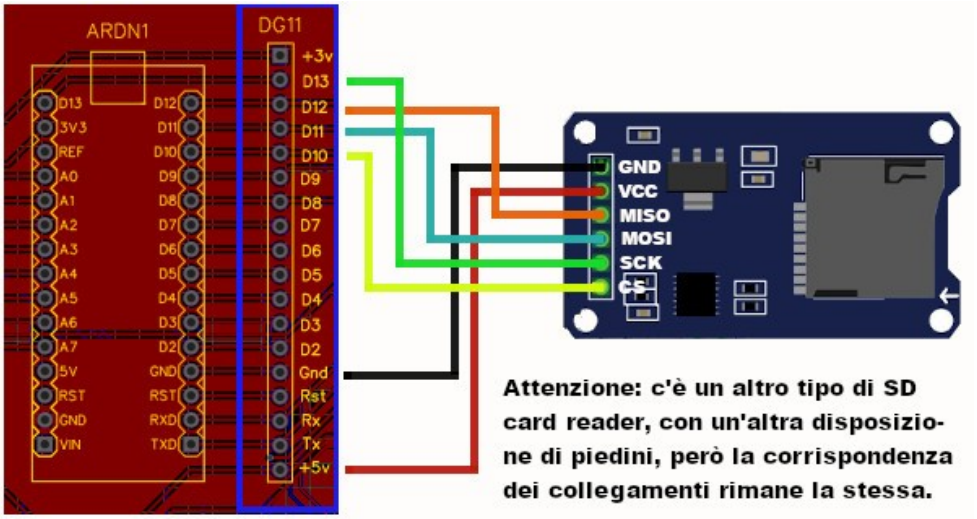
Ci sono due tipo di card reader: quello illustrato nella foto qui di fianco e quello dello schema sottostante. Come si vede, il primo ha otto piedini, disposti su due file; il secondo ha solo sei pin, su di un'unica fila. Ma le connessioni del modulo a DG11 sono gli stessi (vedi schema dei collegamenti). Infatti l'unica differenza tra i due è che quello a otto pin ne ha due per il collegamento a massa /GND) e la doppia alimentazione, a 3,3v o a 5v; mentre quello a sei pin ne ha solo uno relativo alla massa e una singola alimentazione a 5v.

Un grande vantaggio del modulo è che può essere utilizzato con la libreria SD fornita con l'IDE Arduino. La libreria SD semplifica l'inizializzazione, la lettura e la scrittura della scheda. Pertanto è necessario caricare la libreria **SPI e SD** prima di trasferire il programma su Arduino Nano.
Il modulo viene fornito con un regolatore di tensione. Pertanto, sia i pin 5V e 3.3V dell'Arduino possono essere utilizzati per l'alimentazione. Nello schema che segue, la SD card reader viene alimentata a 5V.

Ecco la tabella dei collegamenti:

SD card reader	→	**DG11**
GND		Gnd
VCC		+ 5v
CS		10
MOSI		11
MISO		12
SCK		13

Il lettore di CD car che ha otto pin invece di sei, ha in più un secondo collegamento di massa (GND) e un alimentazione a +3,3 v, tutto sommato superflua nel nostro caso, ma necessaria su microcontroller che funzionano esclusivamente a + 3,3v.



I programmi per il lettore di SD card:

I programmi che seguono sono reperibili cliccando sulla IDE di Arduino, su "***esempi/SD/...***".

Nome del programma:	**Cardinfo**
Porte:	**Modulo principale:**
D10, D11, D12, D13 (digitali)	Il modulo SD card reader
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale i dati relativi alla SD inserita nel card reader.
Librerie richieste:	SPI.h; SD.h
Note:	nessuna
Link:	https://arduinofacile.altervista.org/progetti/moduli/micro-sd-card-con-arduino/

Se la scheda SD è supportata, correttamente formattata e i collegamenti sono esatti, si otterrà sul monitor digitale un output del genere:

```
Total Blocks:      245312

Volume type is:    FAT16
Volume size (Kb):  122656
Volume size (Mb):  119
Volume size (Gb):  0.12

Files found on the card (name, date and size in bytes):
DCIM/          2009-08-22 16:20:44
  140SSCAM/      2009-08-22 16:20:44
    SDC18034.JPG  2009-08-22 16:20:46 1448170
DATABASE/      2009-08-24 17:22:26
  ACE.LOG        2009-08-24 17:22:36 131072
  ACE.DAT        2009-08-24 17:22:36 288768
TEST.TXT       2000-01-01 01:00:00 36
```

Nome del programma:	**Listfiles**
Porte:	**Modulo principale:**
D10, D11, D12, D13 (digitali)	Il modulo SD card reader
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale i file salvati sulla SD card.
Librerie richieste:	SPI.h; SD.h
Note:	nessuna
Link:	https://arduinofacile.altervista.org/progetti/moduli/micro-sd-card-con-arduino/

Ecco l'output del programma listfiles:

```
Initializing SD card...initialization done.
DCIM/
        140SSCAM/
?Initializing SD card...initialization done.
DCIM/
        140SSCAM/
                SDC18034.JPG            1448170
DATABASE/
        ACE.LOG         131072
        ACE.DAT         288768
TEST.TXT                36
done!
```

Il programma seguente scrive un file "text.txt", poi lo apre in lettura e stampa il contenuto del file stesso, ovvero "testing 1,2,3".

Nome del programma:	**Write_read**
Porte:	**Modulo principale:**
D10, D11, D12, D13 (digitali)	Il modulo SD card reader
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Il programma seguente scrive un file "text.txt", poi lo apre in lettura e stampa il contenuto del file stesso.
Librerie richieste:	SPI.h; SD.h
Note:	nessuna
Link:	https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/

Ecco ciò che appare sul monitor seriale:

```
Initializing SD card...initialization done.
Writing to test.txt...done.
test.txt:
testing 1, 2, 3.
testing 1, 2, 3.
testing 1, 2, 3.
```

Il file è stato scritto e riletto!

Un ricevitore a 433 MHz

Schema di collegamento del ricevitore a 433 MHz

Il ricevitore a 433 MHz abitualmente usa la porta digitale D11, e viene collegato nativamente allo zoccolo DG7. Ma in qualche caso, una libreria utilizzata richiede che il piedino di data venga collegato alla porta D2, pertanto è necessario utilizzare lo zoccolo "universale DG11.

Come si vede nell'immagine, il primo pin a sinistra è relativo all'alimentazione a +5 volt, mentre l'ultimo a destra è la massa.
I due piedini centrali sono collegati insieme, quindi al segnale (in questo caso al piedino ***D2***) è sufficiente collegarne indifferentemente uno dei due.

Un programma per il ricevitore a 433 MHz

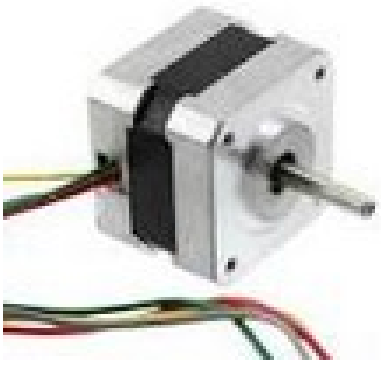
Abbiamo già trovato diversi programmi per il ricevitore a 433 MHz relativamente allo zoccolo **DG7,** ma per un progetto che richiede una libreria particolare, è necessario che venga utilizzata tassativamente la porta digitale D2. Il progetto è stato inserito tra quelli per la porta digitale "universale" DG11, ma potrebbe anche essere utilizzato lo zoccolo DY2 (quello relativo al display LCD 1602), perché anche su quello è presente la porta digitale D2. Questione di preferenze…

Naturalmente, affinché il progetto funzioni, è necessario che sia disponibile anche un secondo Arduino, con a bordo il trasmettitore a 433 MHz e il programma corrispondente.

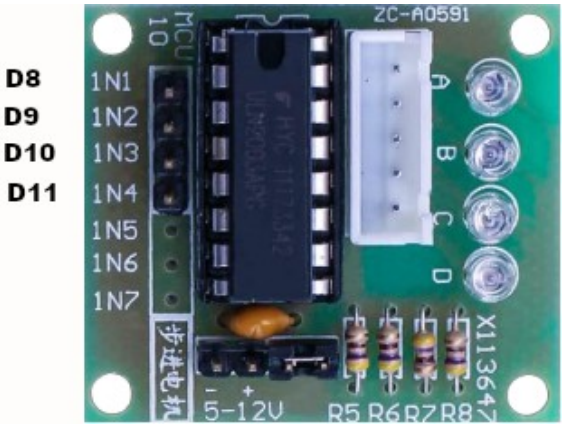
Il modulo ricevitore a 433 MHz

Nome del programma:	**433_servo_rx**
Porte:	**Modulo principale:**
D2 (digitale)	Ricevitore a 433 MHz
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D5 (digitale)	Servomotore (tipo MG90S) su DG9
Scopo del programma:	Sul programma del trasmettitore (***433_servo_tx***) è presente un potenziometro. Quando i due moduli sono in contatto, l'asse del servo ruota in un senso o nell'altro proporzionalmente alla rotazione del perno del potenziometro.
Librerie richieste:	RCSwitch.h, Servo.h
Note:	nessuna
Link:	https://srituhobby.com/433mhz-rf-transmitter-and-receiver-module-with-arduino/

Lo stepper motor



Uno stepper motor



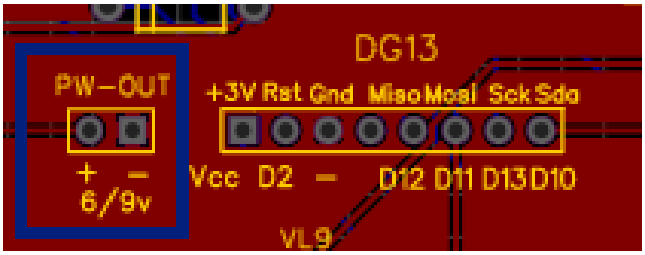
Il controller per lo stepper motor, ULN2003

Uno stepper motor è un motore passo-passo, con dei movimenti molto precisi, utilizzati massicciamente nelle stampanti 3D. Il tipo di motore che viene trattato in questo paragrafo è quello con due bobine separate (con quattro fili). Il controller ULN2003 gestisce il motore, trasformando le informazioni provenienti da Arduino in impulsi che fanno muovere il motore.

Il controller usa le porte digitali 8 (INI1), 9 (INI2), 10 (INI3), 11 (INI4).

Attenzione! Gli stepper motor hanno un notevole assorbimento di corrente e se alimentati direttamente potrebbero danneggiare i circuiti di Arduino Nano.

Per cui **è tassativo attivare l'alimentazione esterna**, inserendo l'alimentatore sul connettore di ingresso PWR EXT 6/9 v (in basso a sinistra della bs. Quando si inserisce si accende il led "DL1"



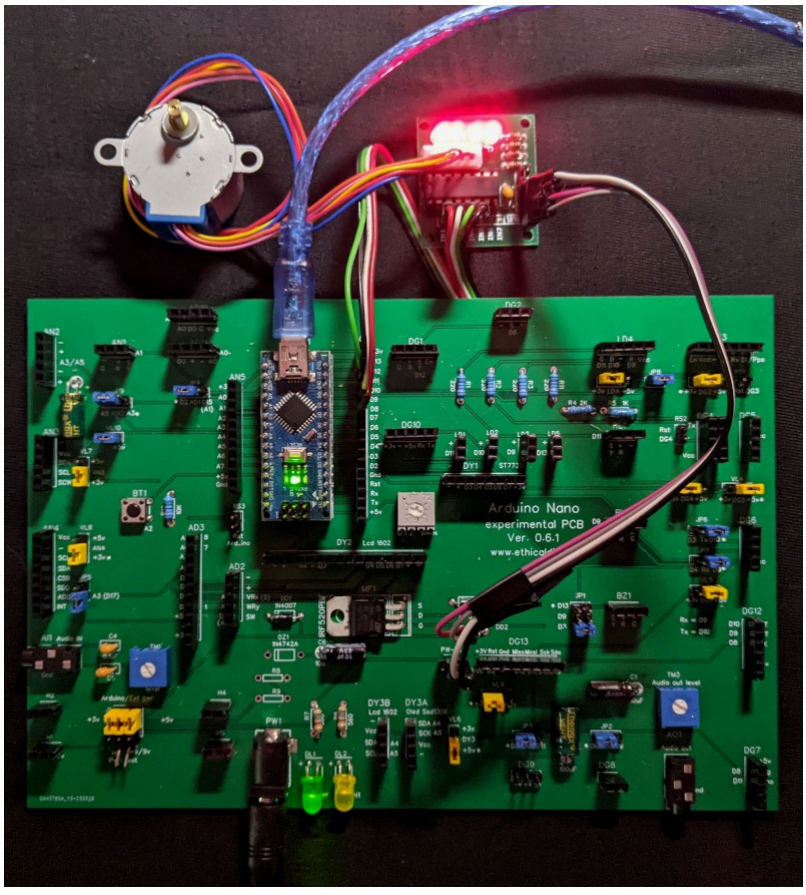
Collegare l'alimentazione dello stepper motor sul connettore PW-out.
E' necessario che sia collegato un alimentatore esterno che fornisca da 6 a 9 volt.

Programmi per lo stepper motor

Nome del programma: Stepper_example
Porte: **Modulo principale:**
D8/./D11 (digitali) **Stepper motor + ULN2003**
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Il programma lancia un ciclo continuo in cui lo stepper motor esegue un giro di 360° prima in senso orario, poi in senso antiorario
Librerie richieste: Stepper.h
Note: Variabile StepPerRevolution: per un giro di 360°, vale 2048; variabile rolePerMinute: indica la velocità di rotazione, da 0 a 17.
Link: nessuno

Nome del programma: Stepper_with_pot
Porte: **Modulo principale:**
D8/./D11 (digitali) **Stepper motor + ULN2003**
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Porte: **Comp. Accessori:**
A3 (analogica) Potenziometro (10KOhm) su AN2
Scopo del programma: Lo stepper motor esegue una rotazione con ampiezza proporzionale alla rotazione della manopola del potenziomentro, da 0 a circa 360°. E' inserita una variabile "test". Quando vale "1", il motore ruota alternativamente in senso orario e antiorario; se vale "2", solo in senso orario; se vale "3", solo in senso antiorario.
Librerie richieste: Stepper.h
Note: Variabile StepPerRevolution: per un giro di 360°, vale 2048; variabile rolePerMinute: indica la velocità di rotazione, da 0 a 17.
Link: nessuno

Nome del programma:	**Stepper_with_remote**
Porte:	**Modulo principale:**
D8/./D11 (digitali)	**Stepper motor + ULN2003**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D12 (digitale)	IR receiver KY-022 su zoccolo DG1
Scopo del programma:	Lo stepper motor esegue una rotazione di 360° in base al tasto premuto su di un telecomando. In questo caso, se si preme “vol+”, ruota in senso antiorario; se si preme “vol-”, ruota in senso orario”. I valori del telecomando variano da modello a modello; per trovare i valori corretti, vedere i programmi relativi al telecomando, clicca qui.
Librerie richieste:	Stepper.h
Note:	Variabile StepPerRevolution: per un giro di 360°, vale 2048; variabile rolePerMinute: indica la velocità di rotazione, da 0 a 17.
Link:	nessuno



Lo stepper motor in funzione con il suo controller.
I connettori IN1 → D8; IN2 → D9; IN3 → D10; IN4 → D11
L’alimentazione (esterna!) viene presa da PW OUT (sul lato sinistro, vicino allo zoccolo DG13)
Lo stepper motor è collegato con 4 connettori al controller
I 4 led indicano il funzionamento
L’alimentatore esterno è connesso all’ingresso di alimentazione “Ext Pwr 6/9 v”.

Multiplexer + Keypad

Per maggiori informazioni sul multiplexer, vedere il capitolo relativo su DY2. Clicca qui

Programmi con keypad in ingresso.

I seguenti programmi richiedono qualche spiegazione, perché le connessioni sono maggiormente complesse.

Il sistema di input è la tastiera Keypad 4 x4, ovvero una tastiera con 16 tasti. La tastiera richiede l'uso di ben 8 porte; sebbene essa potesse usare tutte porte digitali, si è scelto di collegare quattro pin sulle analogiche e solo quattro su quelle digitali, per lasciarne una parte libera per il multiplexer.

Ecco l'immagine con i collegamenti utilizzati per la keypad:

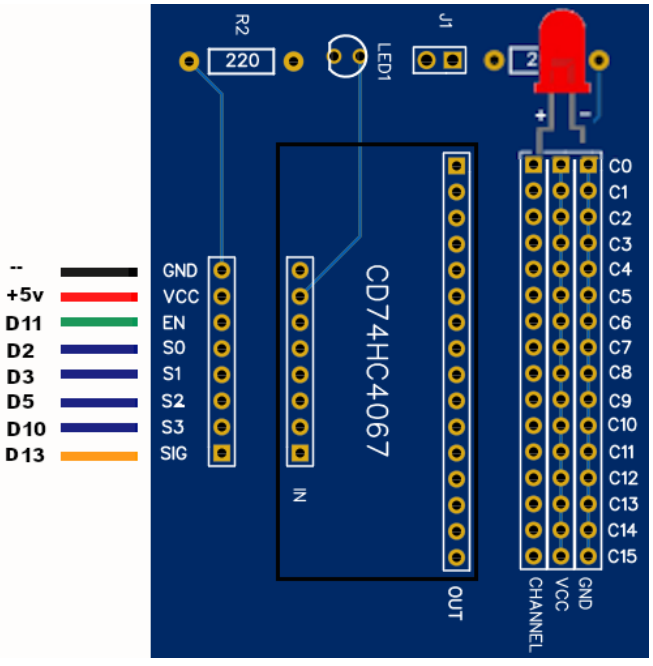
Se si usa la vecchia basetta, versione 0.5.1, le connessioni analogiche andranno effettuate sullo zoccolo AN5; le porte digitali su DG11.
Se invece si usa la nuova versione 0.6.1, si potranno collegare direttamente allo zoccolo AD3.

Mentre il multiplexer viene collegato usando le seguenti porte:

In questa immagine è stato inserito un solo led, ma naturalmente possono essere presenti in ogni slot, dallo 0 al 15.

Le connessioni verso Arduino sono le seguenti:
EN → D2
S0 → D3
S1 → D4
S2 → D5
S3 → D5
SIG → D7

Normalmente si possono usare qualsiasi porta digitale, ma in questo caso è necessario essere precisi, in modo da non sovrapporsi alle porte utilizzate dalla tastiera, dal dispaly OLED o dal buzzer.



Nota: attenzione: la Keypad usa molte porte (otto), sia digitali che analogiche, pertanto per non andare in conflitto, si sono dovute usare altre porte per il multiplexer, che non essendo doisponibili sullo zoccolo DY2, ci costringono a usare quello "universale", ovvero DG11.

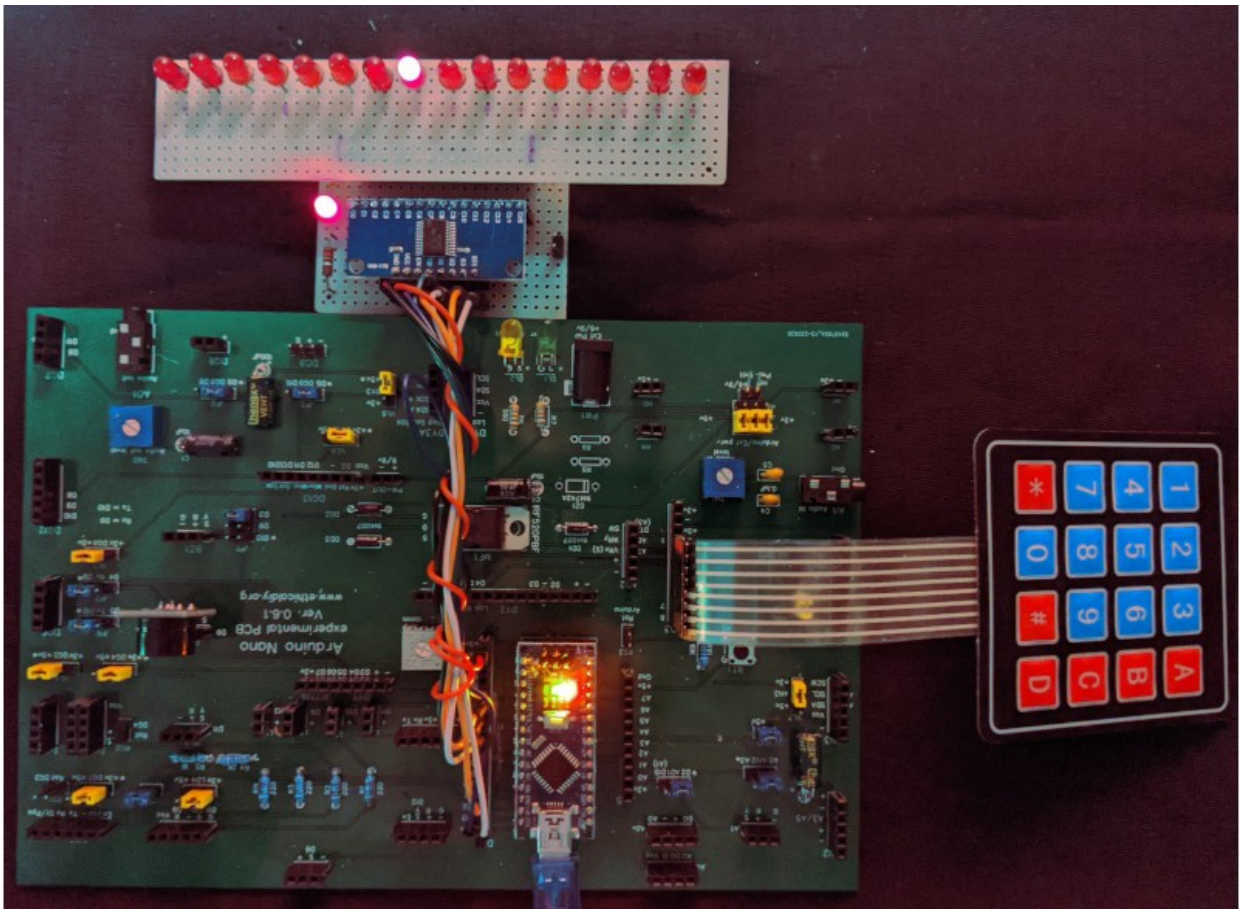
Uso del keypad.

Tasti da 0 a 9: fanno illuminare il led corrispondente. Dopo la selezione, premere il tasto "#". Prima di attivare la selezione successiva, Spegnere con il tasto "*".
Combinazione dei tasti 1+0, 1+1, 1+2, 1+3, 1+4, 1+5: fanno illuminare il led corrispondente. Dopo la selezione, premere il tasto "#". Prima di attivare la selezione successiva, Spegnere con il tasto "*".
Tasto "A": fa accendere in sequenza i led, da 0 a 15.
Tasto "B": fa accendere in sequenza i led, da 15 a 0.
Tasto "C": fa accendere in sequenza i led, da 0 a 15 e poi da 15 a 0.
Tasto "D": fa accendere in sequenza i led, da 0 a 7, poi da 7 a 0; di seguito da 8 a 15 e da 15 a 8.

Nome del programma:	**Arduino_deMUX**
Porte:	**Modulo principale:**
S0 = 2; S1 = 3; S2 = 5; S4 = 10; Sig = 13; En = 11 (Digit.)	Multiplexer CD74HC4067
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
A0, A1, A2, A3 (analogiche)	Keypad 4 x 4 su AN5 e DG11 (oppure se presente su AD3)
D4, D6, D7, D8 (digitali)	
C0/./C15	Led sul multiplexer
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno è inserito un led, che vengono comandati dal keypad. Vedi indicazioni.
Note:	nessuna
Link:	nessuno

Nome del programma:	**Arduino_deMUXa**
Porte:	**Modulo principale:**
S0 = 2; S1 = 3; S2 = 5; S4 = 10; Sig = 13; En = 11 (Digit.)	Multiplexer CD74HC4067
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
A0, A1, A2, A3 (analogiche)	Keypad 4 x 4 su AN5 e DG11 (oppure se presente su AD3)
D4, D6, D7, D8 (digitali)	
C0/./C15	Led sul multiplexer
D9	Buzzer su BZ1
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno è inserito un led, che vengono comandati dal keypad. Il buzzer emette alcune note, quando si premono i tasti "A", "B", "C", "D".
Note:	nessuna
Link:	nessuno

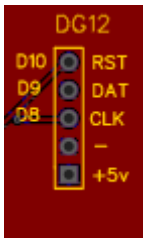
Nome del programma:	**Arduino_deMUX_oled**
Porte:	**Modulo principale:**
S0 = 2; S1 = 3; S2 = 5; S4 = 10; Sig = 13; En = 11 (Digit.)	Multiplexer CD74HC4067
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
A0, A1, A2, A3 (analogiche)	Keypad 4 x 4 su AN5 e DG11 (oppure se presente su AD3)
D4, D6, D7, D8 (digitali)	
C0/./C15	Led sul multiplexer
D9	Buzzer su BZ1
A4, A5	Display Oled 128 x 32 su DY3
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno è inserito un led, che vengono comandati dal keypad. Il buzzer dà alcune segnalazioni alla pressione dei tasti, e il display fornisce indicazioni sintetiche sulle azioni in corso.
Note:	nessuna
Link:	nessuno



Una delle animazioni con il multiplexer e la keypad: premendo il tasto “C”, si accendono i led in sequenza, prima in un senso e poi nell’altro. Si noti le basette sperimentali, sia per il multiplexer, sia per i led.

Per le basette stampate**, clicca qui**

Lo zoccolo digitale DG12



Lo zoccolo DG12 si trova in basso all'estrema destra della bs. E' stato aggiunto in questa nuova versione della bs per poter ospitare dei moduli con tre porte digitali, come il DS1302, un RTC (orologio).

Su questo zoccolo si potrebbe collegare il display 5641AS + ic 7HC595. Clicca qui per le informazioni.

Questo zoccolo va in conflitto con DG3,4,5,6,7.

Moduli ospitati sullo zoccolo DG12

Attualmente su questo zoccolo viene inserito nativamente solo DG1302 (RTC), ovvero un preciso orologio che restituisce la data e l'ora. La batteria di backup può durare fino a 10 anni.
Nulla vieta però di utilizzare questo zoccolo anche per moduli digitali che utilizzano solamente una o due porte.
DG1302 può essere in alcuni progetti una valida alternativa ad un altro RTC, DS1307, che però utilizza il protocollo I2C

DG1302

Socket DG12 (digital)			1	2	3	4	5
Description	Code	Note	VCC (+5v)	GND	CLK	DAT	RST
DS1302 RTC Clock			+	-	D8	D9	D10

Programmi per DG12

Nome del programma: ds1302_1
Porte: **Modulo principale:**
D8, D9, D10 (digitali) RTC DS1302
Porte: **Comp. Accessori:**
D2/./D7 Display LCD 1602 su DY2
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Il modulo orologio restituisce data e ora sia sul monitor seiale che sul display LCD
Note: Vedi note a fondo pagina.
Link: nessuno

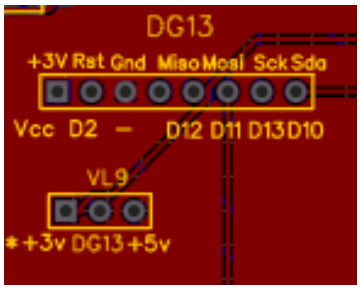
Nota: il programma ds1302_1a, usa il display LCD1602 con protocollo I2C, sullo zoccolo DY3/b

Per impostare la data e l'ora:

```
// Impostazione valori nella memori del DS1302
//rtc.setDOW(SATURDAY);        // Imposta il giorno della settimana a SUNDAY
//rtc.setTime(16, 03, 03);     // Imposta l'ora come 11:32:00 (Formato 24hr)
//rtc.setDate(4, 2, 2023);   // Imposta la data come 7 novembre 2020
```

Togliere le doppie barre "//" dalle ultime tre righe mostrate e impostare il giorno, la data e l'ora. Avviare una volta il programma affinché vengano salvate. Poi rimettere le barre di commento, per evitare che ogni volta reimposti l'ora inserita manualmente nel programma.

Lo zoccolo DG13



Lo zoccolo DG13 si trova sulla parte sinistra della bs, in basso.

Utilizza le porte digitali da D10 a D13.

Con il jumper VL9 si può scegliere la tensione di alimentazione.

Attenzione: RC522 funziona tassativamente a 3,3 v!

Il lettore di schede RFID RC522 è molto versatile, e si presta a tutta una serie di controlli di accesso, aperture di porte, ecc. Nella sezione dei programmi si troveranno alcuni spunti.

RC532 utilizza cinque porte digitali e ***deve essere tassativamente alimentato a 3,3 v***, pena la distruzione del sensore stesso!

Schema di collegamento di RC522

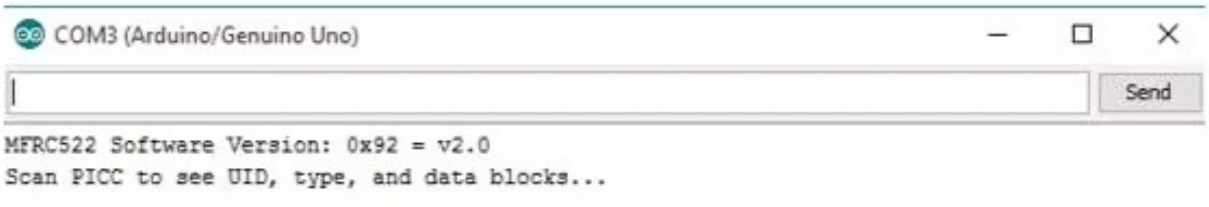
Il modulo RFID 522 si connette nativamente sullo zoccolo DG13

Programmi per modulo RFID RC522

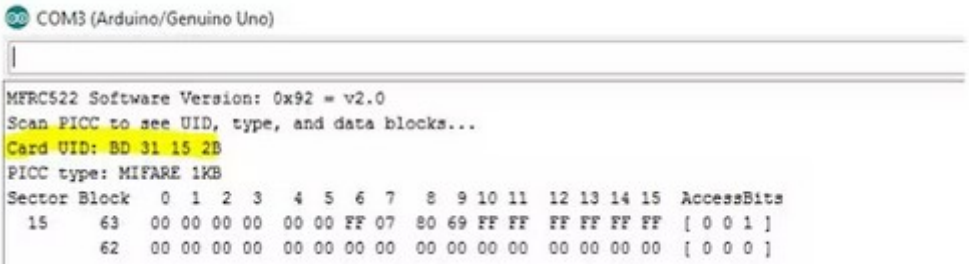
PinWiring to Arduino Uno

SDA----------Digital 10
SCK----------Digital 13
MOSI----------Digital 11
MISO----------Digital 12
IRQ----------unconnected
GND----------GND
RST----------Digital 2
3.3V----------3.3V (DO NOT CONNECT TO 5V)

La prima serie di programmi che utilizzano questo connettore montano il lettore **RFID RC522**. Questo modulo funziona solo a 3,3v e viene irrimediabilmente danneggiato a 5v, quindi prestare molta attenzione. In questa immagine si vede la sequenza delle connessioni, leggermente diversa da quella tradizionale: Infatti di solito il piedino "RST" è collegato su "D9", mentre in questi progetti è stato collegato a "D2".
I display "DY1" e "DY2" utilizzano entrambi le porte digitali da "D3 a D7", quindi avendo in comune "D6" non possono essere usati in questa configurazione. Al posto verrà usato un display OLED collegato alla porta "DY3".



Dopo aver collegato il modulo ad Arduino come indicato e aver scaricato la libreria MFRC522.h, aprire la IDE di Arduino, cliccare su File > Esempi > MFRC522 > DumpInfo.
Caricare il programma; dovrebbe apparire una videata del tipo:
Avvicinare la card Rfid al sensore e mantenerla in posizione fino a che non appaiono sul monitor seriale le informazioni:



Nome del programma:	**DumpInfo**
Porte:	**Modulo principale:**
D2, D10, D11, D12, D13 (digitali)	Modulo **RFID RC522**
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Permette di conoscere il codice della propria card.
Librerie richieste:	MFRC522.h
Note:	Attenzione a rispettare la polarità del lettore Rfid, **funziona a 3,3v!**
Link:	nessuno

Prendere nota del codice “card UID”, per esempio “BD 31 15 2B” e inserirle nel programma al punto indicato:

```
Serial.println();
Serial.print("Message : ");
Serial.print("RELAY: ");
content.toUpperCase();
if (content.substring(1) == "BD 31 15 2B")
{
```

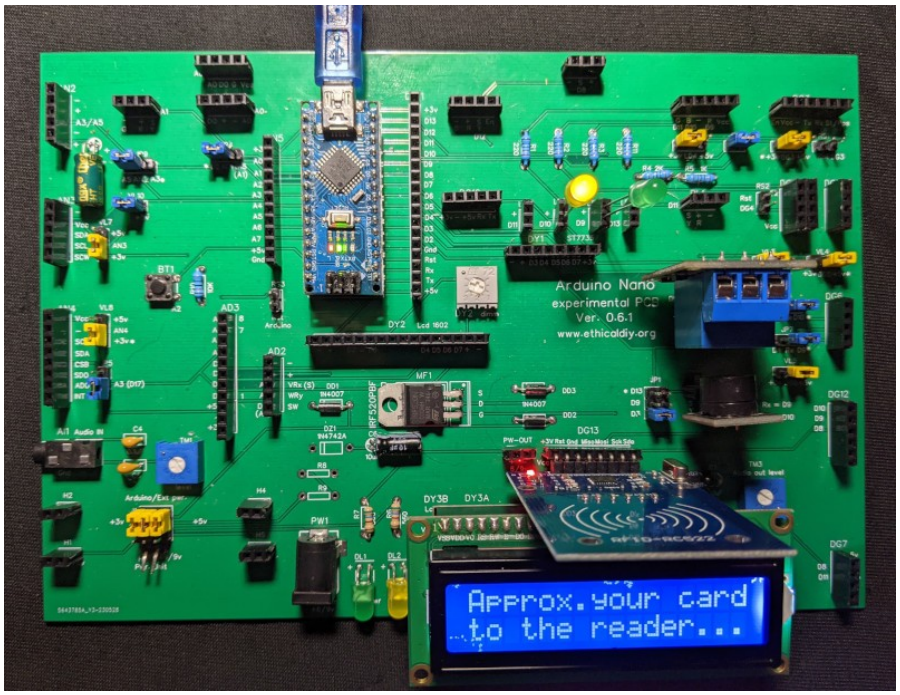
A questo punto, il programma dovrebbe permettere l’accesso. I listati seguenti, pur aggiungendo via via maggiori funzionalità, hanno tutti la stessa struttura.

Nome del programma: **Rfid_1**

Porte:	**Modulo principale:**
D2, D10, D11, D12, D13 (digitali)	Modulo **RFID RC522**
Porte:	**Comp. Accessori:**
D9	Led L1 - verde
D9	Relay KY-019
D11	Led L3 - blu
D3	Buzzer KY-006
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Il led3, blu, segnala semplicemente che il sistema è in funzione. Avvicinando la card al lettore. Se il codice è corretto, si percepisce un bip acuto, il relay si attiva, aprendo per esempio una porta o sbloccando un tornello, si accende il led verde e sul monitor serale appare "Authorized access for 5 seconds". Dopo 5 secondi, il relay si disattiva, si spegne il led verde. Nel caso il codice della card sia errato, si sente un suono di bassa frequenza e sul monitor seriale appare la scritta "access denyed".
Librerie richieste:	Spi.h, MFRC522.h
Note:	Attenzione a rispettare la polarità del lettore Rfid, **funziona a 3,3v!**
Link:	https://create.arduino.cc/projecthub/Raushancpr/diy-idea-with-rfid-89e41d?ref=tag&ref_id=rfid&offset=4

Nome del programma:	**Rfid_2**
Porte:	**Modulo principale:**
D2, D10, D11, D12, D13 (digitali)	Modulo **RFID RC522**
Porte:	**Comp. Accessori:**
D3	Buzzer KY-006 (KY-012) su BZ1
D9	Led L1 - verde
D9	Relay KY-019
D10	Led LD2 - blu
A4 – A5 (alalogiche)	Display OLED 128 x 64 su DY3
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Il led3, blu, segnala semplicemente che il sistema è in funzione. Avvicinando la card al lettore. Se il codice è corretto, si attiva il relay, aprendo per esempio una porta o sbloccando un tornello, si accende il led verde e sul monitor serale appare "Authorized access". Nel caso il codice della card sia errato, si sente un suono di bassa frequenza e sul monitor seriale appare la scritta "access denyed". Le stesse informazioni appaiono anche sul display Oled.
Librerie richieste:	Spi.h, MFRC522.h; Adafruit_GFX.h; Adafruit_SSD1306.h
Note:	Attenzione a rispettare la polarità del lettore Rfid, **funziona a 3,3v!**
Link:	https://create.arduino.cc/projecthub/Raushancpr/diy-idea-with-rfid-89e41d?ref=tag&ref_id=rfid&offset=4
Nome del programma:	**Rfid_2a**

Porte:	**Modulo principale:**
D2, D10, D11, D12, D13 (dig)	Modulo **RFID RC522**
Porte:	**Comp. Accessori:**
D3	Buzzer KY-006 (KY-012) su BZ1
D9	Led L1 - verde
D9	Relay KY-019
D10	Led LD2 - blu
A4 – A5 (alalogiche)	Display LCD 1602 con adattatore I2C su DY3/b
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Identico a Rfid2. Solamente le info vengono visualizzate sul display LCD1602 con adatt. I2C.
Librerie richieste:	Spi.h, MFRC522.h; Adafruit_GFX.h; Adafruit_SSD1306.h
Note:	Attenzione a rispettare la polarità del lettore Rfid, **funziona a 3,3v!**
Link:	https://create.arduino.cc/projecthub/Raushancpr/diy-idea-with-rfid-89e41d?ref=tag&ref_id=rfid&offset=4



La richiesta di avvicinare la card al lettore RC522

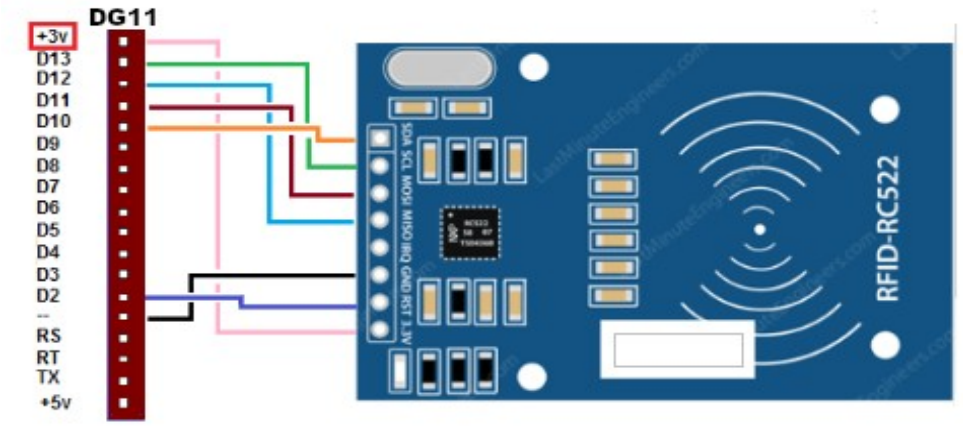
Nome del programma:	**Rfid3**
Porte:	**Modulo principale:**
D2, D10, D11, D12, D13 (digitali)	Modulo **RFID RC522**
Porte:	**Comp. Accessori:**
D9	Led L1 - verde
D9	Relay KY-019

D3	Buzzer KY-006
D5	Servomotor MG90S
D10	Led LD2 - blu
Plotter seriale: no	**Monitor Seriale: sì** (9600 bit)
Scopo del programma:	Il led3, blu, segnala semplicemente che il sistema è in funzione. Avvicinando la card al lettore. Se il codice è corretto, si percepisce un bip acuto, il relay si attiva, aprendo per esempio una porta o sbloccando un tornello, si accende il led verde e sul monitor serale appare "Authorized access for 5 seconds". Se è collegato il servomotore, il suo asse ruota di 90°, sbloccando per esempio una porta. Dopo 5 secondi, il relay si disattiva, si spegne il led verde e il rotore del servo torna in posizione ad angolo iniziale, ovvero 0, bloccando la porta. Nel caso il codice della card sia errato, si sente un suono di bassa frequenza e sul monitor seriale appare la scritta "access denyed".
Librerie richieste:	Spi.h, MFRC522.h
Note:	Attenzione a rispettare la polarità del lettore Rfid, **funziona a 3,3v!**
Link:	https://create.arduino.cc/projecthub/Raushancpr/diy-idea-with-rfid-89e41d?ref=tag&ref_id=rfid&offset=4

Nome del programma:	**Rfid_4**
Porte:	**Modulo principale:**
D2, D10, D11, D12, D13 (digitali)	Modulo **RFID RC522**
Porte:	**Comp. Accessori:**
D9	Led L1 - verde
D9	Relay KY-019
D3	Buzzer KY-006
D5	Servomotor MG90S
D10	Led LD2 - blu
A4 – A5 (alalogiche)	Display OLED 128 x 64 su DY3
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	Il led3, blu, segnala semplicemente che il sistema è in funzione. Avvicinando la card al lettore. Se il codice è corretto, si percepisce un bip acuto, il relay si attiva, aprendo per esempio una porta o sbloccando un tornello, si accende il led verde e sul monitor serale appare "Authorized access for 5 seconds". Se è collegato il servomotore, il suo asse ruota di 90°, sbloccando per esempio una porta. Dopo 5 secondi, il relay si disattiva, si spegne il led verde e il rotore del servo torna in posizione ad angolo iniziale, ovvero 0, bloccando la porta. Nel caso il codice della card sia errato, si sente un suono di bassa frequenza e sul monitor seriale appare la scritta "access denyed". Le stesse informazioni appaiono anche sul display Oled.
Librerie richieste:	Spi.h, MFRC522.h; Adafruit_GFX.h; Adafruit_SSD1306.h
Note:	Attenzione a rispettare la polarità del lettore Rfid, **funziona a 3,3v!**
Link:	https://create.arduino.cc/projecthub/Raushancpr/diy-idea-with-rfid-89e41d?ref=tag&ref_id=rfid&offset=4

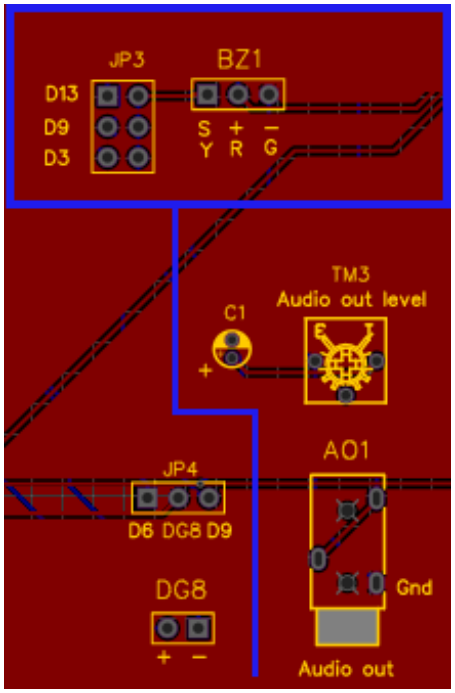
Collegamenti alternativi

Schema di collegamento di RC522



In caso di necessità, il modulo Rfid può essere collegato con successo anche su DG11, senza dover cambiare nulla nei programmi, collegandolo però con un cavetto a otto cavi.

Lo zoccolo BZ1 (digitale) e l'uscita "Audio out"



Lo zoccolo BZ1 si trova nella parte relativamente bassa della *bs*, verso destra, e permette di collegare un buzzer, un piccolo altoparlante piezoelettrico: KY-006 (passivo) o KY-012 (attivo, amplificato).

Esso è collegato all'uscita "Audio out" in basso verso destra, a cui si può attraverso un jack standard da 3,5 mm connettere una cuffia o un amplificatore.

Spostando i ponticelli cdi JP3, he si vede all'estrema sinistra dell'immagine, si può connettere alle porte digitali D13 (default), oppure alle PWM D3 o D9, se necessario. Ovviamente la porta utilizzata dal buzzer non può essere utilizzata da altre applicazioni.
Se si usa D3, non si può utilizzare i display su DY1 e DY2; se si usa la D9, non si può collegare il led (rosso) su LD3 né il relay su RL2.

Alla porta BZ1 possono essere anche collegati i moduli che alloggiano si DG1, con l'accortezza di ruotarli di 180°.

Il connettore **BZ1** è specifico per il buzzer (un piccolo altoparlante piezoelettrico). Abitualmente viene connesso alla porta 13. Per alcune applicazione sonore (per esempio per il programma *granular_synth*, si ha necessità di essere collegato su di una porta PWM (Pulse Width Modulation), ma purtroppo D13 non permette questa funzione. Per questo motivo è stato inserito sulla basetta un piccolo selettore, denominato "buzzer", in cui si può scegliere "D13" (default) o D3 (PWM), o ancora D9 (sempre PWM), per poter gestire alcuni sintetizzatori musicali che utilizzano le librerie "Mozzi". A questo zoccolo è connessa anche un'uscita audio "Audio out", nell standard dei jack da 3,5 mm (quelli usati abitualmente per le cuffie audio) a cui si può collegare una cuffia oppure l'ingresso di un amplificatore. Il volume dell'uscita audio è regolata da un piccolo trimmer, "TM3".

Come sempre, lo zoccolo BZ1 può essere utilizzato per collegare anche altri moduli, che utilizzano le stesse porte digitali. In questo caso, è possibile collegare su di esso un relay KY-019, senza ulteriori modifiche, nel caso le due porte dedicate ad essi, RL1 e RL2 non siano sufficienti. Si collega anche il led IR Trasmitter, alla porta D3 (vedi sotto).

Nota: fare attenzione, in base al progetto selezionato, di inserire il jumper per collegare il buzzer alla porta richiesta. Se si utilizza la porta D3 o la D9, non si potrà inserire moduli sulle porte che le utilizzano, pena conflitti. La porta digitale D3, per esempio, è utilizzata dal display LCD 1602, oppure il display TFT 7735, entrambi connessi su DY2; D9 è usata per i led che alloggiano su LD3, LD4 e il relay su RL2, ed altri.

Moduli che usano lo zoccolo BZ1

Socket BZ1 (digital)			1	2	3 *		
Description	Code	Note	-	+ (+5v)	S		
Active buzzer	KY-012 - HW512	+ uscita audio	-	+	D13	D9	D3
Passive buzzer	KY-006 - HW508						
IR Trasmitter	KY-005 – HW489	Piedinatura ruotata			/	/	

Immagine dei moduli che montano su BZ1:

KY-006 passive buzzer

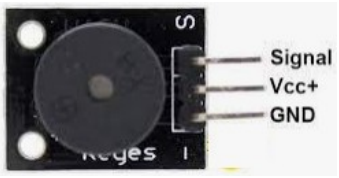
KY-012 active buzzer

KY-005 IR trasmitter

Nota 1: Allo zoccolo BZ1 possono essere anche collegati tutti i moduli che alloggiano si DG1, con l'accortezza di ruotarli di 180° e di modificare debitamente il programma.
Vedi i dati relativi allo zoccolo DG1.

Nota 2: Questo zoccolo ha la particolarità di poter essere collegato a tre porte digitali diverse: D13 (dafault), oppure D3 o D9, in base alle necessità

Programmi per il buzzer KY-006 – KY-012



I due buzzer si distinguono semplicemente perché KY-012 è amplificato, mentre KY-006 è semplicemente passivo, ed emette quindi un suono meno potente. I due si riconoscono perché KY-012 viene consegnato con un adesivo sul mini-cicalino; nel caso quest'ultimo fosse andato smarrito, sulla parte superiore è serigrafata la scritta "HYDZ"

Nome del programma: Buzzer_1
Porte: D13 (digitale)
Modulo principale: Buzzer (piccolo altoparlante piezoelettrico) KY-006 KY-012 (amplif.)
Monitor Seriale: sì (9600 baud)
Plotter seriale: no
Scopo del programma: Produce una semplice sequenza di note, tipo "allarme nucleare"
Note: Scegliere sul selettore buzzer "D13".
Link: nessuno

Nome del programma:

Porte:	**Buzzer_2**
D13 (digitale)	Buzzer (piccolo altoparlante piezoelettrico) KY-006 KY-012 (amplif.)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D8 (trig)	Ultrasonic Sensor KY-050; HC-SR04
D11 (echo)	
Scopo del programma:	In base alla distanza dal sensore ultrasonico, il buzzer emette una nota, bassa quando si è vicini, sempre più alta quando progressivamente ci si allontana.
Note:	Scegliere sul selettore buzzer “D13”.
Link:	nessuno

Nome del programma:	**Buzzer_3**
Porte:	**Modulo principale:**
D13 (digitale)	Buzzer (piccolo altoparlante piezoelettrico) KY-006 KY-012 (amplif.)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
A1 (analogico)	Photoresistor - KY-018
D11 (digitale)	Led L1 - verde
Scopo del programma:	Nei primi 5 secondi, quando il led è acceso, si può calibrare la sensibilità della fotoresistenza, che influirà poi la gamma sonora. Quando il led si spegne, il buzzer inizia a produrre suoni, più bassi quando la luce è intensa, progressivamente più alta quando ponendo una mano tra la fonte luminosa e il sensore. Diventa quindi una specie di rudimentale theremin, uno dei primi strumenti musicali elettronici.
Note:	Scegliere sul selettore buzzer “D13”.
Link:	nessuno

Nome del programma:	**Buzzer_4**
Porte:	**Modulo principale:**
D13 (digitale)	Buzzer (piccolo altoparlante piezoelettrico) KY-006 KY-012 (amplif.)
Monitor Seriale: sì (9600 baud)	**Plotter seriale:** no
Porte:	**Comp. Accessori:**
A3 (analogico)	Potentiometer (10 KΩ, lineare)
D11 (digitale)	Led L1 - verde
Scopo del programma:	Il programma è simile al precedente, solo che la frequenza del suono è determinata da un potenziometro invece che dalla fotoresistenza. In base alla rotazione della manopola, il suono progressivamente sale da basso ad alto e viceversa.
Note:	Scegliere sul selettore buzzer “D13”.
Link:	nessuno

Nome del programma: **Fur Eloise**

Porte:	**Modulo principale:**
D13 (digitale)	Buzzer (piccolo altoparlante piezoelettrico) KY-006 KY-012 (amplif.)
Monitor Seriale: sì (9600 baud)	**Plotter seriale:** no
Scopo del programma:	In questo programma vengono definite le note della scala cromatica e viene eseguito il celebre inizio di “per Elisa”.
Note:	Scegliere sul selettore buzzer “D13”.
Link:	https://www.meccanismocomplesso.org/generare-toni-musicali-a-440hz-e-432hz-con-arduino/

Nome del programma:	**Scala_1**
Porte:	**Modulo principale:**
D13 (digitale)	Buzzer (piccolo altoparlante piezoelettrico) KY-006 KY-012 (amplif.)
Monitor Seriale: sì (9600 baud)	**Plotter seriale:** no
Scopo del programma:	In questo programma viene eseguita una scala musicale.
Librerie richieste:	pitches.h
Note:	Scegliere sul selettore buzzer “D13”.
Link:	https://www.meccanismocomplesso.org/generare-toni-musicali-a-440hz-e-432hz-con-arduino/

Programmi per IR trasmitter KY-005

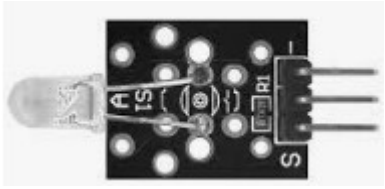


fig. 1 serigrafia corretta

fig. 2 serigrafia errata

Il modulo “IR Trasmitter” (trasmettitore led a raggi infrarossi) deve essere collegato ***sempre*** alla porta D3, e per questo utilizza il connettore BZ1.
Attenzione alla polarità, perché questo modulo non sempre presenta la stessa serigrafia. Abitualmente ha quella standard per un buon numero di moduli, ovvero da sinistra a destra, “S” (signal), “+” (+5v) e “-” (massa). Nel mio esemplare, sempre da sinistra a destra appare: “+”, niente nel piedino centrale, “S” a destra. Evidentemente è stata usata una basetta per altri usi. Non lasciarsi confondere, inserirlo seguendo l’ordine della figura 1.
Nota: il sensore IRTrasmitter va inserito ruotato di 180° rispetto al buzzer. (confrontare le due immagini).
Non si può escludere che anche altri moduli che si trovano in commercio abbiano una serigrafia errata. Porre molta attenzione, e se inserendo un modulo sullo zoccolo non si vedono risultati, oppure le luci dei led di Arduino si affievoliscono, spegnere immediatamente!

In questa sezione viene presentato semplicemente un abbozzo di programma, perché il

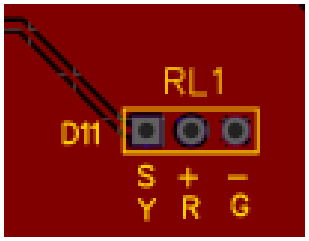
trasmettitore deve essere sincronizzato con i programmi del ricevitore, KY-022 (**vedi programmi relativi**), in modo da inviare comandi che siano "compresi" dal ricevitore.
Nel caso non funzionasse al primo colpo, come fare a verificare se il problema è nel trasmettitore o nel ricevitore?

E' abbastanza facile. Per il ricevitore: caricare il primo programma e premere un tasto qualsiasi del proprio telecomando, in corrispondenza del ricevitore. Se sul monitor seriale appare qualcosa, il modulo funziona.

Per il trasmettitore: attivare il modulo. Aprire la fotocamera sul proprio smartphone e puntarla verso il led. Se si vede una tenue lucina (invisibile a occhio nudo), il modulo funziona.

Nome del programma:	**IR_Trasmitter_1**
Porte:	**Modulo principale:**
D3 (digitale)	**KY-005** su zoccolo BZ1, jumper su porta digitale D3
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Invia una serie di istruzioni a otto bit. Da ricevere con il modulo KY-022
Note:	Scegliere sul selettore buzzer "D3", perché è necessaria una porta "PWM".
Link:	nessuno

Lo zoccolo RL1



RL1 si trova a destra della basetta, abbastanza in alto, di fianco agli zoccoli LD1/LD2/LD3 (per i led).
Può ospitare un relay KY-019, contestualmente collegato alla porta LD1; per cui il led corrispondente (abitualmente verde), ne indica il funzionamento.
La porta utilizzata è D11.

Alla porta RL1 possono essere anche collegati i moduli che alloggiano si DG1, con l'accortezza di ruotarli di 180°.

Attenzione: nel caso si intenda pilotare all'uscita del relay un dispositivo connesso alla rete elettrica a 230 V, richiedere l'intervento di un tecnico specializzato. La tensione a 230 V può essere molto pericolosa, addirittura mortale!

RL1. Questo zoccolo è stato progettato per connettere nativamente un relay KY-019, ma ad esso si può collegare un qualsiasi relay a tre fili che funzioni a +5v, rispettando la piedinatura, nel caso non sia posta nello stesso ordine. RL1 è connesso alla porta digitale "D11", la stessa usata da LD1, che abitualmente ospita un led verde. Questo abbinamento non è casuale: infatti quando il relay viene eccitato, se il led è presente sullo zoccolo, si accende. Questo serve per avere una indicazione visibile dell'attivazione del relay stesso. Le porte RL1 e RL2 rendono Arduino un potente strumento, che in base alla sua ricca serie di sensori, può pilotare degli apparecchi esterni che possono funzionare *anche* a tensioni di rete, quali lampadine, elettrodomestici, attuatori vari, ecc. ***Nel caso di collegamento alla rete elettrica, è necessario affidarsi a del personale esperto e qualificato, perché uno shock dovuto a una scossa a 220 v può essere mortale, oppure avere degli effetti gravi per il proprio corpo.***

KY-019

Moduli che usano il connettore RL1:

Socket RL1 (digital -D11)			1	2	3
Description	Code	Note	-	+ (+5v)	S
Relay	KY-019 – HW482		-	+	D11

Nota: Allo zoccolo RL1 possono essere anche collegati tutti i moduli che alloggiano si DG1, con l'accortezza di ruotarli di 180° e di modificare debitamente il programma, variando la porta di collegamento del modulo da (D) 12 a (D) 11. La lettera "D" è tra parentesi perché nella dichiarazione delle variabili viene omessa; la riga del programma sarà per esempio, "int 12", oppure "int 11". **Vedi i dati relativi allo zoccolo DG1**.

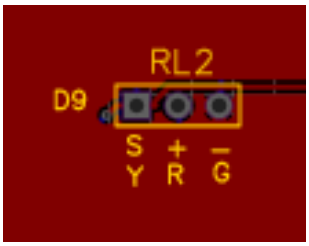
Programmi per il relay KY-019 (su RL1)

In questa sezione si inserisce un paio di semplici programmi dimostrativi. Il relay viene usato in moltissimi programmi. Vedi la sezione "applicazioni" dal foglio elettronico esterno "tabelle", in cui appare l'elenco completo dei programmi presenti in questo manuale, e contestualmente tutti i moduli utilizzati. Teoricamente, ogni volta che un'applicazione richiama il led presente su "LD1", potrebbe essere collegato parallelamente un relay.

Nome del programma:	**RL1_1**
Porte:	**Modulo principale:**
D11 (digitale)	**Relay KY-019**
Monitor Seriale: sì (9600 baud)	**Plotter seriale:** no
Porte:	**Comp. Accessori:**
D11 (digitale)	Led LD1 – verde (opzionale)
D11 (digitale)	Led LD4 – luce verde (opzionale)
Scopo del programma:	Quando si preme il pulsante BT1, si attiva il relay e si accende il led su LD1 (verde), o la luce verde del led a tre colori posto su LD4
Note:	Nessuna
Link:	nessuno

Nome del programma:	**RL1_2**
Porte:	**Modulo principale:**
D9 (digitale)	**Relay KY-019**
D11 (digitale)	**Relay KY-019**
Monitor Seriale: sì (9600 baud)	**Plotter seriale:** no
Porte:	**Comp. Accessori:**
D9 (digitale)	Led LD3 – rosso (opzionale)
D11 (digitale)	Led LD1 – verde (opzionale)
Scopo del programma:	Quando si preme il pulsante BT1, si attiva il relay e si accende il led su LD1 (verde); altrimenti resta attivo il relay 2 e il led rosso.
Note:	Nessuna
Link:	nessuno

Lo zoccolo RL2



RL2 si trova a destra della basetta, a media altezza, sotto RL1.
Può ospitare un relay KY-019, contestualmente collegato alla porta LD3; per cui il led corrispondente (abitualmente rosso), ne indica il funzionamento.
La porta utilizzata è D9

Alla porta RL1 possono essere anche collegati i moduli che alloggiano si DG1, con l'accortezza di ruotarli di 180°.

Attenzione: nel caso si intenda pilotare all'uscita del relay un dispositivo connesso alla rete elettrica a 230 V, richiedere l'intervento di un tecnico specializzato. La tensione a 230 V può essere molto pericolosa, addirittura mortale!

RL2. Esso ha le stesse caratteristiche e richiede le stesse precauzioni che per RL1, di cui è il gemello. Semplicemente si collega alla porta digitale D9, la stessa del led alloggiato su LD3, abitualmente rosso, ma naturalmente si può utilizzare un led del colore preferito. Sono state progettate due porte per i relays affinché si possano creare dei progetti più elaborati, che permettano di pilotare più apparati esterni, come nel caso di antifurti, di progetti di domotica, di giardinaggio, per esempio per predisporre l'attivazione di pompe per innaffiare l'orto o il giardino in modo differenziato.

Moduli che usano il connettore RL2:

Socket RL2 (digital – D9)			1	2	3
Description	Code	Note	-	+ (+5v)	S
Relay	KY-019 - HW482	+ uscita audio	-	+	D9

Nota 1: come si può vedere, BZ1, RL1 e RL2 hanno la stessa disposizione di piedini. Se per esempio si dovessero usare tre o quattro relays, uno potrebbe essere alloggiato su BZ1 (in questo caso naturalmente non si potrà usare il buzzer), ma anche collegare il buzzer su RL1 o RL2, nel caso fosse necessario dal progetto una porta PWM, come D9 o D11. Viva la fantasia!

Nota 2: se poi addirittura si dovesse testare un progetto con quattro relay, l'ultimo potrebbe essere collegato sullo zoccolo DG1, atto allo scopo, prestando attenzione che lo zoccolo è a quattro connettori; quello contrassegnato "EN" non viene utilizzato.

Nota 3: la porta LD4 duplica quelle presenti in LD1, LD2, LD3 ed è stata progettata per usare un led a tre colori. Quindi può essere usata per sapere quale relay è attivo in base al colore di un singolo led!

Nota 4: alla porta RL1 possono essere anche collegati i moduli che alloggiano si DG1, con l'accortezza di ruotarli di 180°,e di modificare debitamente il programma, variando la porta di collegamento del modulo da (D) 12 a (D) 9. La lettera "D" è tra parentesi perché nella dichiarazione delle variabili viene omessa; la riga del programma saràper esempio, "int 12", oppure "int 9".
Vedi i dati relativi allo zoccolo DG1 .

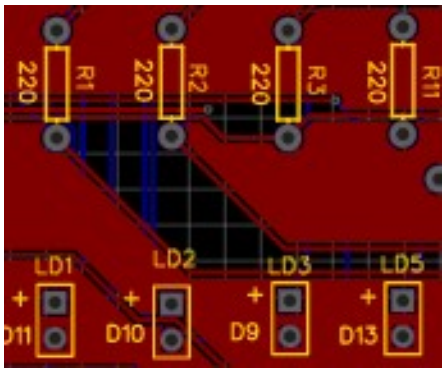
In base a queste considerazioni, nel caso fosse necessario testare un progetto con tre relais, si può utilizzare anche lo zoccolo BZ1, modificando opportunamente il programma. Addirittura si potrebbe arrivare a usare un quarto relay, collegandolo alla porta DG1, come già indicato nella descrizione di quello zoccolo. Invece è raro (ma non impossibile) il caso in cui siano necessari 2 o più buzzer in uno stesso progetto.

Programmi per il relay KY-019 (su RL2)

Nome del programma:	**RL2_1**
Porte:	**Modulo principale:**
D9 (digitale)	**Relay KY-019**
Monitor Seriale: sì (9600 baud)	**Plotter seriale:** no
Porte:	**Comp. Accessori:**
D9 (digitale)	Led LD3 – rosso (opzionale)
D9 (digitale)	Led LD4 – luce rossa (opzionale)
Scopo del programma:	Quando si preme il pulsante BT1, si attiva il relay e si accende il led su LD1 (verde), o la luce verde del led a tre colori posto su LD4
Note:	Nessuna
Link:	nessuno

Nome del programma:	**RL2_2**
Porte:	**Modulo principale:**
D9 (digitale)	**Relay KY-019**
D11 (digitale)	**Relay KY-019**
Monitor Seriale: sì (9600 baud)	**Plotter seriale:** no
Porte:	**Comp. Accessori:**
D2/./D7	Display LCD 602 su zoccolo DY2
D9 (digitale)	Led LD3 – rosso (opzionale)
D11 (digitale)	Led LD1 – verde (opzionale)
Scopo del programma:	Quando si preme il pulsante BT1, si attiva il relay e si accende il led su LD1 (verde); altrimenti resta attivo il relay 2 e il led rosso.
Note:	Nessuna
Link:	nessuno

Gli zoccoli LD1, LD2, LD3, LD5



I tre zoccoli LD1, LD2, LD3, rispettivamente collegati alle porte PWM D11, D10, D9.
Il connettore LD4 usa le stesse porte, quindi se si collegano contemporaneamente i led su LD1/2/3/4, si accenderanno all'unisono.

Gli zoccoli si trovano nella parte destra della bs, in alto.

Sono presenti tre resistenze, per ridurre la corrente che attraversano i led, per evitare di bruciarli.

Attenzione alle polarità nell'inserimento dei led!

LD1, LD2, LD3. Questi tre piccoli zoccoli possono ospitare un led su ognuno di essi. Abitualmente su LD1 si inserisce un led verde, su LD2 un led blu e su LD3 un led rosso.

Led rosso

Naturalmente ognuno di noi può usare i led del colore che preferisce. Si è usata questa disposizione, perché nel caso si inserisca il led a tre colori nello zoccolo LD4, si accenderebbero contemporaneamente con gli stessi colori di ogni singolo led connesso su LD1, ecc. A ognuno di questi zoccoli è connessa una resistenza di 220 ohm, affinché una tensione e una corrente troppo elevata non bruci in breve tempo il led collegato.
LD5 è stato aggiunto su questa nuova versione della bs, perché si è visto che a volte le prime tre porte dedicate sono già utilizzate. Per usare il led presente su LD5, variare la porta da D9, D10 o D11 in D13. **Attenzione:** può andare in conflitto con il buzzer, che di default usa la stessa porta.

Moduli che usano il connettore LD1, LD2, LD3, Ld5:

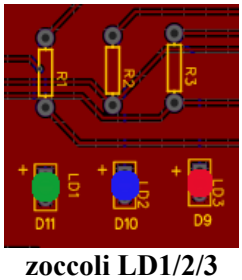
Sockets LD1, LD2, LD3, LD5 (digital PWM)			1	2
Description	Code	Note	GND	S
Led 1 – Green	---	resistenza 220 Ω	-	D11
Led 2- Blu	---	resistenza 220 Ω	-	D10
Led 3 – Red	---	resistenza 220 Ω	-	D9
Led 5 – generic	Questa porta non è PWM	resistenza 220 Ω	-	D13
Seven colours flash led	KY-034 – HW481	resistenza 220 Ω	-	D9

Nota: i led sono semiconduttori, e come tali sono polarizzati. Guardare attentamente l'immagine de led rosso: si vedrà un connettore leggermente più lungo dell'altro; quello è l'anodo, e va orientato in modo che corrisponda al segno "+" presente in alto a sinistra dello zoccolo.

Su questi zoccoli non vengono usati dei moduli, bensì i led "nudi".
Qui sono raccolti alcuni test per i connettori relativi ai led. Ce ne sono quattro:

- LD1: abitualmente per il led verde (D11)
- LD2: abitualmente per il led blu (D10)
- LD3: abitualmente per il led rosso (D9)
- LD5: un led di colore vario: giallo, bianco, ecc. (D13)

Su questa porta si possono collegare anche i led a due colori (KY-011); i led a tre colori (RGB led), KY-016); i led a tre colori SMD RGB, KY- 009). Purtroppo



zoccoli LD1/2/3

questi ultimi tre hanno piedinatura diversa, quindi è necessario usare un connettore maschio/femmina a quattro fili, facendo attenzione ai collegamenti, per non danneggiare i led.

Nota: LD4 condivide le stesse porte digitali di LD1, LD2, LD3. Pertanto i segnali arrivano in parallelo alle due configurazioni; in altre parole, LD4 non è che un duplicato "sintetico" dei tre connettori singoli e non lavora indipendentemente da essi. Nelle immagini si vedono anche le resistenze necessarie per ridurre la corrente che arriva ai led, per evitare di bruciarli in breve tempo. componenti in miei possesso (provenienza Elegoo). Per cui conviene sempre verificare con attenzione. E' importante trovare il "ground, ovvero il "-" per non alimentare i led al contrario.

Programmi per i led su Ld1, LD2 LD3, LD5

Nome del programma:	**Two_colour_test**
Porte:	**Modulo principale:**
D10, D11	led singoli su porte LD1 ed LD2.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	I led rosso e verde lampeggiano a ritmo alternato.
Note:	Attenzione alla sequenza dei piedini su LD4.
Link:	nessuno

Nome del programma:	**3_led_test**
Porte:	**Modulo principale:**
D9, D10, D11	led singoli su porte LD1 LD2 ed LD3.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	I led a tre colori si accendono con diversi schemi
Note:	I led RGB e SMD hanno una disposizione dei piedini diversa da quella offerta dal connettore L4. Usare un cavo a quattro poli maschi/femmina, facendo attenzione alla polarità.
Link:	nessuno

Nome del programma:	**3_led_millis()**
Porte:	**Modulo principale:**
D9, D10, D11	led singoli su porte LD1 LD2 ed LD3, oppure il led a tre colori sullo zoccolo LD4.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	I led a tre colori lampeggiano a diverse frequenze, ma non usano la funzione delay() che interrompe qualsiasi operazione, bens millis(), che fa lampeggiare i diversi led indipendentemente, senza stoppare il programma.
Note:	Inessuna
Link:	nessuno

Nome del programma:	**3_led_pwm_test**
Porte:	**Modulo principale:**
D9, D10, D11	led singoli su porte LD1 LD2 ed LD3.
Monitor Seriale: no	**Plotter seriale: no**

Scopo del programma: Si sfrutta la potenzialità delle porte PWM, facendo variare progressivamente la luminosità dei led.
Note: I led RGB e SMD hanno una disposizione dei piedini diversa da quella offerta dal connettore L4. Usare un cavo a quattro poli maschi/femmina, facendo attenzione alla polarità.
Link: nessuno

Nome del programma: **4_led_test**
Porte: **Modulo principale:**
D9, D10, D11, D13 led singoli su porte LD1, LD2, LD3 e LD5
Monitor Seriale: no **Plotter seriale: no**
Scopo del programma: I quattro led led si accendono con diversi schemi
Note: nessuna
Link: nessuno

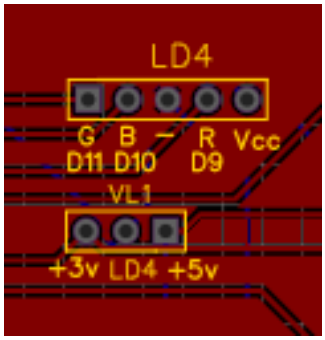
Nome del programma: **4_led_millis()**
Porte: **Modulo principale:**
D9, D10, D11 led singoli su porte LD1 LD2 ed LD3, LD5
Monitor Seriale: no **Plotter seriale: no**
Scopo del programma: I quattro led lampeggiano indipendentemente a diverse frequenze, ma non usano la funzione delay() che interrompe qualsiasi operazione, bens millis(), che fa lampeggiare i diversi led indipendentemente, senza stoppare il programma.
Note: nessuna
Link: nessuno

Nome del programma: **seven_colour_1**
Porte: **Modulo principale:**
D11 (digitale) **KY-034, seven colour led su LD1.** In questo modulo è inserito un led a tre colori, che però non vengono pilotati singolarmente, ma vengono combinati insieme, in modo da formare sette colori in sequenza.
Monitor Seriale: no **Plotter seriale: no**
Scopo del programma: Caricando il programma, immediatamente il led lampeggia in sequenza nei sette colori. Questo modulo può essere solo acceso o spento.
Note: Collegare il piedino “G” al “+” di LD1; collegare “-” tra il modulo e il piedino “-” di LD1
Link: nessuno

Nome del programma: **seven_colour_2**
Porte: **Modulo principale:**

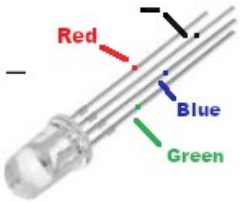
D11 (digitale)	**KY-034, seven colour led su LD1.** In questo modulo è inserito un led a tre colori, che però non vengono pilotati singolarmente, ma vengono combinati insieme, in modo da formare sette colori in sequenza.
Porte:	**Comp. Accessori:**
A2 (analogica)	Pulseante BT1 (integrato sulla basetta)
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	Programma leggermente meno “brutale” del precedente, il led si accende e lampaggia nei 7 colori quando si preme il pulsante “BT1” presente sulla bs.
Note:	Collegare il piedino “G” al “+” di LD1; collegare “-” tra il modulo e il piedino “-” di LD1
Link:	nessuno

Lo zoccolo LD4



Lo zoccolo LD4 si trova in alto, sulla destra della bs.
Se si usa un led a tre colori con i cavi "nudi" (vedi immagine sotto), fare attenzione alla polarità. Questo zoccolo è stato progettato per led a catodo comune, e non funziona per quelli ad anodo comune, che potrebbero essese rapidamente danneggiati.
Le resistenze riducono la corrente che attraversano le tre giunzioni dei led, per non bruciarli.
Il connettore LD4 usa le stesse porte, rispettivamente di LD1, LD2 e LD3 quindi se si collegano contemporaneamente i led su LD1/2/3/4, si accenderanno allunisono.
Questo zoccolo può essere usato anche per altre applicazioni, infatti è presente un quinto connettore, che può fornire 3,3 oppure 5 v, modificando la posizione del jumper VL1.

LD4. Questo zoccolo può ospitare nativamente un led a tre colori (vedi immagine), a patto che abbia la stessa sequenza dei piedini, oppure collegandoli con un cavettoi led RGB Led, SMD Led (KY-009) o Two colour led (KY-011). Da un punto di vista hardware usa le stesse porte digitali dei tre led singoli, ovvero D11 per il verde, D10 per il blu e D9 per il rosso. Usando i *programmi adeguati*, si possono miscelare i colori, con effetti interessanti. Anche per questo zoccolo si usano tre resistenze da 220 ohm, per non danneggiare i led stessi.
Nota: anche in questo caso, fare attenzione alla sequenza dei piedini da collegare allo zoccolo; connettendoli in modo errato, si rischia di danneggiarli in modo irreversibile. **Usare solo led a catodo comune.**



Moduli che usano il connettore LD4:

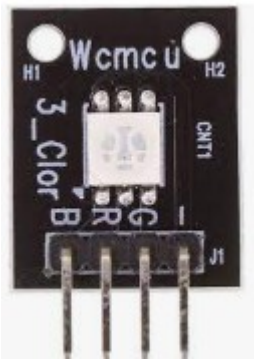
Socket LD4 (digital PWM) + multiconnector (A/D)			1	2	3	4	5
Description	Code	Note	G	B	-	R	+5v/+3,3v
Led multicolor	KY-034 - HW481	3 x res. 220 Ω	D11	D10	-	D9	/
RGB led	KY-016 - HW479	Usare connettore – diversa piedinatura	D11	D10	-	D9	/
SMD RGB Led	KY-009 - HW478	Usare connettore – diversa piedinatura	D11	D10	-	D9	/
Magic light cup	KY-027 – HW499	Usare solo i piedini 2,3,4,5		D10 [L]	-	D9 [S]	+ 5v
Two colours led	KY-029 – HW-477	Usare solo i piedini 2,3,4		D10 [R]	-	D9 [G]	/
Two colours led	KY-011 – HW-480	Usare connettore – diversa piedinatura	D11	-	/	D9	/

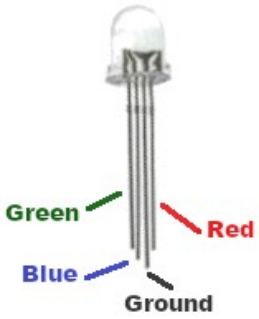
Nota: questo zoccolo può essere usato anche per moduli che usino una, due o anche tre porte digitali, inserendo i connettori opportuni. Lo zoccolo può anche fornire, in base alla posizione del jumper, 5v oppure 3,3 v. Osservare però che la tensione che arriva sui piedini digitali contrassegnati "G" (porta D11), "B" (D10) e "R" (D10) è inferiore a 5 volt, a causa della presenza delle resistenze. Verificare se essa è sufficiente per gestire il modulo che eventualmente si connetterà. Questa riflessione eviterà di farci impazzire, pensando che il modulo in prova non funzioni!

I led, la loro piedinatura; da inserire sullo zoccolo LD4

I led in oggetto, a tre o quattro connettori, hanno una piedinatura particolare, che verrà illustrata nella tabella seguente. Quindi controllarla bene, prima di inserirli nei connettori, per evitare di danneggiarli.

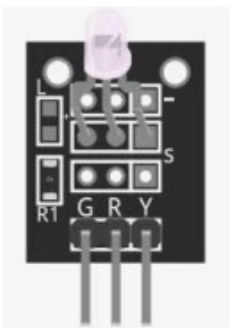
Immagini dei moduli che usano LD4:

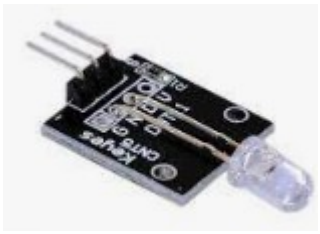




Led SMD a tre colori KY-009		**Led RGB, a tre colori KY-016**		**Led a tre colori**		**Magic light cup KY-027**	
Modulo	**bs**	**Modulo**	**bs**	**Modulo**	**bs**	**Modulo**	**bs**
B = blue	B	B = red	B	Green	G	G = ground (-)	-
R = red	R	G = green	G	Blue	B	+ = +5v	+ 5v
G = green	G	R = blue	R	Ground	-	S = switch	B
- = ground	-	- = ground	-	Red	R	L = led	R
Nota: piedinatra diversa. Usare connettore a 4 fili.		**Nota:** piedinatra diversa. Usare connettore a 4 fili.		**Nota**: questo led monta ***nativamente*** sullo zoccolo LD4. Attenzione alla piedinatura.		**Nota:** richiede un connettore a quattro fili, causa piedinatura diversa.	

Nota: la serigrafia delle immagini trovate su Internet non corrisponde a quella trovata sui componenti in miei possesso (provenienza Elegoo). Per cui conviene sempre verificare con attenzione. E' importante trovare il "ground, ovvero il "-" per non alimentare i led al contrario.





Led a due colori, KY-011		Led a due colori KY-029		Led a sette colori KY-034	
Modulo	**bs**	**Modulo**	**bs**	**Modulo**	**bs**
G = ground	-	- = Ground	-	+ = signal	G
R = red	B	R = Red	B	“ “	B
Y = green	G	G = Green	G	- = -	- (ground)
Nota: può essere montato direttamente sullo zoccolo. Usa solo tre connettori.		**Nota:** può essere montato direttamente sullo zoccolo. Usa solo tre connettori.		**Nota:** può essere montato direttamente sullo zoccolo. Usa solo tre connettori. Ribaltato di 180° rispetto a KY-011/KY-029,	

I programmi per i moduli (KY-016, KY-011, KY-016, KY-027, KY-029, KY-034)

Nome del programma:	**Magic_light**
Porte:	**Modulo principale:**
D10 (switch), D9 (led rosso)	**KY-029, magic light cup su LD4.** Questo è un modulo un po’ particolare, perché integra sullo stesso supporto un tilt switch al mercurio e un led.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	In base all’inclinazione del modulo stesso, il led si accende o si spegne.
Note:	Attenzione alla sequenza dei piedini su LD4.
Link:	nessuno

Nome del programma:	**Two_colour_test**
Porte:	**Modulo principale:**
D10, D11	led singoli su porte LD1 ed LD2.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	I led rosso e verde lampeggiano a ritmo alternato.
Note:	Attenzione alla sequenza dei piedini su LD4.
Link:	nessuno

Nome del programma:	**3_led_millis()**
Porte:	**Modulo principale:**
D9, D10, D11	led singoli su porte LD1 LD2 ed LD3, oppure il led a tre colori sullo zoccolo LD4.
Monitor Seriale: no	**Plotter seriale: no**

Scopo del programma: I led a tre colori ampeggiano a diverse frequenze, ma non usano la funzione delay() che interrompe quasiasi operazione, bens millis(), che fa lampeggiare i diversi led indipendentemente, senza stoppare il programma.

Note: I led RGB e SMD hanno una disposizione dei piedini diversa da quella offerta dal connettore L4. Usare un cavo a quattro poli maschi/femmina, facendo attenzione alla polarità.

Link: nessuno

Nome del programma: **3_led_test**

Porte: **Modulo principale:**

D9, D10, D11 led singoli su porte LD1 LD2 ed LD3.

Monitor Seriale: no **Plotter seriale: no**

Scopo del programma: I led a tre colori si accendono con diversi schemi

Note: I led RGB e SMD hanno una disposizione dei piedini diversa da quella offerta dal connettore L4. Usare un cavo a quattro poli maschi/femmina, facendo attenzione alla polarità.

Link: nessuno

Nome del programma: **3_led_pwm_test**

Porte: **Modulo principale:**

D9, D10, D11 led singoli su porte LD1 LD2 ed LD3.

Monitor Seriale: no **Plotter seriale: no**

Scopo del programma: Si sfrutta la potenzialità delle porte PWM, facendo variare progressivamente la luminosità dei led.

Note: I led RGB e SMD hanno una disposizione dei piedini diversa da quella offerta dal connettore L4. Usare un cavo a quattro poli maschi/femmina, facendo attenzione alla polarità.

Link: nessuno

Nome del programma: **SMD_RGB**

Porte: **Modulo principale:**

D9, D10, D11 led singoli su porte LD1 LD2 ed LD3.

Monitor Seriale: no **Plotter seriale: no**

Scopo del programma: Si sfrutta la potenzialità delle porte PWM, facendo variare progressivamente la luminosità dei led. E' stato progettato per i moduli RGB e SMD, ma può essere utilizzato anche per i led singoli o quello a tre colori "nudo".

Note: I led RGB e SMD hanno una disposizione dei piedini diversa da quella offerta dal connettore L4. Usare un cavo a quattro poli maschi/femmina, facendo attenzione alla polarità.

Link: nessuno

Nome del programma: **seven_colour_1**

Porte: **Modulo principale:**

D11 (digitale) **KY-034, seven colour led su LD1.** In questo modulo è inserito un led a tre colori, che però non vengono pilotati singolarmente, ma

	vengono combinati insieme, in modo da formare sette colori in sequenza.
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	Caricando il programma, immediatamente il led lampeggia in sequenza nei sette colori. Questo modulo può essere solo acceso o spento.
Note:	Collegare il piedino “G” al “+” di LD1; collegare “-” tra il modulo e il piedino “-” di LD1
Link:	nessuno

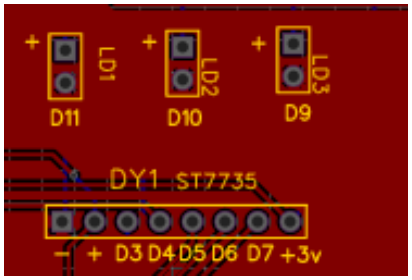
Nome del programma:	**seven_colour_2**
Porte:	**Modulo principale:**
D11 (digitale)	**KY-034, seven colour led su LD1.** In questo modulo è inserito un led a tre colori, che però non vengono pilotati singolarmente, ma vengono combinati insieme, in modo da formare sette colori in sequenza.
Porte:	**Comp. Accessori:**
A2 (analogica)	Pulseante BT1 (integrato sulla basetta)
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	Programma leggermente meno “brutale” del precedente, il led si accende e lampaggia nei 7 colori quando si preme il pulsante “BT1” presente sulla bs.
Note:	Collegare il piedino “G” al “+” di LD1; collegare “-” tra il modulo e il piedino “-” di LD1
Link:	nessuno

I display

Sulla *bs* possono essere alloggiati ben quattro diversi tipi di dislay, alcuni collegati su porte digitali (il TFT 1,77” ST7735, sullo zoccolo DY1 e LCD 1602, sullo zoccolo DY2) e altri alle porte analogiche, gli OLED 128x32 e 128x64, oppure LCD 102 con adattatore I2C che condividono lo

stesso zoccolo, DY3.

Lo zoccolo DY1



Lo zoccolo DY1 si trova all'incirca al centro della basetta, alla destra di Arduino.
Ospita il display TFT 1,7" ST7735

Usa le porte digitali da D3 a D7 (5 porte). Va in conflitto con DY2, ma del resto sulla basetta non possono fisicamente alloggiare insieme, perché andrebbero in contatto.

DY1. Il display a cui fa riferimento è un piccolo video, di 1,8 pollici da 128x64 punti a colori, grafico, che può mostrare una grande numero di informazioni, sotto forma di caratteri e immagini a colori. Per contro, la programmazione è leggermente più impegnativa del display Lcd, ma dato le maggiori possibilità è del tutto plausibile. Questo display è usato in vari programmi a corredo della *bs*.

DY1

Richiede alcune librerie, indicate nei programmi in cui viene usato. Usa molte porte digitali (cinque), tutte condivise anche con lo zoccolo DY2, per cui essi non possono funzionare insieme. Ma questo anche per la disposizione della *bs*: fisicamente non trovano posto entrambi insieme. Nei programmi che si trovano su internet vengono usato in modo del tutto libero le varie porte digitali necessarie, (sebbene molto spesso seguano sempre lo stesso schema), mentre nel nostro sistema dovremo necessariamente usare quelle disponibili, ovvero da D3 a D7. Di seguito verrà illustrato come effettuare la trasposizione.

ST7735

Ecco un programma originale, reperito in Internet:

```
sketch_jan16a §
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <SPI.h>
#define TFT_CS     10
#define TFT_RST    8   t
#define TFT_DC     9
#define TFT_SCLK 13
#define TFT_MOSI 11
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK, TFT_RST);
```

E stato trasposto nel modo seguente:

```
test
#include <Adafruit_GFX.h>    // include Core graphics library
#include <Adafruit_ST7735.h> // include Hardware-specific library
#include <SPI.h>
#define TFT_CS   7
#define TFT_RST  5
#define TFT_DC   6
#define TFT_SCLK 3
#define TFT_MOSI 4
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK, TFT_RST);
```

Tutto il resto del programma è rimasto invariato.
Naturalmente nel programma è necessario dare alcune informazioni, come la posizione e il colore del carattere (o delle immagini e dello sfondo, le posizioni dei caratteri ecc, ma queste informazioni si possono trovare su Internet e sui programmi allegati a questo manuale.

Moduli che usano ilo zoccolo DY1:

Socket DY1 1,8" TFT Display (digital)			1	2	3	4	5	6	7	8
Description	Code	Note	GND	+5v	CLK	DIN	RESET	DC	SPI	+3.3v
Display TFT ST7735	160(RGB)x128		-	+	D3	D4	D5	D6	D7	+
Encoder	KY-040 – HW040		-	+	D3	D4	D5	/	/	/

Come si può vedere, lencoder KY-040, oltre che sullo zoccolo generico DG11, può allogiare anche sui primi cinque pin a sinistra dello zoccolo DY1.

Il display ST7735

Qui di seguito si troveranno alcuni programmi semplicemente didattici, per introdurre questo interessante display.
Verrà usato in molti programmi applicativi.
Specialmente nel programma “test”, molto semplice, si imparerà come inserirlo nei propri programmi.

Progetti per il display ST7735 – TFT 1,7”

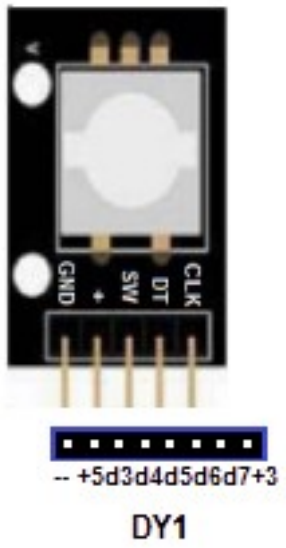
Nome del programma: Test
Porte: **Modulo principale:**

D3/./D7 (digitale)	Display ST7735 TFT 1.77” 160(RGB)x128 Display a colori
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Appare sul display tutta una serie di caratteri, colori e disegni che possono essere visualizzate su questo versatile, piccolo display.
Librerie richieste:	Adafruit_GFX.h; Adafruit_ST7735.h; SPI.h
Note:	Nessuna.
Link:	nessuno

Nome del programma:	**Hello_word**
Porte:	**Modulo principale:**
D3/./D7 (digitale)	Display ST7735 TFT 1.77” 160(RGB)x128 Display a colori
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Appare sul display tutta una serie di caratteri, colori e disegni che possono essere visualizzate su questo versatile, piccolo display.
Librerie richieste:	Adafruit_GFX.h; Adafruit_ST7735.h; SPI.h
Note:	Nessuna.
Link:	nessuno

Encoder KY-040

L’encoder KY-040 permette di misurare con molta precisione uno spostamento, per cui potrebbe essere utile nel misurare lo spostamento di un carrello, per esempio come quello utilizzato sull’asse X, Y o Z di una stampante 3D autocostruita, un robot, ecc. Questo encoder usa 3 porte digitali: D3, D4, D5, pertanto non è compatibile con i display TFT e LCD, che usano rispettivamente gli zoccoli DY1 e DY2. Si potrà usare per mostrare le informazioni i display OLED 128x32 o 128x64 che si collegano allo zoccolo DY3, perché usano delle porte analogiche.
Originariamente l’encoder è stato inserito sullo zoccolo DG11, con la necessità di un connettore. In questo caso, cambiando le porte, l’encoder può alloggiare nativamente sui primi 5 connettori dello zoccolo DY1.



Ecco lo schema di collegamento dell'encoder su DY1:

GND → –
\+ → +
SW → D3
DT → D4
CLK → D5

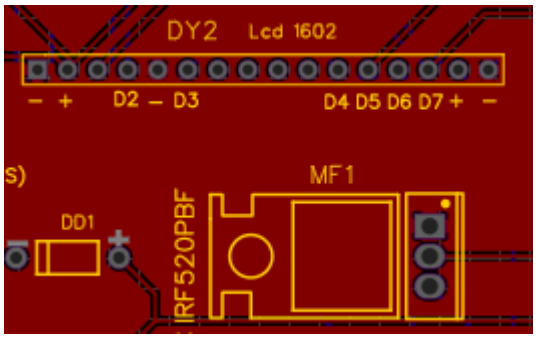
Programmi per Encoder KY-40

Nome del programma: **Encoder_1**
Porte: **Modulo principale:**
D3. D4 , D5 (digitali) **Rotary encoder KY-040**
Monitor Seriale: sì **Plotter seriale: no**
(9600 bit)
Scopo del programma: Girando in senso orario il perno dell'encoder, si vedranno i valori incrementare sul monitor seriale. Ruotando in senso antiorario, diminuiscono. Premendo sul perno, i valori si azzerano.
Note: nessuna
Link: **nessuno**

Nome del programma: **Encoder_2**
Porte: **Modulo principale:**
D3. D4 , D5 (digitali) **Rotary encoder KY-040**
Porte: **Comp. Accessori:**
D9 Led L3 rosso su LD3
D10 Led L2 blu su LD2
Monitor Seriale: sì **Plotter seriale: no**
(9600 bit)
Scopo del programma: Girando n senso orario il perno dell'encoder, si vedranno i valori incrementare sul monitor seriale e il led blu aumenta progressivamente di luminosità. Ruotando in senso antiorario, i valori diminuiscono e la luce del led blu si affievolisce. Se i valori vanno in negativo, progressivamente si accende il led rosso; se tornano verso lo zero, si affievolisce. Premendo sul perno, i valori si azzerano.
Note: nessuna
Link: Nessuno

Nome del programma:	**Encoder_3**
Porte:	**Modulo principale:**
D3. D4 , D5 (digitali)	**Rotary encoder KY-040**
Porte:	**Comp. Accessori:**
A4 (analogiche)	Dispaly LCD 1602 + adattatore I2C su zoccolo AN3
A5	
D5	Servomotore MG90S su zoccolo DG9
Monitor Seriale: sì (9600 bit)	**Plotter seriale: no**
Scopo del programma:	IL servomotore all'avvio si posiziona con l'asse a 90°. Girando n senso orario il perno dell'encoder, si incrementa l'angolo di rotazione del servo, fino a 180°. Ruotando in senso antiorario, si decrementa l'angolo di rotazione del servomotore fino a 0°. Sul display LCD appaiono i dati angolari.
Librerie richieste:	LiquidCrystal_I2C.h; Servo.h
Note:	nessuna
Link:	https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

Lo zoccolo DY2



Lo zoccolo DY2 si trova in posizione centrale della *bs*. Appena sopra di esso, è presente un trimmer per variare la luminosità del Dispslay.

Ospita un display LCD 1602 (16 caratteri x 2 linee, con caratteri bianchi su sfondo blu).

Il 1602 usa 6 porte digitali (da D2 a D7). Quindi va in conflitto con DY1 e con AD1 (se la porta digitale è settata su D2).

Usando un adattatore che lo trasforma in una periferica I2C, può venire collegato sulla porta DY3, liberando così le 6 porte digitali!

DY2. Questo zoccolo usa ben sette porte digitali, da D2 a D7, quindi è piuttosto "affamato" di risorse, poiché le porte che utilizza non possono essere usate contestualmente per altre attività. Il display LCD 1602 è usato in molti progetti, in quanto è flessibile e facile da programmare,grazie anche alla libreria dedicata) e molto versatile. Permette di inserire 16 caratteri di testo su due linee, i caratteri sono bianchi su sfondo blu, quindi ben visibili ed è ideale dove le informazioni devono essere chiare (con un trimmer si regola la luminosità) e non si richiede di mostrare troppe informazioni. E' utilizzato in molti programmi che troverete a corredo della *bs*.

LCD 1602

Anche per LCD 1602 dovremo fare una trasposizione delle porte:

```
#include <LiquidCrystal.h>  /* dichiarazione di utilizzo della libreria  "LiquidCrystal"*/
LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // inizializza la libreria con i numeri delle porte usate
```

Stralcio di un programma trovato su internet, che chiameremo "originale"

Ecco come lo modificheremo per poterlo utilizzare sulla bs:

alphabete

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);
```

Moduli che usano lo zoccolo DY1:

Il connettore DY2 è stato disegnato per ospitare il display LCD 1602, utilizzato in moltissimi programmi, grazie alla sua versatilità e semplicità di programmazione.
Usa le porte digitali da D2 a D7.

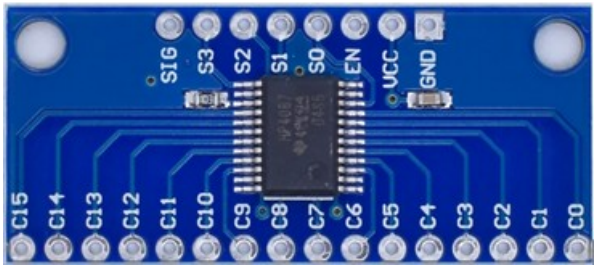
Progetti per il display LCD1602

Nome del programma:	**Alphabete**
Porte:	**Modulo principale:**
D2/./D7 (digitale)	**LCD 1602** - Display Liquid crystal 16x2 (16 caratteri, 2 linee)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Appare sul display tutta la serie delle lettere, dalla “a” alla “z”LiquidCrystal.h
Librerie richieste:	LiquidCrystal.h
Note:	Nessuna.
Link:	nessuno

Nome del programma:	**Auguri**
Porte:	**Modulo principale:**
D2/./D7 (digitale)	**LCD 1602** - Display Liquid crystal 16x2 (16 caratteri, 2 linee)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Un programmino “natalizio”: sul display appaiono f gli auguri di Natale e di buon anno!
Librerie richieste:	LiquidCrystal.h
Note:	Nessuna.
Link:	nessuno

Nome del programma:	**HelloWord_lcd**
Porte:	**Modulo principale:**
D2/./D7 (digitale)	**LCD 1602** - Display Liquid crystal 16x2 (16 caratteri, 2 linee)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sul display allare la tradizionale scritto “Hello Word!” e un contatore incrementale.
Librerie richieste:	LiquidCrystal.h
Note:	Nessuna.
Link:	nessuno

Il multiplexer/demultiplexer (CD74HC4067)



Il CD74HC4067

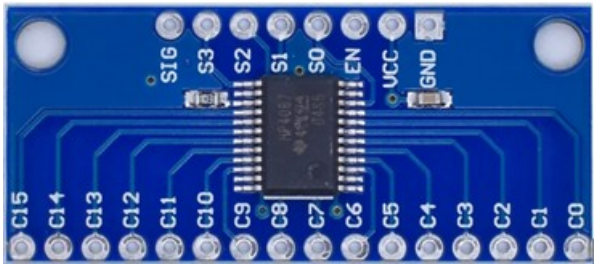
Questo modulo è molto interessante. Uno dei problemi di Arduino è che quando si progettano sketch complessi, il numero delle porte si riduce drasticamente. Il multiplexer permette di espandere le porte disponibili: ben 16, utilizzando solo 5 porte di Arduino! Un bel guadagno… Le porte così ottenute, possono essere tutte digitali o analogiche, utilizzabili sia in ingresso che in uscita.

Come si vede dalle poche righe introduttive, il multiplexer/demultiplexer (abitualmente accorciato in "mux" o "demux" è molto versatile, con solo un paio di limitazioni: all'atto della stesura del programma è necessario stabilire se le porte sono in ingresso o in uscita, analogiche o digitali, e le scelte sono effettive per ***tutte*** e sedici in blocco. Non si possono avere parte di esse in ingresso e altre in uscita, o alcune analogiche e altre digitali.

Inoltre quando si usano in ingresso, è possibile controllare lo stato di una sola porta per volta; abitualmente si inseriscono nei programmi delle routine che permettono di verificarle in rapida sequenza, a una velocità di scansione sufficiente per la maggior parte delle applicazioni. Quando viene usato in uscita (per pilotare dei led, relays, ecc.), si può gestire lo stato di una singola porta per volta, allo stesso modo che un telecomando sceglie un singolo canale; attivando una porta, portandola per esempio in stato "high" disabilita tutte le altre portandole in "low". Naturalmente anche in questo caso si può agire su di esse in rapida sequenza (con la solita routine), come si vedrà anche in qualche programma di test.

Consapevoli di ciò, si possono ideare una notevole serie di programmi; per esempio inserendo delle fotocellule in ingresso, conoscere il percorso di merce o persone; oppure creare un sofisticato antifurto; accendere delle luci in sequenza… dipende solo dalla nostra creatività!

La gestione delle porte.



L'integrato che gestisce le porte è inserito su di una piccola basetta, che permette la gestione ordinata degli ingressi (in alto, 8 pin) e delle uscite, in basso, 16 pin, da C0 a C15.

Sulla destra si può vedere la tabella con il codice binario che permette di gestire le 16 porte. Può sembrare complessa, ma in realtà non è così: ci sono quattro porte digitali di controllo, da **S0 a S3.** In base ai valori "0" (low) o "1" (high), viene attivata la corrispondente porta di ingresso/uscita. **EN** (enable), se è a valore "0", permette il traffico in ingresso/uscita; se si trova al valore logico "1",

La tabella per la gestione delle porte

S0	S1	S2	S3	$\overline{E}$	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

il traffico viene inibito, fino a quando il valore di “EN” non torna a “LOW”. In molti programmi si desidera che la comunicazione sia sempre attiva, e in questo caso si può anche non gestire questa porta.

“**SIG**” è la porta che comunica le informazioni sia in ingresso che in uscita con il microcontroller, per cui è fondamentale che sia gestita. Se SIG viene collegata a una porta analogica di Arduino (da A0 a A7), le sedici porte saranno gestite in modo analogico, con valori da 0 a 1024; altrimenti se collegato a una porta digitale, analogamente i moduli collegati saranno trattati in modo digitale.

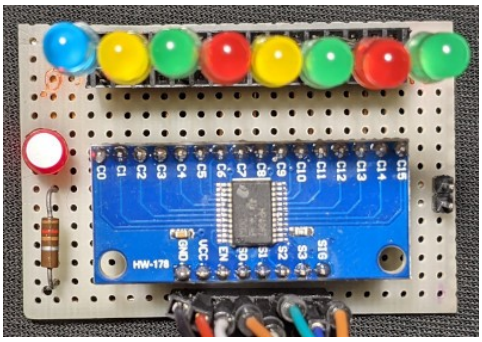
Rimangono solamente “**VCC**” e “**GND**”, il cui uso è intuitivo. Questo modulo può essere alimentato con una tensione fino a +7v; quindi accetta sia i + 3,3v che i +5 v di Arduino; attenzione ai moduli da collegare ad esso, in modo che ricevano la tensione corretta, pena il malfunzionamento o la distruzione del modulo maldestramente collegato! “GND” è la massa.

Con queste informazioni dovremmo essere in grado di poter utilizzare con profitto il nostro mux/demux. **Clicca qui** per visualizzare (***da un sito esterno***) il datasheet di questo interessante modulo.

Una basetta (o due) per il mux/demux

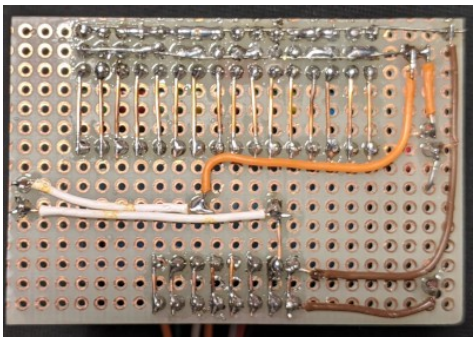
Per rendere più semplice l’uso di questo modulo, che richiede necessariamente un grande numero di cavetti, ho usato per i test un piccolo circuito costruito artigianalmente (ma molto funzionale) su di una basetta millefori (vedi immagine). Ma non tutti hanno una grande destrezza con saldatore e cavetti; per questo motivo ho anche progettato una basetta professionale, che si può vedere – e scaricare i file - dall’apposita pagina delle appendici. **Clicca qui**.

il



prototipo visto frontalmente...

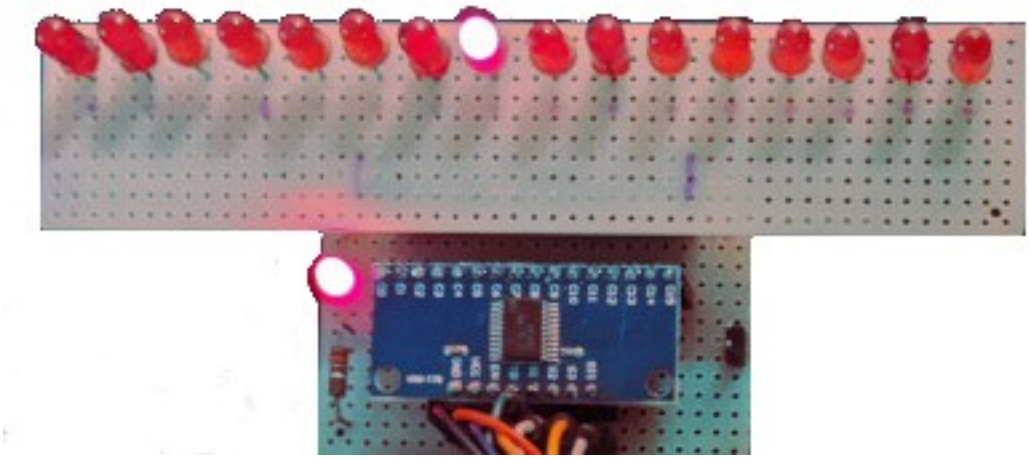
… e



posteriormente

In questa immagine si vede un progetto in modalità “DeMux”, ovvero le porte sono usate in uscita. Sono inseriti solo 8 led (invece di 16) per problema di spazio.

E’ stata inserita una tripla fila di 16 connettori femmine, in cui dalla fila superiore si ricava la massa (-); in quella centrale i 5v e in quella inferiore, numerata da 0 a 15, il segnale che controlla l’attivazione della porta corrispondente. Si è utilizzata questa modalità per poter collegare direttamente o con dei connettori a tre cavi i sensori digitali e analogici a tre pin, come per esempio il sensore di temperatura e umidità DHT11.



Questa seconda basetta, sempre costruita sperimentalmente su millefori, ospita 16 led, collegati con tre file di connettori maschi sulla basetta su cui è inserito il multiplexer. In questo modo può utilizzare i 16 led senza l'uso di ulteriori cavi.

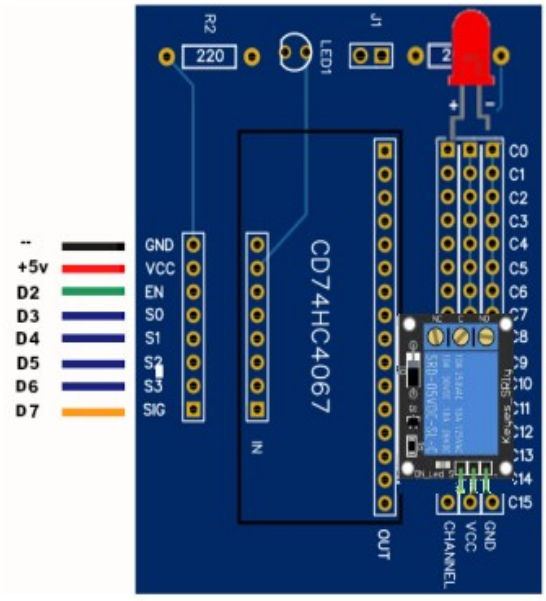
Altri programmi con il multiplexer insieme a una keypad (tastiera) sono stati inseriti al paragrafo relativo a DG11. **Clicca qui** per visualizzarli.

Clicca qui per andare alla pagina della basetta per ilmultiplexer.
Clicca qui per andare alla pagina della basetta per i 16 led da collegare al multiplezer

I collegamenti verso Arduino e verso l'esterno

Nelle immagini successive si vedranno le connessioni in dettaglio sia per collegare dei sensori (ingresso, modalità mux) che attuatori (uscita, modalità demux).

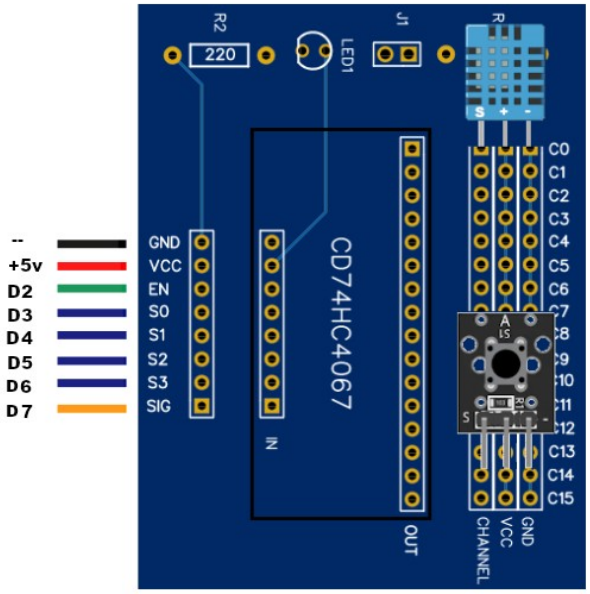
Fortunatamente la libreria utilizzata per il CD74HC4067 lascia la libertà di quali porte digitali per collegare sia le porte di controllo (da S0 a S3), quella di attivazione (EN, da usare solo se necessario, per disattivare temporaneamente il modulo), che infine quella di scambi dati "SIG", che può essere collegata sia a una porta analogica che digitale, in base alle esigenze.



Il modulo in configurazione "DeMux" (uscita)

Il modulo è inserito nel riquadro nero, centrale, denominato "CD74HC4067. Sulla sinistra sono presenti i collegamenti verso Arduino Nano, mentre sulla destra le sedici uscite. Come si vede, per ogni uscita sono presenti tre pin. Partendo da sinistra, denominati "Channel", Vcc e Gnd, in modo da poter collegare una vasta gamma di moduli.
Su "C0", il primo in alto, è connesso un led, che naturalmente usa solo due connettori, Channel (anodo) e Gnd (catodo).
In basso, su "C15" è connesso un relay KY-019, che usa i tre connettori: "S "su channel, "+ "su Vcc e "–" su Gnd.

In questo riquadro, sono inseriti due sensori di entrata: un sensore di temperatura e umidità, DHT11 e un pulsante, KY-004.
La scelta non è stata casuale: infatti entrambi possono essere utilizzati come moduli analogici, che digitali.
Nel caso si opti per il primo caso, il pin "Sig" deve essere collegato a una porta ***analogica*** di Arduino; nel secondo, a una porta ***digitale***.
Naturalmente in uno stesso sketch, i sensori devono essere configurati allo stesso modo: o tutti analogici o tutti digitali; non è possibile una configurazione mista.
Anche in questo caso, "S " deve essere collegato su channel, "+ "su Vcc e "–" su Gnd.



Il modulo in configurazione "Mux" (ingresso)

In questa basetta, oltre ai collegamenti verso Arduino e verso i moduli esterni, si nota un led e una resistenza R2, da 220 Ohm. Non erano indispensabili, ma visto che il CD74HC4067 non ha a bordo alcun led che ne indichi il funzionamento, li ho inseriti per comodità.

Sulla stessa basetta c'è anche un jumper "J1" e una seconda resistenza seminascosta, R1, sempre da 220 Ohm. Questa resistenza è stata inserita per precauzione: se il piedino "Vcc" viene collegato ai +5v di Arduino, e si utilizzano dei led come attuatori in uscita, essi verrebbero alimentati con una tensione troppo alta e brucerebbero rapidamente. Così i +5v sono filtrati rispetto all'alimentazione dei led attraverso R1, di un valore adeguato per preservare i led. In questo caso, J1 deve restare aperto, non ponticellato. Ma se si collegano dei sensori che funzionano a +5v, specialmente quelli analogici, con la resistenza inserita non potrebbero ricevere la la tensione nominale, falsando le letture. Il jumper J1 ha proprio questa funzione: quando ponticellato, cortocircuita la resistenza e quindi i moduli vengono alimentati a +5v.

Programmi per CD74HC4067

Configurazione DeMux (attuatori in uscita)

Nome del programma:	**Multiled_1**
Porte:	**Modulo principale:**
EN = 2; S0 = 3; S1 = 4; S2 = 5; S4 = 6; SIG = 7 (Dig.)	Multiplexer CD74HC4067
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno è inserito un led. In base al valore che si inserirà in Yselect(x) (x compreso tra 0 e 15), si accenderà il led corrispondente. Questo sketch può essere usato come routine in programmi più strutturati. Naturalmente al posto dei led si possono usare relay o altri attuatori.
Note:	Vedi nota a fondo pagina
Link:	nessuno

Nome del programma:	**Multiled_2**
Porte:	**Modulo principale:**
EN = 2; S0 = 3; S1 = 4; S2 = 5; S4 = 6; SIG = 7 (Dig.)	Multiplexer CD74HC4067
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno è inserito un led. Per mezzo di una routine, i led si accendono in rapida successione, da 0 a 15, e poi da 15 a 0, ricordando un poco lo scanner di Supercar (una famosa serie di telefilm degli anni '80).
Note:	Vedi nota a fondo pagina
Link:	nessuno

Nome del programma:	**Multiled_3**
Porte:	**Modulo principale:**
S0 = 2; S1 = 3; S2 = 5; S4 = 10; SIG = 13 (Digitali)	Multiplexer CD74HC4067
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
A4, A5 (analogiche)	Display Oled 128 x 32 su DY3
D9	Buzzer KY-006 (o KY-012) su BZ1
Scopo del programma:	Programma simile al precedente. Oltre al lampeggio dei led, un buzzer segnala i passaggi con l'emissione di una nota e sul display appare il canale usato, oltre al codice binario corrispondente.
Note:	Vedi nota a fondo pagina
Link:	nessuno

Configurazione Mux (attuatori in entrata)

Nome del programma:	**Multi_input1**
Porte:	**Modulo principale:**
EN = 2; S0 = 3; S1 = 4; S2 = 5; S4 = 6; SIG = 7 (Dig.)	Multiplexer CD74HC4067
Porte:	**Comp. Accessori:**
C0/./C15 (multiplexer)	Switch KY-004 (o qualsiasi altro sensore digitale a tre piedini)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno può essere inserito un sensore digitale. In base al valore che si inserirà in Yselect(x) (x compreso tra 0 e 15), il monitor seriale restituisce un valore "0" oppure "1". Questo sketch può essere usato come routine in programmi più strutturati.
Note:	Vedi note a fondo pagina.
Link:	nessuno

Nome del programma:	**Multi_input2**
Porte:	**Modulo principale:**
EN = 2; S0 = 3; S1 = 4; S2 = 5; S4 = 6; SIG = 7 (Dig.)	Multiplexer CD74HC4067
Porte:	**Comp. Accessori:**
C0/./C15 (multiplexer)	Switch KY-004 (o qualsiasi altro sensore digitale a tre piedini)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno può essere inserito un sensore digitale. Una routine esegue una scansione di Yselect(x) (x compreso tra 0 e 15), ovvero di tutte le porte. Il monitor seriale restituisce un valore "0" oppure "1", in base al sensore, se attivo o in attesa
Note:	Vedi note a fondo pagina.
Link:	nessuno

Nome del programma:	**Multi_input3**
Porte:	**Modulo principale:**
EN = 2; S0 = 3; S1 = 4; S2 = 5; S4 = 6; SIG = 7 (Dig.)	Multiplexer CD74HC4067
Porte:	**Comp. Accessori:**
C0/./C15 (multiplexer)	Switch KY-004 (o qualsiasi altro sensore digitale a tre piedini)
D11 (digitale)	Led su LD1
D10 (digitale)	Led su LD2
D9 (digitale)	Led su LD3
D13 (digitale)	Led su LD5 (solo su bs 0.6.1), sulla vecchia bs 0.1, inserire il led con una resistenza su DG11 oppure su BZ1 (su porta 13)
A4, A5 (analogiche)	Display LCD 1602 con adattatore I2C. Su AN4 (scheda 0.5.1) Su AN4 oppure su DY3b per la nuova scheda 0.6.1.
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Il multiplexer ha sedici canali, in corrispondenza di ognuno può essere inserito un sensore digitale. Una routine esegue una scansione di Yselect(x) (x compreso tra 0 e 15), ovvero di tutte le porte. Il monitor seriale restituisce un valore "0" oppure "1", in base al sensore, se attivo o in attesa. Le stesse informazioni possono essere visualizzate sul visore LCD, oltre che ai valori binari del canale scelto. Lo stesso valore binario, è mostrato dall'accensione o lo spegnimento dei quattro led, che in questo caso, rappresentano un bit ciascuno.
Note:	Vedi note a fondo pagina.
Link:	nessuno

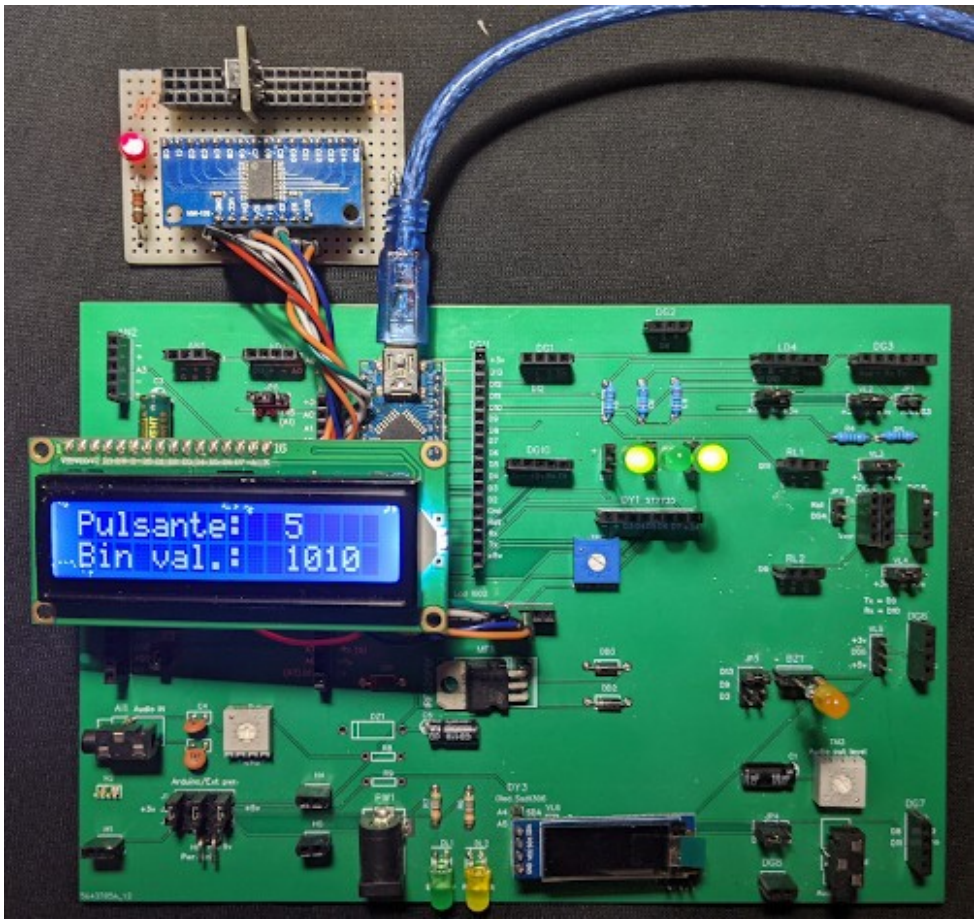


Immagine relativa al programma Multi_input3

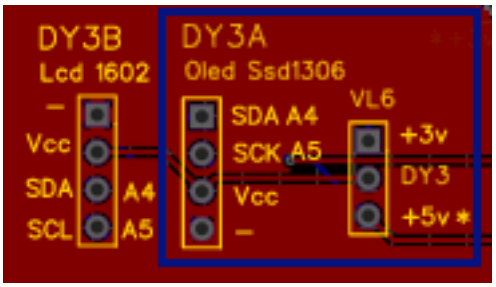
Note comuni ai tre programmi precedenti:

1 - Alcuni sensori tipo switch passano dallo stato “0” a “1” se sono attivi; altri si comportano in modo opposto. Modificare il programma in base a queste riflessioni. Di default il sensore è in stato “0” quando attivato.

2 - Se si modifica leggermente il programma, collegando “SIG” a una porta analogica (per esempio “A7”), si potranno collegare al multiplexer dei sensori analogici (a tre piedini). In questo caso i valori saranno compresi tra “0” e “1024” ($2^{\mathbf{10}}$). Non mischiare sensori analogici e digitali.

3 - usando queste porte per il multiplexer, esso si può collegare con successo allo zoccolo DY2.

Lo zoccolo DY3/a



Lo zoccolo DY3a-b si trova in posizione centrale della bs, in basso.
Esso può ospitare due dispaly OLED: 128x32 e 128x64. (Vedi info più sotto).
Questo display non va in conflitto con DY1 e DY2, perché usa due porte analogiche un po' speciali: A4 e A5 e in genere non va in conflitto con altri moduli che usino le stesse porte, come per esempioAN3 e AN4; è sufficiente che i vari moduli abbiano indirizzi interni differenti.
Usando un adattatore che lo trasforma in una periferica I2C, può venire collegato sulla porta DY3b, liberando 6 porte digitali.

Moduli che usano ilo zoccolo DY3/a:

Socket DY3/a Oled Dispaly IC (analogic)			1	2	3	4
Description	Code	Note	SDA	SLC	+5v	GND
Display OLED monocrome (white)	128x32		A4	A5	+	-
Display OLED Yellow-Blue	128x64		A4	A5	+	-

DY3/a. I display OLED che alloggiano su questo zoccolo utilizzano due porte analogiche: A4 e A5. Queste porte, che possono essere usate allo stesso modo delle altre porte digitali, hanno anche una funzione speciale: A4 ha la funzione SDA (data) e A5 quella di SCL (o SCK), ovvero di clock. La particolarità è che a queste porte possono essere collegati più moduli contemporaneamente, a patto che abbiano indirizzi diversi. Infatti questi display, come il modulo RTC (clock), BMP280 e altri, possiedono un indirizzo dato di fabbrica, che può essere individuato con un programma apposito, e in caso di conflitti può anche essere variato. A questo zoccolo possono essere collegati alternativamente due diversi display della stessa famiglia, con sigla SSD1306 o SSD1307, che differiscono per il numero di punti che gestiscono: 128x32 controlla (ovviamente!) 128 x 32 punti ed è monocromatico (normalmente bianco); mentre 128x64 permette di controllare il doppio di punti in altezza. Ha un'altra particolarità: la prima fila in alto, larga 16 pixel, è di colore giallo; quelli successivi sono blu, permettendo di distinguere, anche a colpo d'occhio, i diversi tipi di informazione. I programmi scritti per l'uno possono essere usati, con minime variazioni anche per l'altro; infatti i programmi che verranno proposti come esempio sono sostanzialmente identici e presentano semplicemente minimi aggiustamenti. I display sono molto piccoli, ma chiaramente leggibili e possono essere utilizzati con profitto in tutti i casi in cui si abbia poco spazio a disposizione, oppure che siano occupate le porte digitali necessari per gli altri display. Inoltre per la particolare tecnologia OLED generano loro stessi una buona luminosità e non necessitano, come quelli LCD o TTF, di essere retroilluminati. Il consumo di energia è del tutto trascurabile.

128x32

128x64

I programmi per i display

Nome del programma:	**I2C_scanner**
Porte:	**Modulo principale:**
A4 – SDA; A5 - SCK (analogico)	**Moduli IC2**. Modulo che restituisce temperatura, pressione e fa un calcolo approssimativo dell'altitudine.
Porte:	**Comp. Accessori:**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Mostra sul monitor seriale l'indirizzo del modulo. Sulle porte SDA/SCK (IC2) possono essere collegati più moduli, sempre che abbiano indirizzi diversi. Adatto anche per altri moduli IC2.
Note:	nessuna
Link:	nessuno

Il display OLED SSD1306 128x 32 (128 x 32 pixel)

Questo display è veramente minuscolo: il realtà è leggermente più piccolo di come lo si vede in questa foto. A differenza dei due precedenti non è retroilluminato, ma è grazie alla tecnologia utilizzata che emette luce. E' monocromatico (bianco), e usa solo due piedini per il segnale e si collega alle porte analogiche A4 e A5, quindi lascia libere tutte le porte digitali, che possono essere utilizzate per vari moduli. Le sue dimensioni lo rendono molto versatile, anche per progetti miniaturizzati.

Nome del programma:	**SSD1306_32_1**
Porte:	**Modulo principale:**
A4 SDA A5 SCK (analogico)	**SSD1306** -Dispaly Oled 128x32 pixel
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sul display appare la tradizionale scritto "Hello Word!"
Librerie necessarie:	Adafruit_GFX; Adafruit_SSD1306
Librerie richieste:	Adafruit_GFX.h; Adafruit_SSD1306.h
Note:	Nessuna.
Link:	Nessuno

Nome del programma: **SSD1306_32_2**
Porte: **Modulo principale:**
A4 SDA **SSD1306** -Dispaly Oled 128x32 pixel
A5 SCK (analogico)
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Sul display appaiono una serie di figure geometriche e di caratteri, per verificare le potenzialità dei display
Librerie necessarie: Adafruit_GFX; Adafruit_SSD1306
Note: Nessuna.
Link: nessuno

Il display OLED SSD1306 128x64

I programmi seguenti sono stati semplicemente modificati per adattarsi all'altrettanto piccolo visore OLED SSD1306 128x64 (128 x 64 pixels), che a differenza del precedente non genera immagini semplicemente di colore bianco, ma visualizza una prima fila di pixel in alto di colore giallo, mentre in tutto il resto dello schermo il colore è blu. A parte il numero diverso di pixel e i colori, per il resto i due display sono perfettamente compatibili e le modifiche richieste sono veramente modeste.

Nome del programma: **SSD1306_64_1**
Porte: **Modulo principale:**
A4 SDA **SSD1306** -Dispaly Oled 128x64 pixel
A5 SCK (analogico)
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Sul display appare la tradizionale scritto "Hello Word!"
Librerie necessarie: Adafruit_GFX.h; Adafruit_SSD1306.h
Note: Nessuna.
Link: nessuno

Nome del programma: **SSD1306_64_2**
Porte: **Modulo principale:**
A4 SDA **SSD1306** -Dispaly Oled 128x64 pixel
A5 SCK (analogico)
Monitor Seriale: sì **Plotter seriale: no**
(9600 baud)
Scopo del programma: Sul display appaiono una serie di figure geometriche e di caratteri, per verificare le potenzialità dei display
Librerie necessarie: Adafruit_GFX.h; Adafruit_SSD1306.h
Note: Nessuna.
Link: nessuno

Lo zoccolo DY3/b

Lo zoccolo DY3 è stato sdoppiato in DY3/a (per i display OLED) e DY3/b (display LCD 1602 con adattatore I2C). Tutti questi display usano le stesse porte, ma con disposizione diversa. Per comodità quindi sono state sdoppiate. Il jumper di alimentazione VL6 è attivo per entrambi gli zoccoli.

Naturalmente i display possono alloggiare su entrambi gli zoccoli, se necessario usando dei connettori a quattro cavi, connettendoli con la piedinatura corretta.

Moduli che usano ilo zoccolo DY3/b:

Socket DY3/b Oled Dispaly IC (analogic)			1	2	3	4
Description	Code	Note	SDA	+5v	SDA	SLC
Display LCD	1602A		-	+	A4	A5

Progetti per display LCD 1602 con adattatore I2C

Il display LCD 1602 "base" è già stato analizzato per lo zoccolo DY2. Il problema è che in questa configurazione, il dispaly richiede ben sei porte digitali; invece se abbinato all'adattatore IC2, richiede solo due porte analogiche: A4 SDA e A5 SCL.

Per collegare il, display con l'adattatore I2C, è necessario usare un cavetto a quattro connettori, perché ha la piedinatura diversa dai display SSD1306.

Il display adattatore può anche essere connesso sullo zoccolo AN3.

Cambiando via hardware l'indirizzo interno del modulo, più apparati I2C possono essere collegati sulle stesse porte. Ecco come impostare gli indirizzi:

Effettuando i ponticelli su A0, A1, A2 (vedi rtabella nella figura successiva) è possibile modificare l'indirizzo del display per evitare confitti con altri moduli che utilizzano il protocollo I2c.

Ecco la tabella:

Indirizzo	A0	A1	A2
0x20	chiuso	chiuso	chiuso
0x21	aperto	chiuso	chiuso
0x22	chiuso	aperto	chiuso
0x23	aperto	aperto	chiuso
0x24	chiuso	chiuso	aperto
0x25	aperto	chiuso	aperto
0x26	chiuso	aperto	aperto
0x27	aperto	aperto	aperto

Utilizzando un adattatore I2C, è possibile collegare a questo connettore anche un display LCD 1602 alle porte analogiche A4 e A5, liberando quindi ben 6 porte digitali, che possono essere utilizzate per inserire altri moduli digitali. Questo adattatore permette, saldando alcuni ponticelli, di vedersi attribuiti diversi indirizzi, in modo da non andare il conflitto con altri eventuali moduli che usano il protocollo I2C.

Programmi per il display LCD 1602 con adattatore I2C

Nome del programma: I2C_scanner
Porte:
A4 – SDA;
A5 - SCK (analogico)
Porte:
Monitor Seriale: sì (9600 baud)
Modulo principale:
Moduli IC2. Modulo che restituisce temperatura, pressione e fa un calcolo approssimativo dell'altitudine.
Comp. Accessori:
Plotter seriale: no
Scopo del programma: Mostra sul monitor seriale l'indirizzo del modulo. Sulle porte SDA/SCK (IC2) possono essere collegati più moduli, sempre che abbiano indirizzi diversi. Adatto anche per altri moduli IC2.
Note: nessuna
Link: nessuno

Nome del programma: LCD_i2c_1
Porte:
A4 SDA
A5 SCK (analogico)
Monitor Seriale: sì (9600 baud)
Modulo principale:
LCD 1602 + adattatore I2C
Plotter seriale: no
Scopo del programma: Sul display appaiono una serie di scritte
Librerie necessarie: LiquidCrystal_I2C.h
Note: Nessuna.
Link: https://win.adrirobot.it/display_lcd/display-lcd-i2c-16x2-con-retroilumminazione-blu.htm

Nome del programma:	**LCD_i2c_2**
Porte:	**Modulo principale:**
A4 SDA A5 SCK (analogico)	**LCD 1602 + adattatore I2C**
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sul display appaiono una serie di scritte scorrevoli
Librerie necessarie:	LiquidCrystal_I2C.h
Note:	Nessuna.
Link:	https://www.meccanismocomplesso.org/lcd1602-utilizzare-un-display-a-cristalli-liquidi-lcd-con-arduino-tramite-i2c/

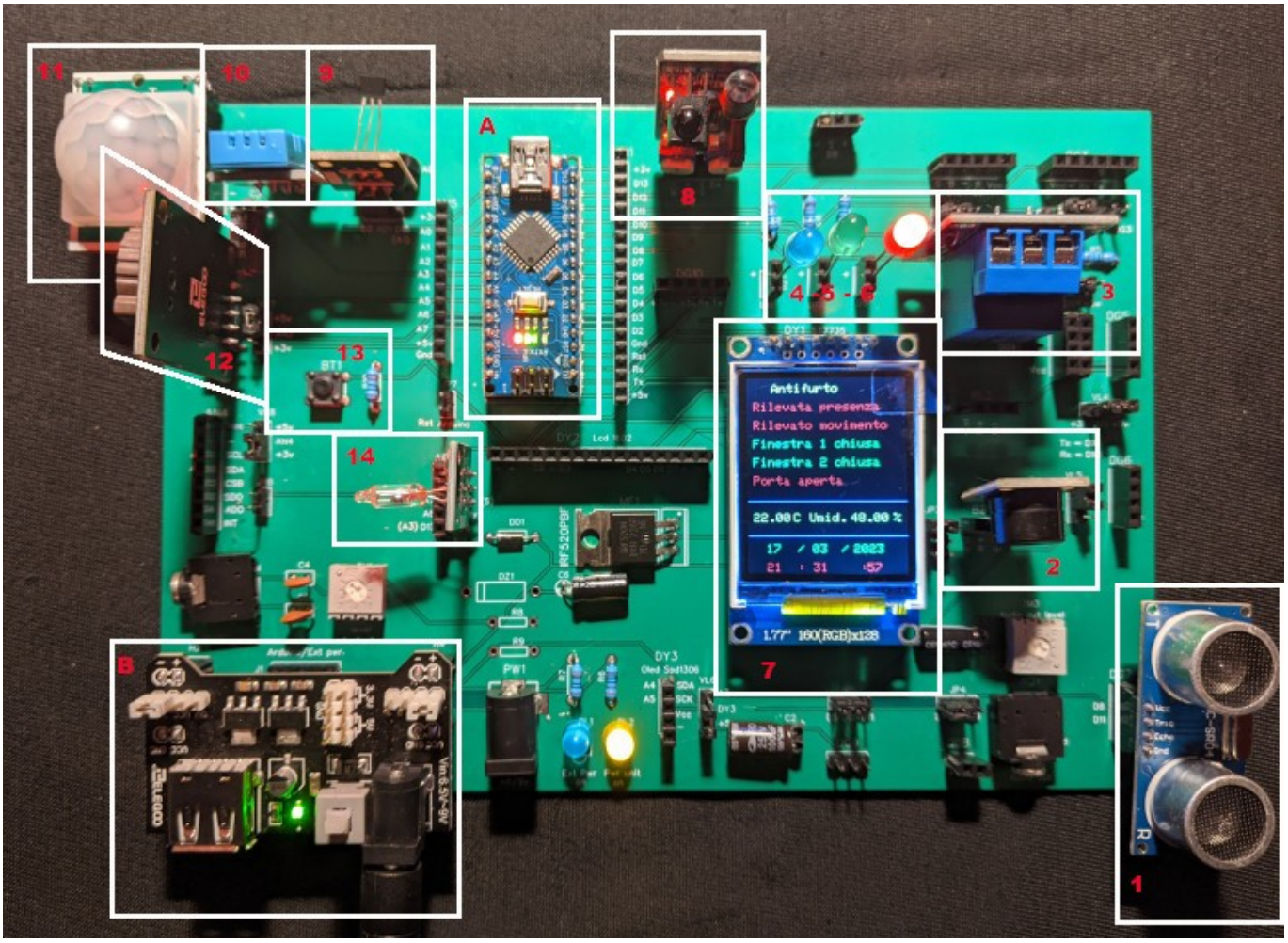
Nome del programma:	**LCD_i2c_3**
Porte:	**Modulo principale:**
A4 SDA A5 SCK (analogico)	**LCD 1602 + adattatore I2C**
Porte:	**Comp. Accessori:**
D10(RST), D9 (DAT), D7 (CLK)	RTC clock DS1602 su DG12
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Sul display appare data, ora, giorno della settimana
Librerie necessarie:	LiquidCrystal_I2C.h
Note:	Nessuna.
Link:	https://www.meccanismocomplesso.org/lcd1602-utilizzare-un-display-a-cristalli-liquidi-lcd-con-arduino-tramite-i2c/

Programmi con più sensori e vari zoccoli

Naturalmente molti dei programmi presentati in questa sezione usano vari moduli, allocati su più zoccoli, ma in linea di massima è sempre possibile individuare quale sia il modulo ***principale***, e quindi legarlo a un determinato zoccolo.

Programma per antifurto

Il programma che segue, denominato "***antifurto***", oltre a utilizzare ben 14 componenti, tra moduli vari e led, non contiene un modulo più rilevante di altri, e quindi il programma, anche per la sua complessità, viene trattato a parte.



Ecco l'immagine della realizzazione pratica dell'antifurto.

Per realizzare questo progetto, abbiamo usato i seguenti moduli, come già indicato nell'introduzione:

1. HC-SR04, un sensore ultrasonico di movimento e prossimità, anche questo modulo ha un raggio di azione di qualche metro. Collegato sullo zoccolo DG7;
2. un buzzer (piccolo altoparlante piezoelettrico), che funge da allarme in caso di rilevazioni anomale da parte degli altri sensori; presenta anche un'uscita audio, per collegarsi a un amplificatore esterno. Su BZ1.
3. un relay che si attiva in caso si verifichi qualunque anomalia rilevata da uno o più sensori; su RL2;
4. un led blu che lampeggia, per indicare il funzionamento del sistema; su LD1;
5. un led verde, che segnala la mancanza di anomalie, o nel caso si siano verificati degli allarmi, si accende dopo il reset del sistema, ottenuto premendo BT1. Il led alloggia su LD2.
6. un led rosso, che si attiva insieme al buzzer e al relay in caso di anomalie, segnalate dai sensori. Su LD3;
7. un display TFT grafico a colori, per mostrare le informazioni raccolte dai vari sensori;
8. Un sensore "obstacle avoiance", KY-032. In genere viene usato per rilevare ostacoli, ma in questo progetto viene utilizzato per verificare la chiusura di una finestra. Zoccolo DG1.
9. KY-003, un sensore per l'effetto hall, per verificare la chiusura di una porta. Inserito nello zoccolo AD1, alloggiato verso sx, sul pin digitale;
10. Un sensore di umidità e temperatura, DHT11, alloggiato sullo zoccolo AN1;
11. Pir HC-SR501, è un sensore passivo a raggi infrarossi, e rivela la presenza di persone, animali e oggetti che emettono calore, nel raggio di alcuni metri. Collegato allo zoccolo AN2, di cui usa i tre pin centrali (attenzione alle polarità!);
12. un modulo datario/orologio, DS1307, collegato su AN3;
13. un pulsante di reset, alloggiato direttamente sulla basetta, BT1;
14. Un sensore a bolla di mercurio, KY-017, usato per verificare l'apertura di una finestra; alloggia sullo zoccolo AD2, di cui usa i tre pin superiori;

A. naturalmente un Arduino Nano;

B. una power unit, perché sulla *bs* ci sono molti componenti, e quindi un rilevante assorbimento di corrente.

Ora analizziamo questo progetto più nel dettaglio, sia per la complessità, che per il suo aspetto didattico.

I moduli 1, 8, 9, 11, 14, sono i sensori che rappresentano il "cuore" del progetto, perché sono i sensori attivi che registrano le variazioni delle situazioni del nostro ipotetico appartamento. Volutamente sono tutti con caratteristiche diverse, in modo da poter spaziare tra varie possibilità.

I moduli 1 e 11, ovvero HC-SR04, sensore di movimento e di prossimità (DG7, porte D8 e D11) e il PIR, sensore a raggi infrarossi (zoccolo AN2, porta A3, ma usata come digitale, ovvero D17), possono controllare due porzioni di locali, con un raggio di alcuni metri, come per esempio la zona ingresso e in prossimità di una porta-finestra che dà su di un giardino o un balcone, per verificare se ci sono movimenti sospetti e imprevisti che possono indicare un'intrusione.

I moduli 8,9,14, sono legati ad aperture impreviste di porte o finestre. "Obstacle avoiance" (zoccolo DG1, porta D12), integra nella stessa scheda un trasmettitore e un ricevitore a 38 Khz, con un raggio di azione di pochi centimetri. Può essere posizionato sopra una finestra, quando la finestra è serrata, il circuito rimane chiuso; ma quando la finestra viene aperta senza consenso, il ricevitore non riceve più un segnale, il circuito si apre e scatta l'allarme. Anche il secondo modulo, un semplice sensore a bolla di mercurio KY-017 (zoccolo AD2, porta A7), è posizionato sopra una seconda finestra, appoggiato sopra il bordo e libero di ruotare. Quando la finestra è chiusa, il sensore è orizzontale e la bolla di mercurio mantiene chiuso il circuito. Se la finestra viene forzata, il sensore non è più appoggiato e per gravità ruota verso il basso; la bolla di mercurio non mantiene più chiuso il sistema, e ancora una volta scatta l'allarme. ***Nota 1***: KY-017 ha tre piedini, mentre lo

zoccolo AD2 può alloggiare un sensore a cinque piedini. Esso va montato verso la parte alta dello zoccolo. A destra dello zoccolo stesso appaiono i codici relativi ai vari piedini; "S" del sensore deve corrispondere a "S" dello zoccolo. ***Nota 2***: il sensore a mercurio può essere sostituito la un qualunque sensore digitale, come un tilt o shock sensor, con due stati digitali: o aperto o chiuso. L'ultimo sensore utilizzato, KY-003 (zoccolo AD1, porta D2) è a effetto Hall (magnetico): Si è ipotizzato di sistemarlo sopra la porta di ingresso. La porta stessa ha fissato sul bordo superiore un magnete, in asse con il sensore. Con lo stesso principio dei due precedenti, quando la porta viene aperta, il sensore non riceve più il campo magnetico, si apre e scatta l'allarme. ***Nota***: KY-003 ha tre piedini, ma viene montato sullo zoccolo AD1, che possiede un alloggiamento per quattro piedini. Guardandolo da davanti, viene montato tutto a destra, ovvero il piedino "S" del sensore deve corrispondere a quello contrassegnato DO" sullo zoccolo. Anche questo sensore può essere sostituito con qualsiasi altro che abbia una funzione di switch, come un opto-sensore KY-010.

Ma quando i sensori ricevono un'anomalia, cosa succede? Accadono diverse cose:

- i led di segnalazione cambiano stato. Quando tutto è regolare, il led verde posizionato sullo zoccolo LD2 (porta D10) è acceso. Se scatta uno o più allarme, esso si spegne e si accende il led rosso, su LD3 (porta D9). Il led azzurro, posizionato su LD1, porta 11, semplicemente lampeggia qualunque cosa accada, indicando il buon funzionamento del sistema;
- contemporaneamente all'accensione del led rosso, si attiva anche il relay posto su RL2, che condivide la stessa porta del led (D9). L'attivazione del relay può far scattare un allarme locale o remoto (telecontrollo), bloccare le uscite, accendere le luci, o intraprendere qualunque altra attività;
- contestualmente il buzzer (zoccolo BZ1, porta D13) emette una nota a 1000 Hz. Poiché è collegato ad un uscita audio standard, la stessa nota amplificata può essere emessa da un altoparlante collegato a un amplificatore esterno;
- sul display viene segnata l'anomalia relativa al sensore (o ai sensori) interessati. Quando tutto è regolare, sul display appaiono le notizie relative ai vari sensori in verde. Quando accade qualcosa di irregolare, cambia la scritta relativa ed essa appare in colore rosso.

Sul display appaiono anche altre informazioni, quali:

- temperatura e umidità, grazie a DHT11, posto sullo zoccolo AN1 (porta A1, usata però come digitale, ovvero D15);
- la data e l'ora attuale, elaborate da DS1307, un orologio (clock) digitale, alloggiato su AN3 (porte A4 e A5).

Ma se un eventuale malintenzionato, accortosi dell'antifurto, decidesse di richiudere la finestra e di fuggire? L'antifurto continuerebbe a restare in allarme; il quale verrebbe riportato alla situazione iniziale solamente con la pressione del pulsante BT1 (porta A2, digitale D16) presente sulla basetta, che ha la funzione di reset. In questo caso, tutte le scritte tornano in verde, si spegne il led rosso e si riaccende quello verde; il relay viene diseccitato e il buzzer termina di produrre la nota di allarme. Sviluppando ulteriormente il progetto si potrebbe anche resettare il sistema a tempo, per esempio dopo cinque minuti, ma queste sono scelte da valutare caso per caso.

E tutto questo sistema , per essere semplicemente testato, viene montato sulla nostra *bs* sperimentale, senza l'uso di alcun filo, ed eseguito molto velocemente. Nel caso si ritenga funzionale questo progetto, terminati i vari test, effettuate le modifiche relative ai sensori e ottimizzato il programma e si desiderasse utilizzarlo realmente, si potrebbe costruire un circuito ad hoc su una basetta specifica, oppure utilizzare la nostra basetta, semplicemente collegando ad essa i cavi proveniente dai vari sensori.

Come si evince dalle descrizioni del programma, si nota che alcuni sensori digitali sono inseriti in porte nominalmente analogiche, che però in questo progetto vengono gestite virtualmente in modo digitali. Se si è interessati a maggiori informazioni, leggi la nota relativa.

Dopo aver letto la spiegazione dettagliata del progetto, vediamo la scheda sintetica, compilata nel solito modo.

Progetto antifurto didattico

Nome del programma:	**antifurto**
Porte:	**Moduli:**
A7 (analogico)	Mercury tilt sensor, KY-017
A3 → D17 (digitale)	PIR, HC-SR501
A1 (analogico)	DHT11, sensore di temperatura e umidità
D9 (digitale)	Relay, KY-019
D9 (digitale)	Led rosso, su LD3
D10 (digitale)	Led verde, su LD2
D11 (digitale)	Led blu, su LD1
D8 e D11 (digitali)	Sensore a ultrasuoni, HC-SR04
D3/./D7 (digitali)	Display TFT a colori ST7735
D2	Sensore effetto Hall (magnetico) KY-003
D12	Avoiance obstacle, KY-032
D13	Buzzer, KY-006 (oppure KY-012)
A2 → D16	Button KY-004
A4, A5	RTC DS1307 su AN3
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Questo è un progetto di antifurto piuttosto articolato. Leggere le informazioni dettagliate esposte nelle pagine precedenti.
Librerie necessarie:	Adafruit_GFX.h; Adafruit_ST7735.h; Wire.h; RTClib.h; SPI.h; dht.h
Note:	Usando un relay e molti moduli, è utile alimentare esternamente Arduino, con un alimentatore da 6/./9 volt in grado di fornire 500/1000 mA.
Link:	nessuno

Una piccola drum machine

Questo progetto, trovato su internet è molto interessante: il nostro Arduino si trasforma in una piccola, ma perfetta drum machine! Qualcuno, con molta pazienza, ha campionato i suoni del Minipops della Korg, una vecchia drum machine. Oggi questo strumento appare del tutto inadeguato, ma negli anni '70 del secolo scorso ebbe una certa fortuna: fu addirittura utilizzato da Michael Jarre per la famosissima Oxigene!
Ora la possiamo testare sul nostro piccolo Arduino Nano.
E' necessario eseguire qualche settaggio sulla nostra bs:

- il potenziometro che sarà collegato su AN2, deve essere collegato alla porta A5, spostando il jumper JP9 e regola la velocità di esecuzione;
- l'altro potenziometro è connesso su AN3 ed è necessario fornire tensione sull'ultimo piedino in basso, contrassegnato "SCW"; settare il jumper VL10. L'uscita audio è tassativa su D11. Si può collegare un buzzer sullo zoccolo RL1, oppure collegare con un cavetto un mini-altoparlante oppure l'ingresso di un amplificatore audio.

Dopo il test, settare nuovamente i jumper interessati in posizione standard, specialmente togliendo il ponticello dal jumper VL10; altrimenti si rischia di alimentare il modulo collegato in modo scorretto, danneggiandolo.

Nome del programma:	**Minipops**
Porte:	**Modulo principale:**
A4, A5 (analogica)	2 x potenziometri lineari, 10 KOhm
Porte:	**Comp. Accessori:**
D11	Buzzer su RL1 (oppure uscita audio per amplificatore)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Minipops è un programma molto particolare e probabilmente al limite delle possibilità di Arduino Nano: infatti viene trasformato in una drum machine! Emula una batteria elettronica con suoni tipici degli anni 70/80degli anni, il Minipops della Korg, con un audio veramente ottimo.
Note:	Si possono collegare i due potenziometri alle porte A4 e A5 degli zoccoli AN2 e AN5, oppure usare due potenziometri del progetto per il synth a cinque potenziometri. **L'uscita audio è sul piedino digitale D11. Si può collegare un buzzer sullo zoccolo RL1 oppure collegare un minialtoparlante sui piedini "-" e "S".**
Link:	https://www.youtube.com/watch?v=8ccEyugm8PU

Nome del programma:	**Minipops_SH**
Porte:	**Modulo principale:**
A4, A5 (analogica)	2 x potenziometri lineari, 10 KOhm
Porte:	**Comp. Accessori:**
D11	Buzzer su RL1 (oppure uscita audio per amplificatore)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Minipops è un programma molto particolare e probabilmente al limite delle possibilità di Arduino Nano: infatti viene trasformato in una drum machine! Emula una batteria elettronica degli anni '80, il Minipops della Korg, con un audio veramente ottimo. Questa seconda versione usa un diverso set di strumenti a percussione, più "heavy metal".
Note:	Idem che per il Minipops
Link:	https://www.youtube.com/watch?v=8ccEyugm8PU

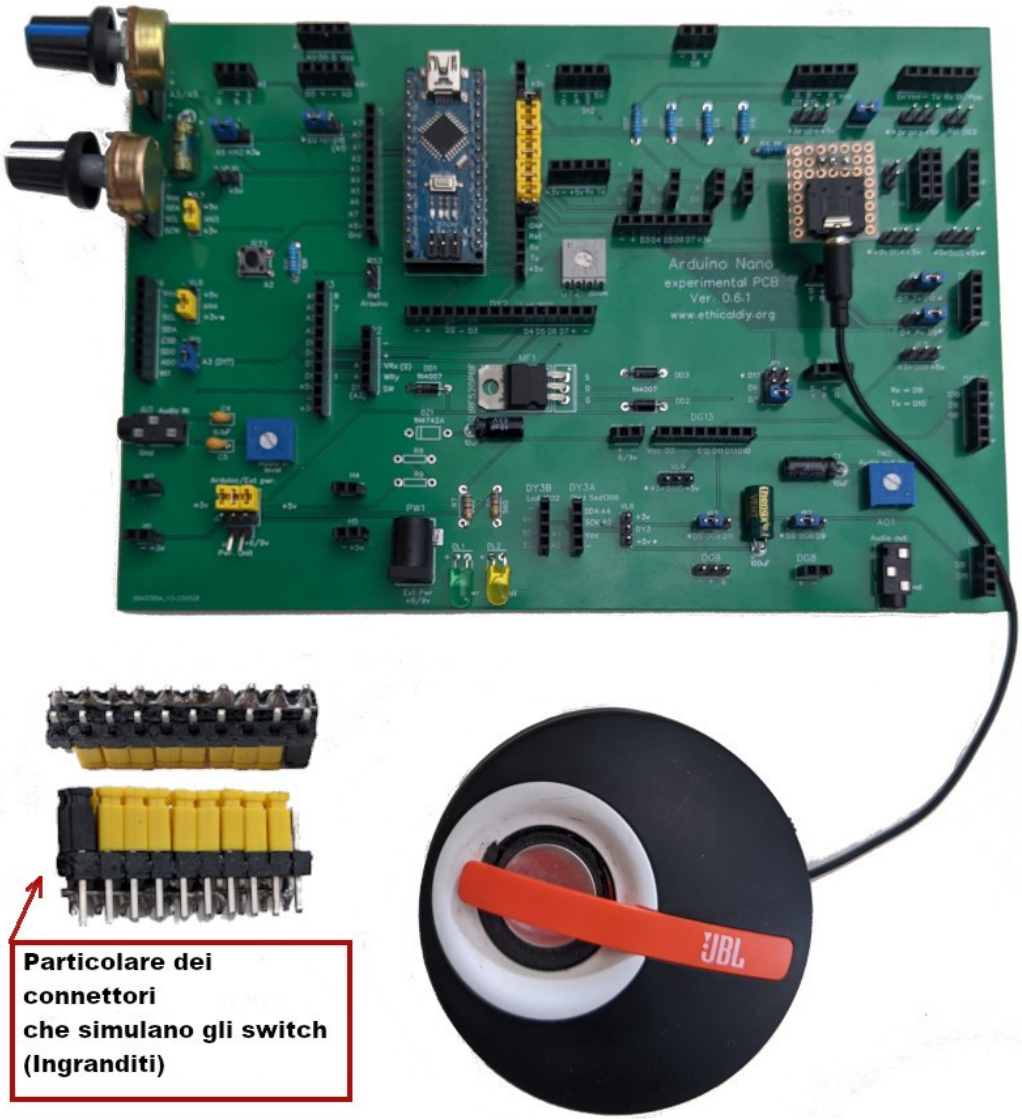
L'uscita audio è collegato tassativamente sul piedino 11.

Clicca qui per andare alla pagina del montaggio della scheda

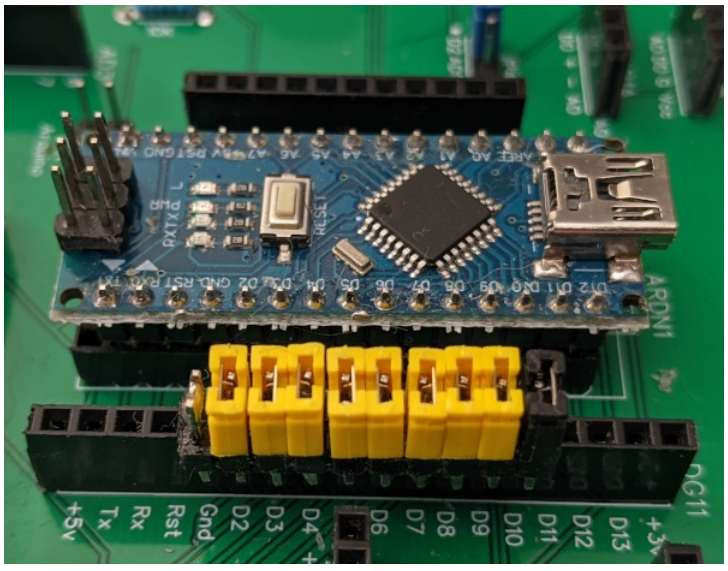
Clicca qui per vedere lo schema elettrico.

Nota: come si vede dallo schema elettrico, nove switch sono connessi alle porte digitali da D2 a D10 di Arduino, I primi otto (D2/./D9)servono per attivare/disattivare componenti del drum set; il nono (su D10) per attivare/disattivare il suono. Tutti hanno un piedino collegato alla massa, e questo semplifica lo schema. Quando il circuito è aperto, i vari componenti sono inseriti. Quindi se si avvia il progetto senza di essi, tutti i suoni sono disponibili.

Per non appesantire lo schema, ho trovato una soluzione semplice ed economica, anche se non molto elegante: ho utilizzato una doppia fila di 10 connettori maschi appaiati(sigla: HDR-M-2.54_2x10), di cui quelli relativi a un lato sono inseriti nello zoccolo DG11, da "GND" a "D10 (vedi immagine); i 10 della fila appaiata esterna sono tutti saldati insieme, nella parte inferiore. Inserendo un jumper collegato atra il piedino a "GND" e quello a lato, tutti gli switch sono potenzialmente collegati a massa. E' sufficiente inserire i vari jumper tra le coppie di piedini, per sentire via via il suono impoverirsi di particolari; se si connette il jumper relativo a "D10", il suono si interrompe totalmente. Se si desidera solo testare questo programma, si può fare tranquillamente a meno del contributo di questi pseudo-switch.



Particolare della basetta con i due potenziometri. Quello più in alto (jumper JP9 su A5) regola la velocità; quello inferiore (jumper VL10 collegato) ritmo adottato (rumba, cha-cha-cha, bossa nova, ecc.). L'uscita audio è sul piedino 11; si nota un piccolo adattatore per collegare un altoparlante amplificato.



Particolare dello zoccolo DG11, a cui è stato applicato l'emulatore degli switch per collegare/scollegare i componenti del drum set. In questa configurazione è tutto disattivato. Se interessa solo testare questo programma, si può fare benissimo a meno di questo artificio.

Un progetto realizzato:
controllo della temperatura di fluidi con due sensori

Ecco il progetto realizzato, composto da due basette; quella più grande contiene Arduino, il modulo alimentatore e i due relays che controllano le resistenze per il riscaldamento del liquido. Sulla destra si vede l'alimentatore. In alto si vede la piccola basetta con i controlli: Il piccolo display che indica la temperatura, i led indicatori, il buzzer per l'allarme.

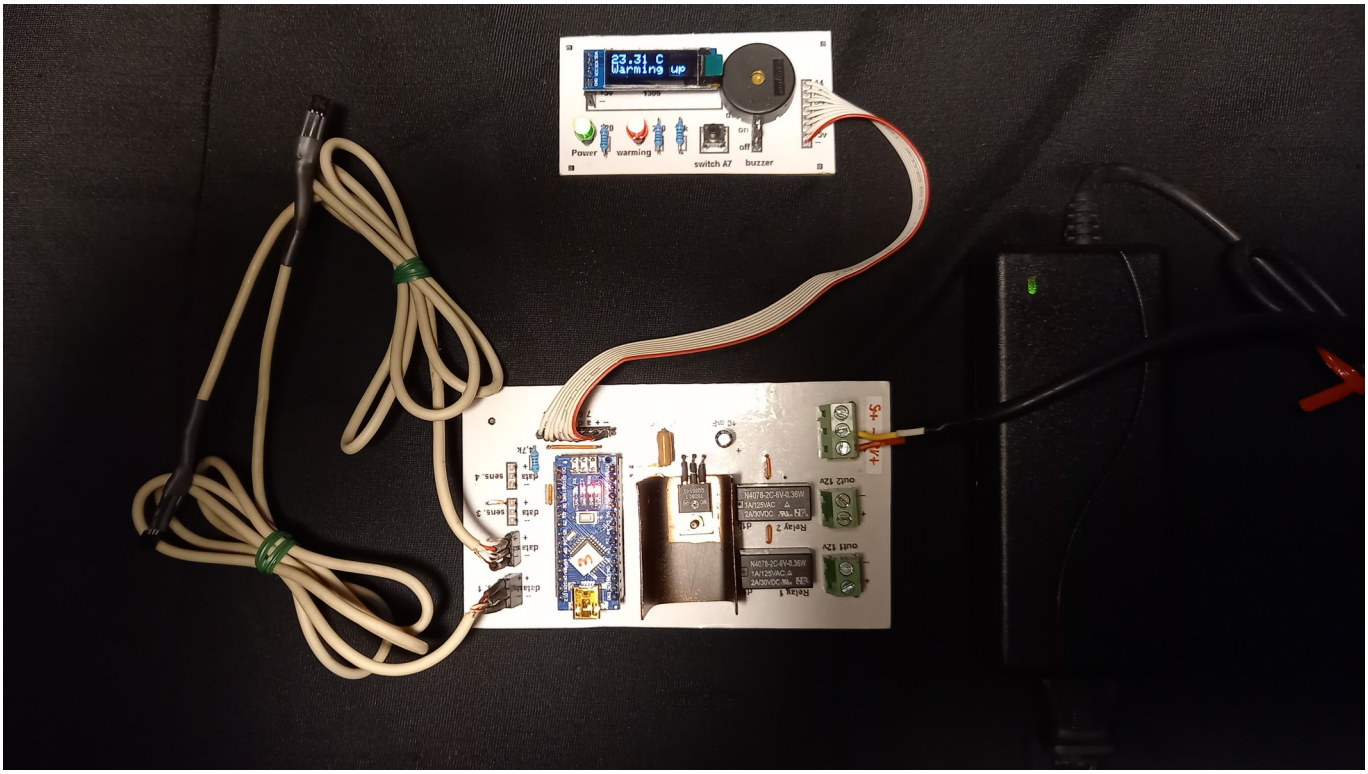


Immagine delle due basette stampate ad "hoc" per la creazione di un prototipo di stampante 3d per uso medicale

Programma con due sensori realizzato con la *bs*

Nome del programma:	**DS18B20_2**
Porte:	**Modulo principale:**
D3 (digitale)	DS18B20 – KY-001. Questo progetto ***usa 2 sensori***, che possono essere collegati insieme, sulla stessa porta. Gli zoccoli sono BZ1 (settato su D3) e DY1 (primi tre pin a sinistra)
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Porte:	**Comp. Accessori:**
D10 (digitale)	Led verde – alimentazione su LD2
D9 (digitale)	Led rosso – riscaldamento su LD3
D9 (digitale)	Relay 1 KY-019 su connettore RL1
D11 (digitale)	Relay 2 KY-019 su connettore RL2
D11 (digitale)	Led rosso – riscaldamento su LD1

D12 (digitale)	Buzzer KY-006 (oppure KY-012)
A2 (analogico)	Pulsante BT1 presente sulla basetta per verificare le 2 temperature
A4 (data); A5 (clock). Analogici.	Display OLED 128x32 con protocollo I2C su connettore DY3
Scopo del programma:	Questo programma è piuttosto articolato, ed è stato eseguito per un prototipo. Su di una siringa particolare, sono state montate due resistenze, comandate dai Relay RL1 ed RL2, che hanno lo scopo di portare il liquido a una temperatura desiderata. Un sensore 18B20 misura la temperatura del liquido, mentre l'altro quello delle resistenze. Se le resistenze sono troppo calde, con il rischio di danneggiare la siringa e il liquido, vengono disinserite, suona un allarme e il led verde lampeggia. Quando la temperatura del liquido è corretta, le resistenze si spengono. Il led verde indica il funzionamento del sistema, quello rosso si attiva quando le resistenze sono in funzione. Come si vede, i led non sono sulle stesse porte. Il display mostra la temperatura del liquido. Premendo BT1 si vede in alternativa la temperatura delle resistenze.
Note:	nessuna
Link:	nessuno

Una serratura a doppia chiave

Nella lista di programmi raccolti in questo manuale, si trovano due progetti separati, uno di una chiave che si apre a combinazione con una keypad, l'altro avvicinando una card a un sensore Rfid. Come esercizio ho provato a unire i due progetti, creando una chiave per fanatici della sicurezza. Per me, che non sono molto bravo nella programmazione, è stata una notevole sfida! Inoltre la keypad, insieme al lettore di schede Rfid utilizzano quasi tutte le porte disponibili di Arduino Nano.

I componenti sono i seguenti:

- una keypad, si può selezionare se usarne una da 4 x 3 tasti o una da 4 x 4;
- un lettore di schede RFID, RC522;
- un servomotore, MG90S (o simile);
- un relay KY-019
- un buzzer, KY-006 o KY-012 (amplificato);
- un led che indica il funzionamento del sistema;
- un led, collegato al relay, che indica l'apertura della serratura;
- un display LCD 1602 con adattatore per protocollo I2C

Il funzionamento del programma è abbastanza semplice: all'avvio il servo è in posizione di chiusura, il relay e il led corrispondenti non attivi. Sul display appare la richiesta di inserire una password; se la password è errata, il buzzer emette un suono a bassa frequenza. Se la password è corretta viene emesso un "bip!" e si viene invitati ad avvicinare la card al lettore Rfid; se la card corrisponde a ciò che si aspetta il programma, il servomotore si posiziona a riposo, aprendo l'eventuale serratura; il relay si attiva, si accende il led e il buzzer emette una nota di una ottava più alta che nel caso di errore.

Si può anche scegliere, facendo le opportune variazioni nel programma che segnaleremo, se mantenere aperta la "serratura" per cinque secondi (o per un tempo a piacere), oppure sempre aperta, fino alla pressione di un tasto sulla keypad che attiva manualmente la chiusura, riprendendo il ciclo di apertura/chiusura fino a quando si spegne il sistema.

Come è già stato accennato, in questo manuale i programmi sono un bonus, per cui non ho dedicato una cura particolare alla loro spiegazione. Ma per questo programma in particolare, darò qualche spiegazione in più.

All'avvio viene chiesta una password:

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
Servo ServoMotor;
char* password = "4271";  //Qui si stabilisce la password di accesso.
// La password può avere lunghezza a piacere, dipende da come si imposta lenpsw
// se si usa la keypad 4x3, si possono usare solo numeri da 0 a 9.
// se si usa la keypad 4x4, si possono usare sia numeri che le lettere A, B, C, D
int lenpsw = 4; //password's length
int keylen = 0;
int position = 0;
```

Le note possono essere già di aiuto. La password può essere lunga a piacere: dipende dal valore impostato nella variabile lenpsw. In questo caso la lunghezza della password è stata fissata a quattro caratteri.

Se si usa una tastiere 4 x 3, si può usare una combinazione di numeri compresi tra 0 e 9; se invece si usa quella 4 x 4, oltre che i numeri si possono usare anche le prime lettere dell'alfabeto: A,B, C, D.

Se la password è errata, suona una nota bassa e si è invitati a ripeterla nuovamente; se invece e corretta, si legge la richiesta di avvicinare la card al lettore. Per interrompere la procedura e tornare all'inserimento password, è sufficiente premete "*" o "#".

Vediamo ora come scegliere il tipo di tastiera, se "4 x3" o "4 x 4" tasti.

1) si attiva la tastiera 4 x 3 (12 tasti)

```
const byte ROWS = 4;  //Qui INIZIA la routine per la keypad 4x3
const byte COLS = 3;
char keys[ROWS][COLS] = {
  { '1', '2', '3' },
  { '4', '5', '6' },
  { '7', '8', '9' },
  { '*', '0', '#' }
};
byte rowPins[ROWS] = { 8, 7, 4, 6 };
byte colPins[COLS] = { A2, A3, A0 };  //Qui FINISCE la routine per la keypad 4x4

/*
const byte ROWS = 4;  //Qui INIZIA la routine per la keypad 4x3
const byte COLS = 4;

char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};
byte rowPins[ROWS] = { 8, 7, 4, 6 };
byte colPins[COLS] = { A2, A3, A0, A1 }; //Qui FINISCE la routine per la keypad 4x4
*/
```

Come si vede in figura, c'è una prima parte del programma attiva (evidenziata nel rettangolo rosso) e una in grigio, non attiva. Quindi, come si legge anche nelle note (che iniziano dopo i caratteri "//", in questo caso è attiva la keypad da 4 x 3 tasti, quindi la password sarà esclusivamente numerica. Da notare i simboli evidenziati dai quadrati verdi: "/*" indica che tutto quello che segue, fino a che non si incontri il simbolo "*/" è semplicemente un commento.

2) si attiva la tastiera 4 x 4 (16 tasti)

```
/*
const byte ROWS = 4;  //Qui INIZIA la routine per la keypad 4x3
const byte COLS = 3;
char keys[ROWS][COLS] = {
  { '1', '2', '3' },
  { '4', '5', '6' },
  { '7', '8', '9' },
  { '*', '0', '#' }
};
byte rowPins[ROWS] = { 8, 7, 4, 6 };
byte colPins[COLS] = { A2, A3, A0 };  //Qui FINISCE la routine per la keypad 4x4

*/
const byte ROWS = 4;  //Qui INIZIA la routine per la keypad 4x3
const byte COLS = 4;

char keys[ROWS][COLS] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};
byte rowPins[ROWS] = { 8, 7, 4, 6 };
byte colPins[COLS] = { A2, A3, A0, A1 }; //Qui FINISCE la routine per la keypad 4x4
```

questo secondo caso, viceversa, si è attivata la tastiera 4 x 4., perciò la password oltre che numeri, potrà contenere anche una combinazione delle lettere A, B, C, D.

Superato positivamente l'inserimento della password, verrà chiesto di avvicinare la propria card al lettore di Rfid. Ogni card ha un suo codice interno, che può essere verificato con il programma "DumpInfo" e trascrivere. Per esempio, abbiamo rilevato il seguente valore per la nostra card: "84 1B FB 9C". Quindi aprire il programma e inserire nella posizione appropriata il valore rilevato:

Al posto del codice mostrato, si potrà inserire nel programma quello della propria card.

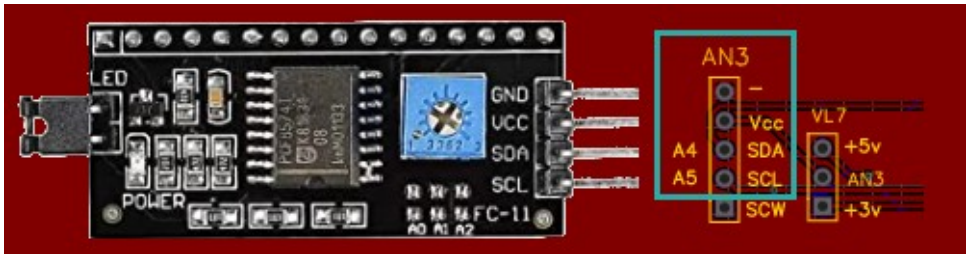
E' possibile effettuare ancora una ulteriore scelta: se decidere di lasciare la serratura aperta fino a quando non si preme sulla tastiera il tasto "*" o "#", oppure che si chiuda automaticamente dopo un certo periodo. Nel programma attualmente è stato inserito un ritardo di 5 secondi: "delay(5000);". Ecco come effettuare facilmente questa scelta modificando una variabile nel programma:

```
int Relay = 9;     //led + relay di controllo
int opentime = 1; //Se opentime = 0: apertura incondizionata; se è = 1: apertura a tempo
int rfok = 0;      //flag per accedere al controllo con Rfid se 1 = ok; se 0 = errato o in attesa


    switch(opentime){

    case 0:
    lcd.setCursor(0, 1);  //INIZIA la routine per l'apertura INCONDIZIONATA della serratura
    lcd.print("* per richiudere");
    Serial.println("Authorized access. * to close again");
    lcd.setCursor(0, 1);
    lcd.print("* per richiudere");
    if (key == '*' || key == '#') {  //if key = "*" or key = "#", reset LockedPosition on
    position = 0;
    LockedPosition(true);
    digitalWrite(Relay, LOW);
    ServoMotor.write(0);
    Serial.println("Stop access");
    position = 0;
    LockedPosition(true);
    rfok = 0;
    } //FINISCE la routine per l'apertura INCONDIZIONATA della serratura
    break;

    case 1:
    lcd.setCursor(0, 1);  // INIZIA la routine per l'apertura A TEMPO della serratura
    lcd.print("per 5 secondi");
    Serial.println("Authorized access for 5 seconds");
    delay(5000);
    digitalWrite(Relay, LOW);
    ServoMotor.write(0);
    Serial.println("Stop access");
    position = 0;
    LockedPosition(true);
    rfok = 0;  // FINISCE la routine per l'apertura A TEMPO della serratura
    break;
    }
}
```

Come si vede nel riquadro selezionato, la variabile "opentime" può avere due valori: se vale "0", si ottiene l'apertura incondizionata, e la serratura verrà richiusa solamente premendo i tasti "*" o "#"; se invece vale "1", la serratura si chiude dopo un tempo specificato.

La porzione del programma che inizia con “case 0:” è relativa all’apertura “incondizionata” della serratura, che verrà richiusa solamente premendo “*” o “#”, come si vede nel riquadro evidenziato in verde. Invece la routine che inizia con “case 1:” è relativa alla chiusura a tempo. Nei programmi di Arduino, il tempo è calcolato in millisecondi; infatti delay(5000); corrisponde a 5 secondi; naturalmente è possibile variare il tempo anche fino a qualche minuto.

Le informazioni sono riportate sullo schermo di un display LCD 1602, provvisto di un adattatore I2C, che richiede solo due porte analogiche (A4, SDA e A5 SCL) per connettersi ad Arduino. Se avessimo a disposizione il display senza adattatore, non lo potremmo usare, perché nativamente richiederebbe ben sei porte digitali, che per questo progetto non abbiamo a disposizione!



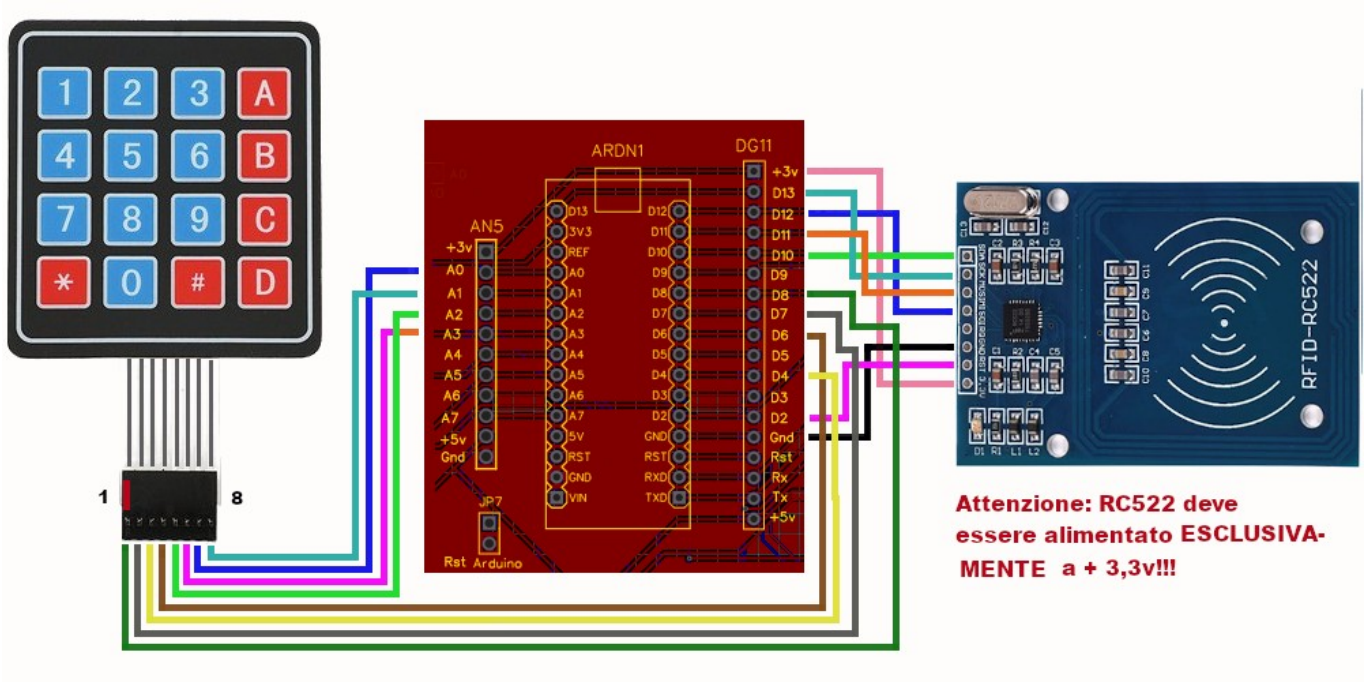
Il dispay LCD 1602 con adattatore I2C verrà collegato ad Arduino utilizzando le prime quattro porte dello zoccolo AN3.

Ora approfondiremo l’aspetto dei collegamenti dei moduli al microcontroller, aspetto particolarmente impegnativo in questo progetto, perché, come accennato in precedenza, si utilizzano quasi tutte le porte di Arduino Nano:

Zoccolo DG11 **Zoccolo AN5**

Porta	Applicazione	Porta	Applicazione
13	RFID Sck	A0	Keypad pos. 7
12	RFID Miso	A1	Keypad pos. 8 (solo per tastiere 4x4)
11	RFID Mosi	A2	Keypad pos. 5
10	RFID SDA + led LD1	A3	Keypad pos. 6
09	Led LD3 + Relay RL2	A4	Display LCD 1602 Sda
08	Keypad pos. 1	A5	Display LCD 1602 Scl
07	Keypad pos. 2	A6	non usato
06	Keypad pos. 4	A7	non usato
05	Servomotore MG90S		
04	Keypad pos. 3		
03	Buzzer su BZ1		
02	RFID Rst		

Ed ecco il diagramma dei collegamenti:



Nota: se si usa la keypad 4 x 3, non si collegherà il cavo 8 alla porta A1; tutti gli altri collegamenti restano uguali.

Nome del programma: Key_lock

Porte:	**Modulo principale:**
A0/./A3 (analogico)	**Keypad 4x4 (in alternativa a Keypad 4x3)**
D4, D6, D7, D8 (digitale)	
A0, A2, A3 (analogico)	**Keypad 4x3 (in alternativa a Keypad 4x4)**
D4, D6, D7, D8 (digitale)	
D2, D10, D11, D12, D13	**RC522 (RFID reader)**
Porte:	**Comp. Accessori:**
D3 (digitale)	Buzzer KY-006 (KY-012) su BZ1
D5 (digitale)	Servo MG90S su DG9
D9 (digitale)	Led verde su LD3
D9 (digitale)	Relak KY-019 su RL2
D10 (digitale)	Led blu su LD1
Monitor Seriale: sì (9600 baud)	**Plotter seriale: no**
Scopo del programma:	Questa è una doppia serratura: richiede prima una password da inserire sulla tastiera; poi bisogba accostare una card al lettore di Rfid card. Se tutto è corretto, si accende il led verse, si ode una nota acuta, il servo sposta il suo asse di 90° (aprendo una possibile serratura) e il relay scatta
Librerie necessarie:	LiquidCrystal_I2C.h, SPI.h, MRC522.h, Servo,h, Keypad.h
Note:	Nessuna.
Link:	https://www.meccanismocomplesso.org/lcd1602-utilizzare-un-display-a-cristalli-liquidi-lcd-con-arduino-tramite-i2c/

Telecamera OV7670

Esiste una piccola telecamera, la OV7670 che può essere collegata ad Arduino con un costo veramente esiguo. Naturalmente non ci si può aspettare miracoli, però funziona!
Su Internet si trovano decine di tutorial per collegare la telecamera su Arduino. Dopo averne letto e testate alcune, ho deciso di seguire passo-passo questo, che mi è sembrato il più ben eseguito, non richiede di installare alcun motore Java sul proprio computer. Eseguiti i collegamenti, instattata la libreria e il tool "ArduImageCaptur", lo sketch ha funzionato al primo colpo.

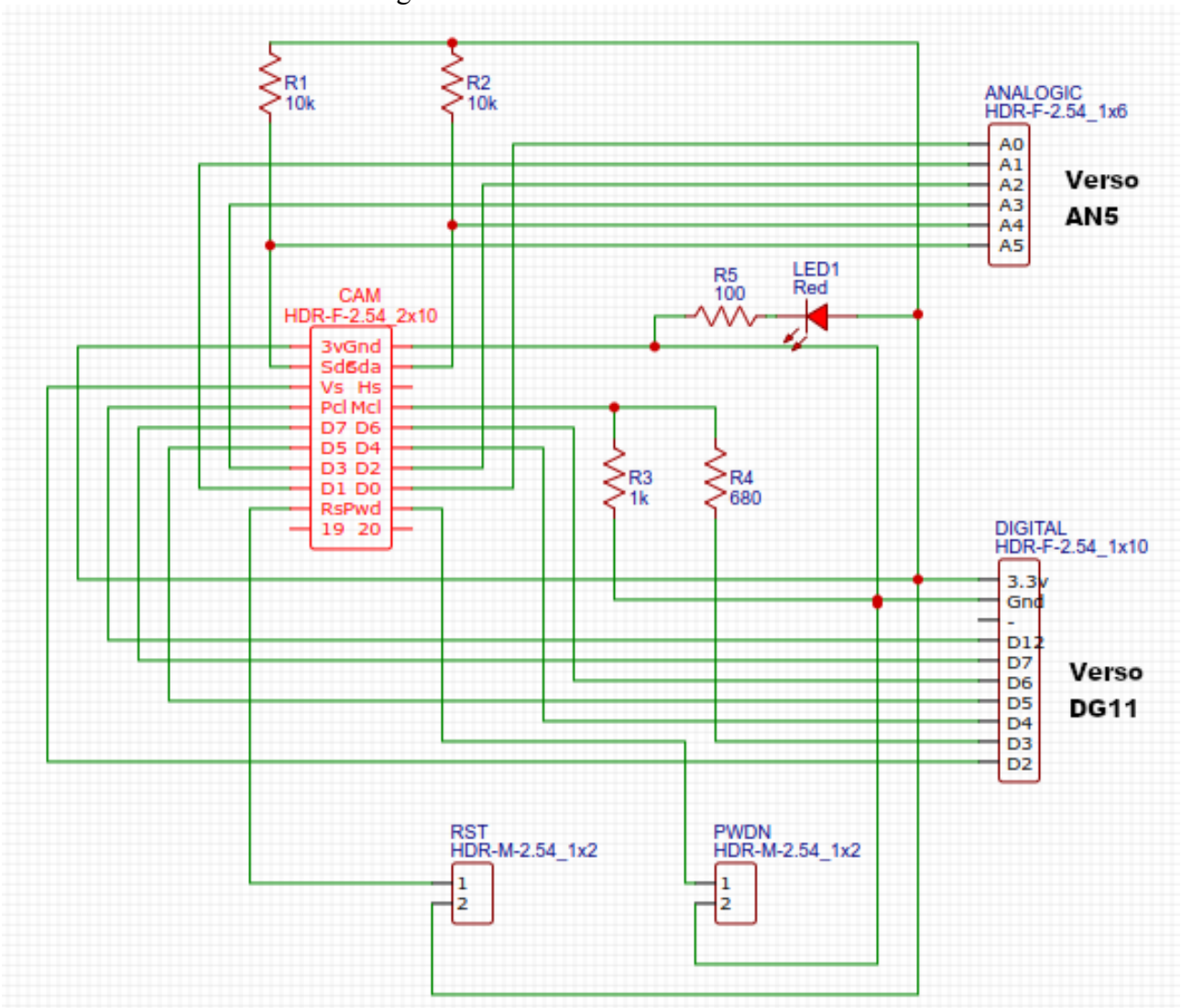
Con una risoluzione di 640x480 a colori, si ottiene un'immagine ogni 3 secondi circa, quindi non si possono ottenere filmati, ma una sequenza di foto che possono anche essere salvate sul proprio computer.

Non credo che possa essere utilizzata per una vera videosorveglianza, però se ne potrà trovare qualche interessante applicazione…

Per prima cosa, è necessario collegare una serie di cavi tra la telecamera e la nostra bs, e collegare anche alcune resistenze, perché la telecamera funziona nativamente a 3,3 V, e sebbene Arduino fornisca un'alimentazione a questo voltaggio, poi le porte di comunicazione lavorano a 5 volt, quindi è necessario prendere alcune precauzioni.

Lo schema elettrico

Ecco lo schema elettrico dei collegamenti:



Può sembrare abbastanza complesso, ma è stato inserito anche lo schema di una basetta per costruirlo facilmente.
Naturalmente questo schema può essere eseguito con un pò di pazienza su di una bredboard, avendo cura di non effettuare errori o cortocircuiti.
Se invece questo sistema può apparire troppo complicato, **troverete nelle appendici** anche le informazioni necessarie per ottenere la basetta pronta per montare i pochi componenti necessari.
Ecco le immagini del progetto eseguito con Easy Eda:

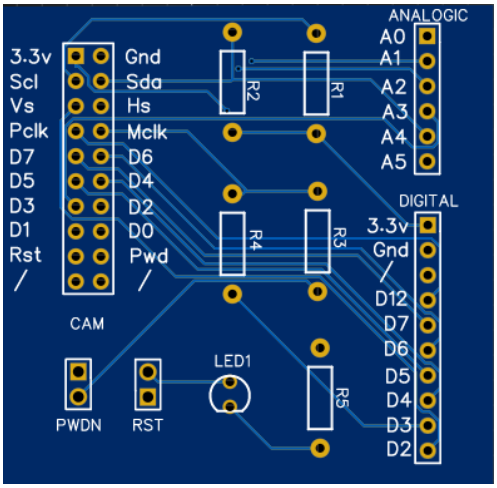


Immagine della basetta in formato 1:1

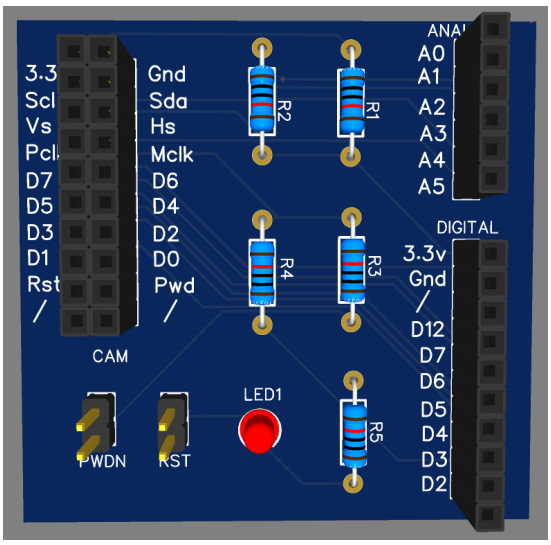
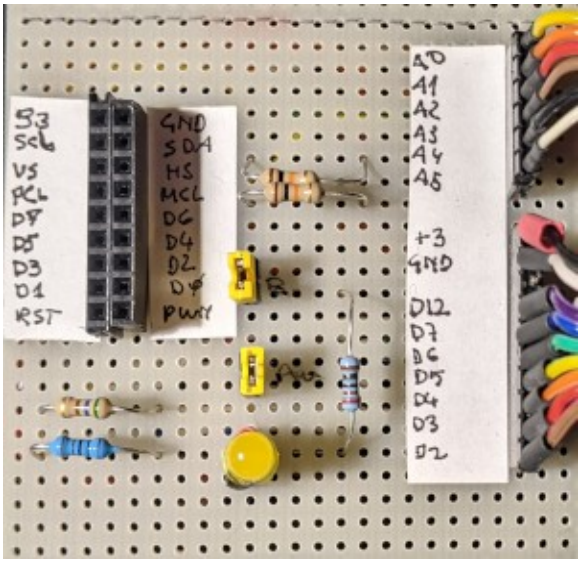


Immagine della stessa basetta, sempre in formato 1:1, montata con i vari componenti.

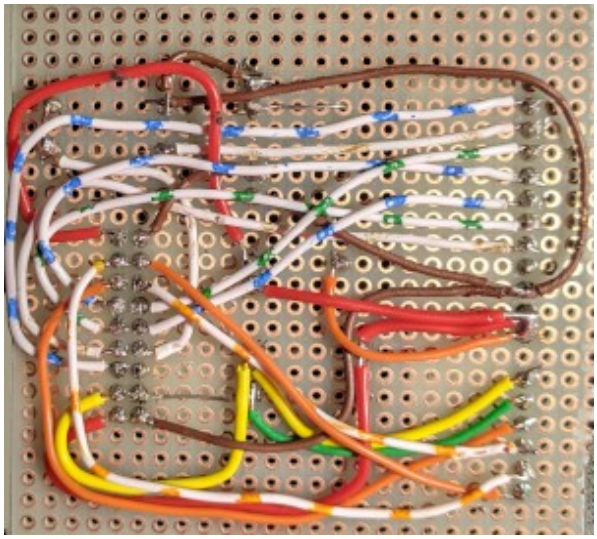
Poiché la telecamera non presenta alcun led che ne indichi l'attività ne ho inserito uno alimentato tramite la resistenza R5, che si accende all'attivazione della telecamera.

Sulla basetta ci sono due jumper, "PwDn" e "Rst". Come intuibile "PwDn" significa "Power Down", ossia toglie l'alimentazione alla telecamera, mentre "Rst" significa "Reset". ***Entrambi debbono essere ponticellati***, altrimenti la telecamera non fornirà alcuna immagine!

Personalmente per i test, data la mia avversione per l'uso delle breadboard, ho assemblato rapidamente un circuito con una basetta millefori, che funziona perfettamente e può essere riutilizzata tantissime volte. Allego un paio di foto. Naturalmente per eseguirla è necessario un po' di esperienza di saldatura e di manualità…



Il frontale del prototipo...



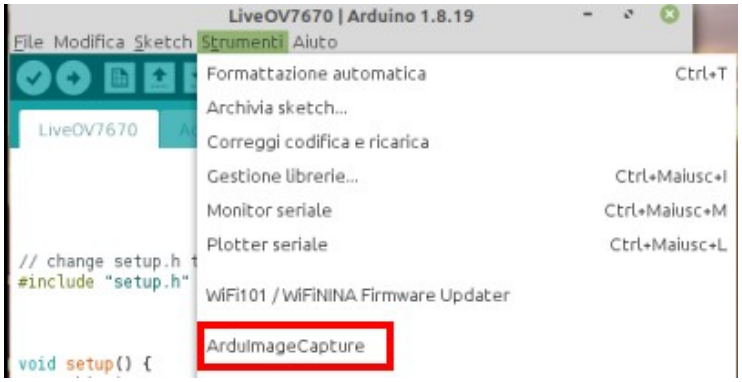
… e il retro, un po' trafficato!

Il programma, le librerie ed altro.

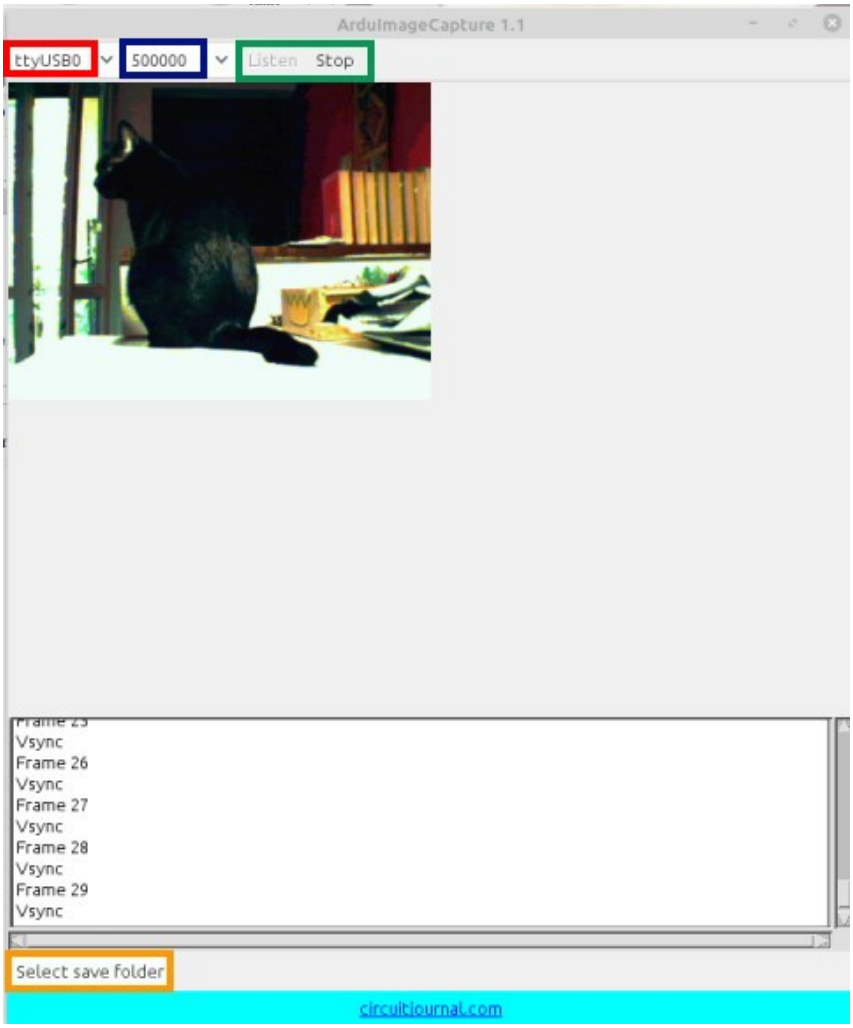
Prima passo: è necessario che la libreria “LiveOV7670Library” sia presente nella cartella delle librerie. Se necessario, inserirla.

Nota: se nella cartella delle librerie di Arduino hai copiato ed espanso il file compresso “libraries.zip” hai già installato tutte le librerie necessarie per testare tutti i programmi proposti in questo manuale.

Secondo passo: nella cartella in cui è stata inserita la IDE di Arduino, creare se necessario una cartella “Tools” e copiare al suo interno il file compresso “ArduImageCapture.1.1.zip” ed estrarlo.
Se tutto è andato a buon fine, aprendo la Ide di Arduino, dovresti trovare in “strumenti”, anche la voce “ArduImageCapture”. Tutto sta procedendo correttamente!



Terzo passo: caricare il programma della fotocamera, compilarlo e lanciarlo.
Cliccare su strumenti e attivare ArduImageCapture.
Si aprirà il programma che cattura le immagini dalla fotocamera. Selezionare la velocità di collegamento (nel mio caso, “500000” e premere su “Listen”. Se i collegamenti alla fotocamera sono interrotti o non corretti, apparirà dopo pochi secondi un quadrato rosso. Se invece funziona tutto correttamente, prima si vedrà un quadrato verde e poi finalmente un’immagine, con un refresh di alcuni secondi.
Ecco il mio gatto!
Alcune indicazioni: nel riquadro rosso appare la porta com usata. In quello verde la velocità di comunicazione. Non sono riuscito a superare i 500000. Si può premere su “Listen/stop” alternativamente per attivare/disattivare la telecamera. Infine con “select save folder”, si potrà selezionare la cartella in cui salvare le immagini.



Naturalmente prima di lanciare il programma, è necessario collegare i vari cavetti tra la telecamera e la bs. Come si può vedere sia dal mio circuito sperimentale che dalla basetta eseguita con Easy Eda, ho creato due diversi connettori, uno per i collegamenti analogici, da A0 ad A5, che andranno sul connettore analogico AN5; un secondo per connettere l'alimentazione (attenzione: 3,3 v!), la massa e le varie porte digitali sullo zoccolo DG11. I collegamenti alle porte sono indicati a lato del connettore specificato. Se si utilizzasse una breadboard, fare molta attenzione ai collegamenti.

Un'immagine catturata con il programma

La cam OV7670

Nota 1: attenzione a salvare le immagini, specialmente se si lascia la telecamera attiva per molto tempo: ogni immagine è di circa 44 K e singolarmente sono veramente piccole, ma in un ora ne vengono salvate circa 1200, e quindi la dimensione della cartella rapidamente diventa rilevante!

Nota 2: inserendo lo strumento nella cartella "tools" di Arduino, tutto funziona perfettamente con la IDE 1.8.9 (il programma che serve per caricare e lanciare gli sketch), mentre non sono riuscito a trovare la voce "ArduImageCapture" in "strumenti" nella versione 2.x della IDE con sistema operativo Linux. Probabilmente tutto funziona correttamente con Windows o Apple. Comunque non è il caso di preoccuparsi.
Aprire la cartella ../Arduino/tools/ArduImageCapture.
All'interno c'è il programma per lanciare manualmente l'applicazione. Nel caso di Linux, il comando è Linux_ArduImageCapture.sh. Renderlo eseguibile (tasto dx, proprietà/permessi. Spuntare "Consentire l'esecuzione del file come programma") e lanciarlo. Si aprirà la finestra di ArduImageCapture. Poi eseguire come descritto precedentemente.

Nome del programma:	**LiveOV7670**
Porte:	**Modulo principale:**
A0/./A5 (analogico)	**Telecamera OV7670**
D2/./D7, D12 (digitale)	
Monitor Seriale: no	**Plotter seriale: no**
Scopo del programma:	Permette di vedere e salvare le immagini della fotocamera.
Librerie richieste:	LiveOV7670-master.zip
Note:	Richiede che nella cartella "tools" di Arduino, sia presente la cartella "ArduImageCapture.1.1.zip", che deve essere decompressa.
Link:	https://circuitjournal.com/arduino-OV7670-to-pc

Clicca qui per andare al progetto della basetta.

Sezione IV

Le librerie di Arduino

Le librerie di Arduino

Per utilizzare alcuni moduli , è necessario caricare delle librerie specifiche.

Una libreria per Arduino è un codice contenente delle funzioni per consentire al prodotto di connettersi/interfacciarsi con Arduino senza essere costretti a implementare ogni volta decine di righe di codice. Il concetto è molto simile a quello che nell'informatica tradizionale è il driver per una stampante.

Ogni componente aggiuntivo come: motori, display lcd, servomotori,ecc., hanno una propria libreria che deve essere implementata nel codice (sketch) per gestire la connessione tra questo componente e la scheda Arduino Uno o ESP8266 o qualsiasi altra scheda.

Qual'è il vantaggio di utilizzare una libreria?

- risparmio di memoria sulla scheda: carichiamo e richiamiamo sulla nostra scheda solo le librerie che ci servono risparmiando memoria sul dispositivo. Le schede sono dispositivi molto meno potenti dei computer ed è fondamentale allocare la memoria nel miglior modo possibile
- modularità: internet of think evolve a velocità impressionanti non sarebbe possibile avere tutte le librerie già installate nel microcontrollore; servirebbe oltre ad una memoria enorme anche un continuo processo di aggiornamento
- tempo: usare una libreria significa poter collegare un oggetto in pochi istanti poiché tutto la tematica di connessione è già stata sviluppata da programmatori professionisti che hanno testato il tutto prima di mettere il prodotto sul mercato.
- sicurezza: la libreria è sviluppata solitamente dal produttore del componente che conosce in modo completo l'oggetto che si dovrà connettere. Lo sviluppo è affidato a programmatori interni all'azienda sviluppatrice.

Come fare a installare una libreria per il programma (sketch) che voglio caricare su Arduino?

Nel caso non si sia ancora capaci a caricare una libreria, suggerisco di fare una ricerca su Internet, dove sitrovano ottimi manuali/video per effettuare questa operazione.
Questo è il link "ufficiale" di Arduino. E' in inglese, ma con il traduttore è comunque del tutto comprensibile: https://docs.arduino.cc/software/ide-v1/tutorials/installing-libraries

Viene fornita qualche libreria con questo manuale?

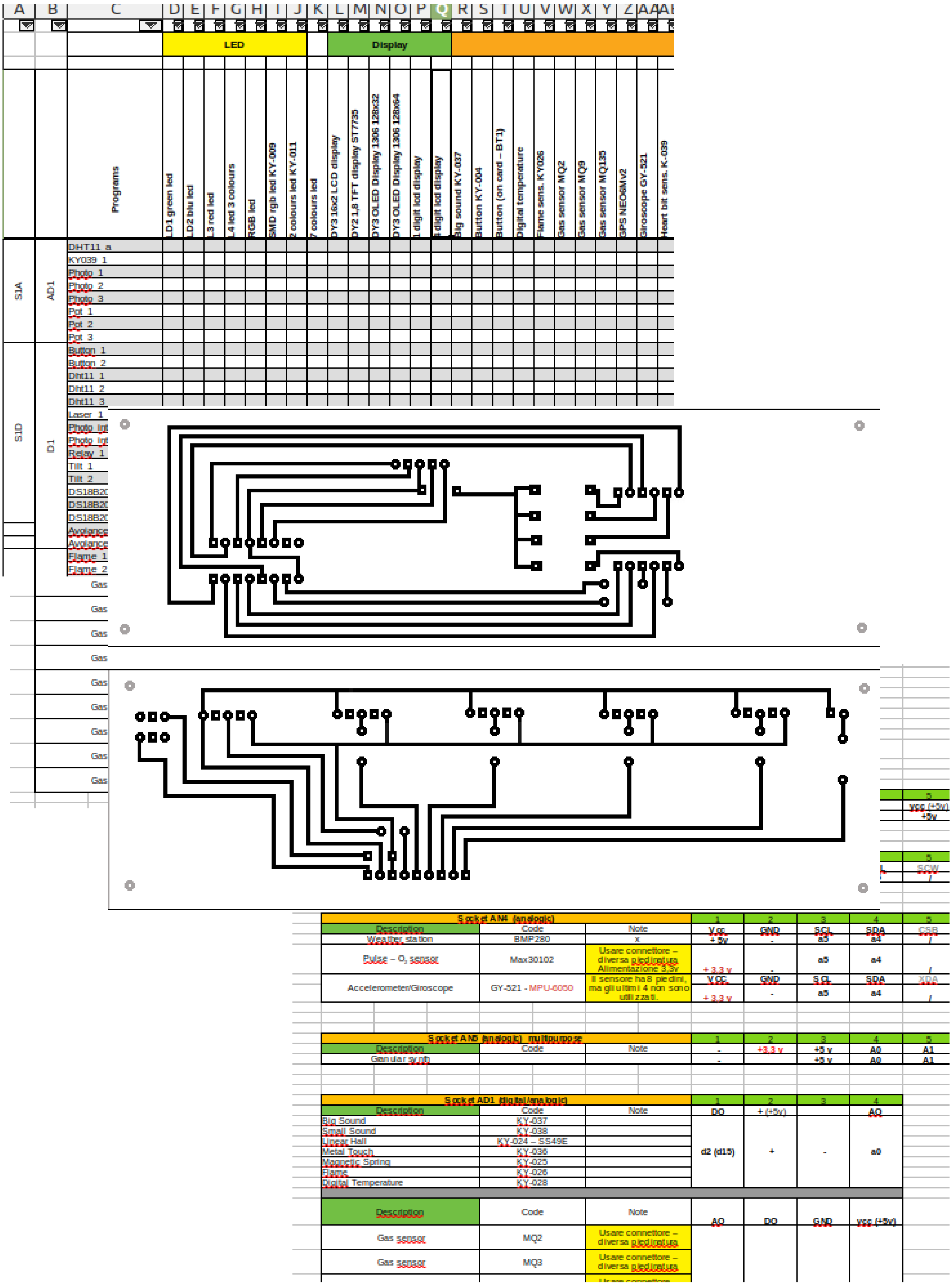
Nella pagina del download ci saranno anche due cartelle in formato "zip", una chiamata "libraries" e una seconda, "libraries_33". Nella prima delle due sono state salvate ***tutte*** le librerie utilizzate da Arduino Nano "base". La seconda, "libraries_33", contiene le librerie necessarie per poter utilizzare anche Arduino Nano 33 (IoT e BLE).

E' sufficiente salvare i due file zip "libraries.zip" e libraies_33.zip" nella propria cartella delle librerie e scompattarle al suo interno. Avremo così tut"te le librerie necessarie per testare i programmi presentati da questo manuale.

In Linux si trova abitualmente in: /*home/*Arduino/libraries
In Windows 10 si trova abitualmente in: "C:\Users\utente\documents\arduino"

Sezione V

Le appendici



Le tabelle comparative

Sono state approntate un certo numero di tabelle comparative, utili per conoscere e utilizzare al meglio la bs. Purtroppo, dato le dimensioni e il numero delle colonne delle tabelle stesse, non sempre è possibile "comprimerle" in un foglio A3 in verticale. Perciò in questo paragrafo verranno indicate le funzioni principali, e si attiverà un link per poterle visualizzare a schermo intero.

Il file che le contiene tutte si chiama, senza grande fantasia, "tabelle" ed è raggiungibile a questo link.
Esso è in formato ".ods", ovvero un foglio di calcolo in formato libero. Le ultime versioni di Excel dovrebbero aprirlo senza problemi; tuttavia se ciò non avvenisse, oppure non aveste la suite per ufficio di Microsoft, o se voleste passare (almeno per queste applicazioni) al software libero, scaricate e installate Libre Office. Non ne resterete delusi!

Ecco la descrizione delle singole tabelle:

- **Applicazioni**. In questa tabella troverai l'elenco di tutti i programmi proposti, divisi per tipo di zoccolo e con l'indicazione di tutti i moduli utilizzati per quel particolare sketch. In questo modo, si potranno saltare tutti i programmi che contengono i moduli che non si posseggono attualmente.

- **Comparazione**. Questa tabella è già comparsa come parte integrante delle appendici, ma qui si trova in forma di foglio di calcolo, quindi è possibile effettuare delle ricerche mirate. Essa è relativa alla comparazione tra i vari kit do sensori che ho acquistato.

- **Cross-list**. Anche questa tabella appare nelle appendici. E' una tabella comparativa tra i codici "KY", "HW" ed eventualmente altri codici.

- **Immagini**. Idem anche per questa tabella. Oltre che alle cross list dei codici, si trovano anche le immagini dei vari moduli.

- **Socket** (zoccoli). In questa tabella si trova l'elenco di tutti i moduli, il socket a cui si collegano, le porte utilizzate, eventuali note. Queste informazioni si trovano divise nelle informazioni contenute nella sezione_II, relativa a ogni zoccolo. Qui si trovano raggruppate insieme.

- **Porte**: indica le porte utilizzate per ogni zoccolo. In questo modo si è consapevoli di eventuali conflitti tra moduli. Essa è molto utile nella stesura di nuovi programmi, per non perdere tempo a cercare strani malfunzionamenti nel codice stesso.

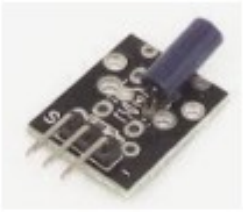
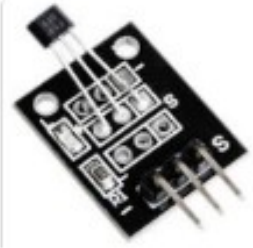
Riferimenti incrociati tra i codici KY – HW dei vari moduli (Cross reference)

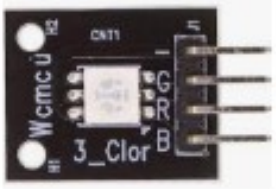
Quando ho iniziato questo progetto, ho trovato qualche difficoltà ad abbinare i vari codici dei moduli con le descrizioni corrispondenti, e non ho trovato in Internet (anche se probabilmente ci sarà) una tabella completa che incroci i vari codici con le corrispondenti descrizioni, così nel tempo ne ho strutturato una personalmente, spero sia sufficientemente accurato.
In un paragrafo successivo di questa sezione, troveremo anche le immagini relative ad ogni modulo.

KY	HW	other	description
KY-001	HW-506	18B20	Temperature sensor module
KY-002	HW-513		vibration (shock) switch module
KY-003	HW-495		Hall magnetic sensor
KY-004	HW-483		Key switch module (button)
KY-005	HW-489		Infrared trasmitter module
KY-006	HW-508		Passive buzzer module
KY-007		HC-SR0501	Pir module
KY-008	HW-493		Laser led
KY-009	HW-478		SMD RGB 3 colours led
KY-010	HW-487		Opto light switch (photo interrupt)
KY-011	HW-480		Two colours led
KY-012	HW-512		Active buzzer module
KY-013	HW-498		Analogue temperature module
KY-014		GY65 – BMP085	Atmospheric pressure sensor
KY-015	HW-507	DHT11	Temperature & humidity sensor
KY-016	HW-479		RGB tree colours led
KY-017	HW-505		Mercury tilt switch
KY-018	HW-486		LDR – photoresistor – analogue output
KY-019	HW-482		Relay
KY-020	HW-501		Tilt sensor
KY-021	HW-497		Reed switch sensor
KY-022	HW-490		Infrared receiver
KY-023	HW-504		Joystick
KY-024	HW-509		Hall effect sensor module plus
KY-025	HW-484		Reed relay – magnetic spring digital output module plus
KY-026	HW-491		Flame sensor module plus
KY-027	HW-499		Magic light cup module
KY-028	HW-503		Temperature sensor module plus
KY-029	HW-477		Two colours led module
KY-030		MQ-2	Gas sensor module

KY	HW	other	description
KY-031	HW-500		Hit (Tap/shock/vibration) sensor
KY-032	HW-201		Obstacle avoiance detector module plus
KY-033	HW-511		Line (tracking) follower module plus
KY-034	HW-481		seven colours automatic flash led module
KY-035	HW-492		Hall effect sensor
KY-036	HW-494		Metal touch sensor module plus
KY-037	HW-485		Amplified microphone sound sensor (big sound) module plus
KY-038	HW-496		Microphone sound sensor (small sound) module plus
KY-039	HW-502		Heart pulse rate detector
KY-040	HW-040		Rotary encoder
KY-050		HC-SR04	Ultrasonic sensor

Tutti i moduli usati corredati di immagini

Pos.	KY	HW	other	description	Image
1	KY-001	HW-506	18B20	Temperature sensor module	
2	KY-002	HW-513		vibration (shock) switch module	
3	KY-003	HW-495		Hall magnetic sensor	
4	KY-004	HW-483		Key switch module (button)	
5	KY-005	HW-489		Infrared trasmitter module	
6	KY-006	HW-508		Passive buzzer module	
7	KY-007		HC-SR0501	Pir module	

8	KY-008	HW-493		Laser led	
9	KY-009	HW-478		SMD RGB 3 colours led	
10	KY-010	HW-487		Opto light switch (photo interrupt)	
11	KY-011	HW-480		Two colours led	
12	KY-012	HW-512		Active buzzer module	
13	KY-013	HW-498		Analogue temperature module	
14	KY-015	HW-507	DHT11	Temperature & humidity sensor	
15	KY-016	HW-479		RGB tree colours led	

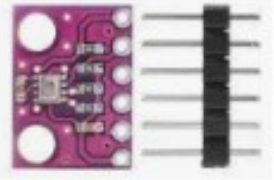
16	KY-017	HW-505		Mercury tilt switch	
17	KY-018	HW-486		LDR – photoresistor – analogue output	
18	KY-019	HW-482		Relay	
19	KY-020	HW-501		Tilt sensor	
20	KY-021	HW-497		Reed switch sensor	
21	KY-022	HW-490		Infrared receiver	
22	KY-023	HW-504		Joystick	

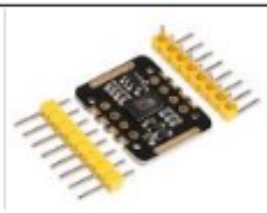
23	KY-024	HW-509		Hall effect sensor module plus	
24	KY-025	HW-484		Reed relay – magnetic spring digital output module plus	
25	KY-026	HW-491		Flame sensor module plus	
26	KY-027	HW-499		Magic light cup module (2 pz)	
27	KY-028	HW-503		Temperature sensor module plus	
28	KY-029	HW-477		Two colours led module	
29	KY-030		MQ-2	Gas sensor module	

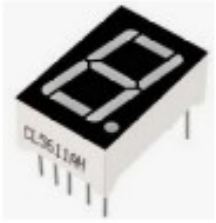
30	KY-031	HW-500		Hit (Tap/shock/vibration) sensor	
31	KY-032	HW-201		Obstacle avoiance detector module plus	
32	KY-033	HW-511		Line (tracking) follower module plus	
33	KY-034	HW-481		seven colours automatic flash Led module	
34	KY-035	HW-492		Hall effect sensor	
35	KY-036	HW-494		Metal touch sensor module plus	
36	KY-037	HW-485		Amplified microphone sound sensor (big sound) module plus	
37	KY-038	HW-496		Microphone sound sensor (small sound) module plus	

38	KY-039	HW-502		Heart pulse rate detector	
39	KY-040	HW-040		Rotary encoder	
40	KY-050		HC-SR04	Ultrasonic sensor	
41			GY-521	accelerator/giroscope module	
42			DS-1307	RTC module	
43			LCD1602	LCD display	
44			Sen18	Water level sensor	
45				4 x 4 Membrane keyboard	

46				Power supply module	
47			DS1302	RTC clock module	
48				SD card reader	
49				Buck converter (alimentation)	
50				Soil moisture sensor	
51			NEO-6MV2	GPS module	
52			MQ-3	Gas sensor module	
53			MQ-4	Gas sensor module	

54			MQ-5	Gas sensor module	
55			MQ-6	Gas sensor module	
56			MQ-7	Gas sensor module	
57			MQ-8	Gas sensor module	
58			MQ-9	Gas sensor module	
59			MQ-135	Gas sensor module	
60			BMP280	temperature/pressure/ altitude	

61			MAX 30102	Heat beat/O_2/human body temperature	
62			HC-05	Bluetooth module	
63			HC-06	Bluetooth module	
64			HM-19	Bluetooth module	
65			ESP8266	Wi-fi module	
66			ESP-01	Adapter for ESP8266	
67			ST7735	Dispaly TFT 1,7”	

68			SSD 1306 128x32	Oled white display 128 x 32 dot I2C	
69			SSD1306 128x64	Oled blue/yellow display 128 x 64 dot I2C	
70			MG90S	Servomotr	
71			Motor (generic)	Electric motor 5/./9 V	
72			5641AS	Four digit display (Red)	
73			5611AH	One digit display (Red)	
74			RC522	RFID card reader	

75			HX711	Load cell (weight)	
76				Potentiometer	
77				Remote command unit	
78				Stepper motor	
79			ULN2003	Stepper motor driver	
80				433 MHz transmitter (TX) module	
81				434 MHz receiver (RX) module	
82				Cam 640 x 480 OV7670	

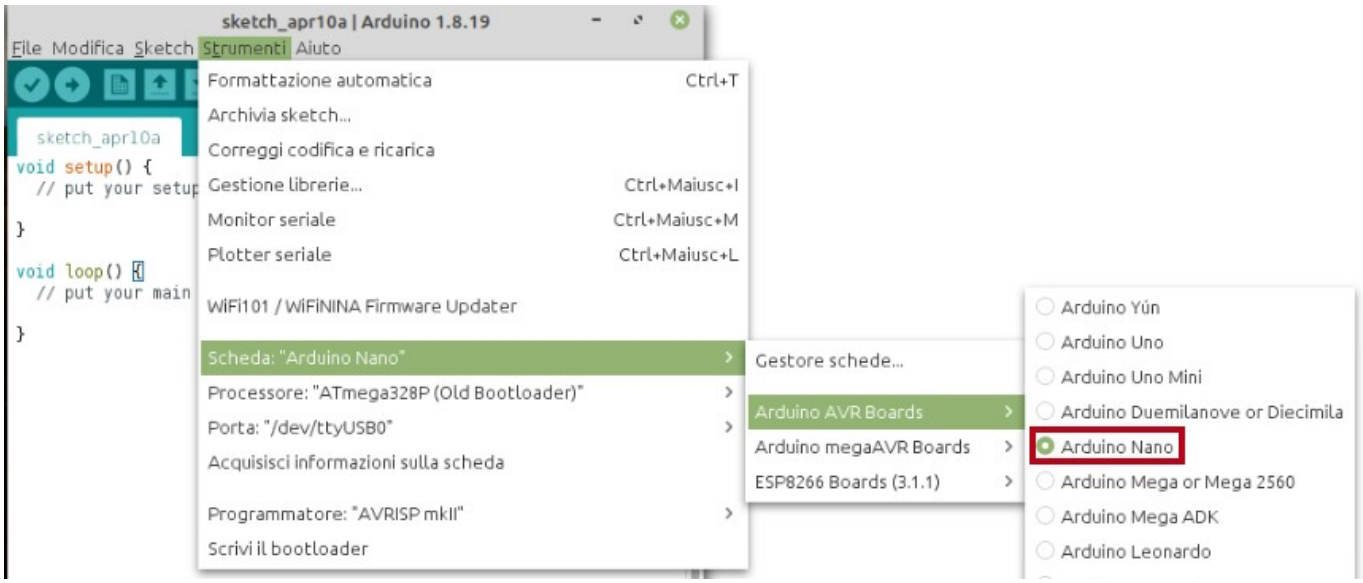
83				GPS Neo 7M	
84				Laser receiver module	
85				Multiplexer CD74HC4067	
86				Light sensor GY302 -BH1750	

N.B.: le foto dei moduli non sono in scala.

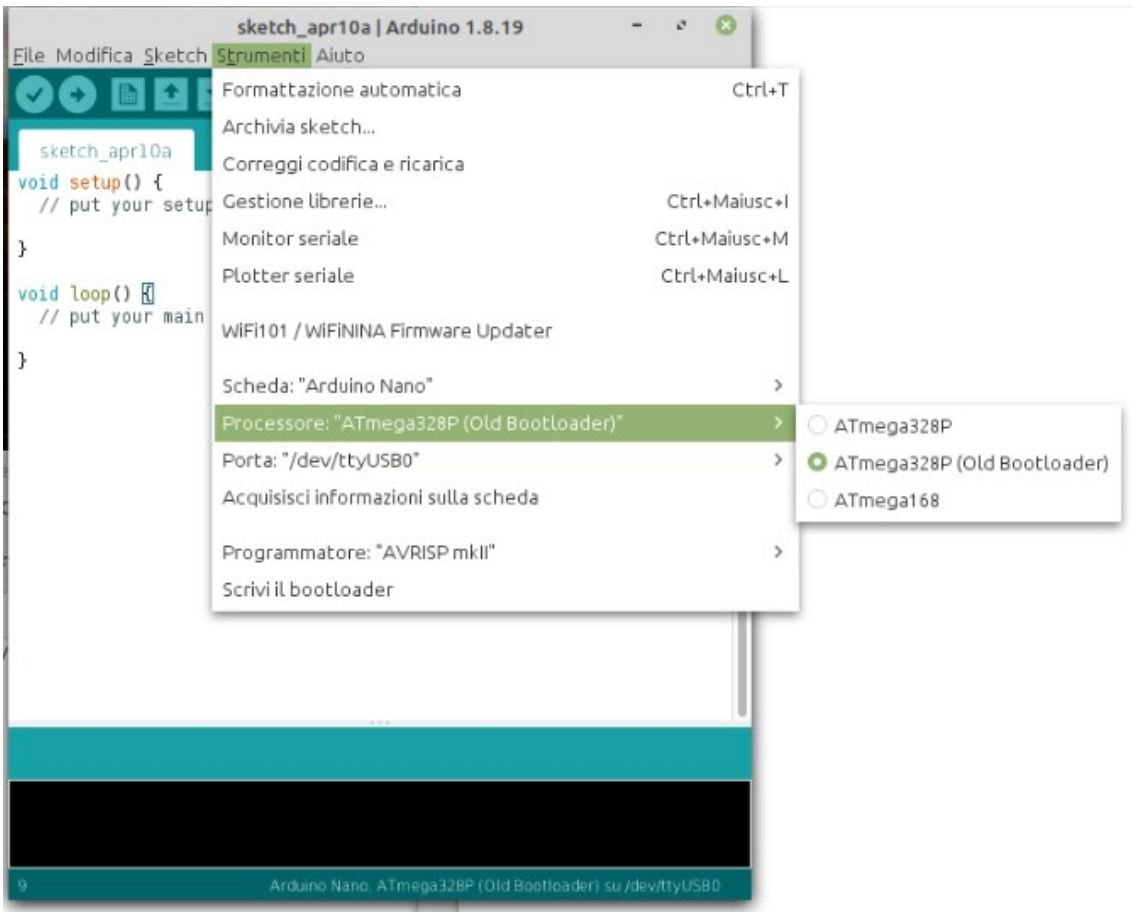
Interfaccia di Arduino Nano con il proprio personal computer

Come per tutte le cose, l'interfaccia di Arduino Nano con il personal computer è facile, quando si sa come fare… Ottenere qualche informazione quindi rende tutto più facile.

Per prima cosa, è necessario aprire la IDE e selezionare il tipo di Arduino che si userà. Ecco come fare con la versione 1.8.9:



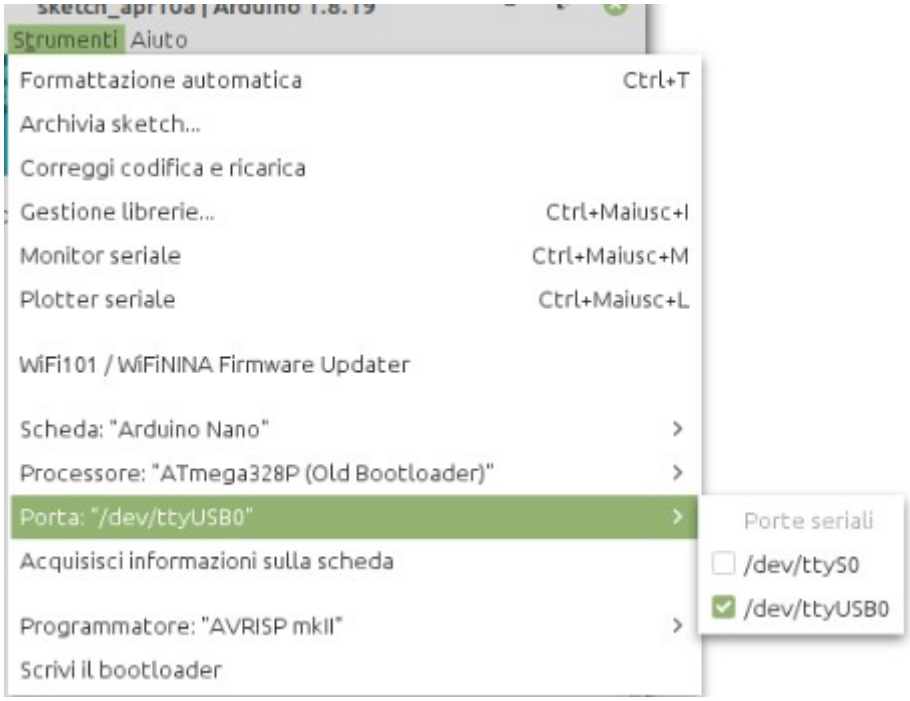
Poi è necessario selezionare il tipo di bootloader (opzionale), se il proprio Arduino è aggiornato all'ultima versione.



Abitualmente, se si acquista ora uno o più Arduino Nano, dovrebbero funzionare con l'ultimo bootloader. Tuttavia, se si acquistano quelli molto economici, per esempio da uno dei vari siti cinesi, montano a volte ancora il vecchio bootloader. In questo caso, è necessario selezionare "ATmega328P (Old Bootloader)", altrimenti si resterà frustrati perché la IDE darà sempre un errore nel caricamento dello sketch su Arduino.

Personalmente ho acquistato Arduino molto economici, che funzionano perfettamente dopo aver effettuato questa soluzione. Ne ho anche acquistate alcune serie da Amazon, realizzati da Elegoo (che trovo leggermente più cari, ma la ditta mi sembra molto affidabile, anche per i kit) che hanno sempre funzionati al primo colpo, senza alcun intervento. Nel caso interessi, ecco il link: Arduino Nano Elegoo

L'ultimo aspetto da verificare è la porta seriale da usare.



In base al proprio computer, si possono trovare una o più seriali. Verificare quella che permette il collegamento ad Arduino.

Clicca qui per informazioni (dal sito ufficiale di Arduino) sull'installazione dell'IDE 1.8.9 con Linux.
P.s.: a volte con Linux non si ha l'autorizzazione per usare le porte seriali, e quindi non si riesce a caricare lo sketch su Arduino. Nella seconda parte delle istruzioni, insegna come ottenere questi permessi.

Clicca qui per informazioni (dal sito ufficiale di Arduino) sull'installazione dell'IDE 2.0

Come iniziare ad acquistare un certo numero di moduli?

Per chi inizia a usare Arduino, c'è il dubbio di quali moduli acquistare e dove.
In questo manuale ne sono stati usati alcuni e naturalmente ce ne sono tanti altri, ma in linea di massima quelli analizzati sono tra i più comuni (ed economici), e permettono di assemblare un gran numero di combinazioni. Le offerte sono tantissime, e possono anche confondere, quindi la prima domanda potrebbe essere: ***quali moduli acquistare?***
In genere ogni singolo modulo ha un costo modesto, di pochi euro, ma se ne vogliamo acquistare a decine, i costi diventano sensibili. Per fortuna, su Internet alcune ditte hanno pensato di assemblare dei kit contenenti molti dei moduli utilizzati, a un costo modesto, decisamente inferiore a quanto si spenderebbe negli acquisti singoli. In questa sezione non ho considerato i kit in cui c'è anche Arduino (abitualmente Uno o simili), perché i componenti elettronici in genere sono "nudi", ovvero non sono montati su basetta e per questo meno utilizzabili con la nostra *bs*. I led, per esempio, necessitano di una resistenza per diminuire la corrente e non bruciarsi rapidamente. Per quelli montati come moduli su una piccola basetta essa è già presente, e questo in linea di massima vale per una grande varietà di componenti, che possono quindi essere direttamente utilizzati, senza altra preoccupazione che montarli con i piedini nella corretta sequenza e alimentarli alla giusta tensione.

Presento qui tre kit, che sono abbastanza comuni e che ho acquistato personalmente.
Premetto che non ho alcun interesse personale nel consigliare una marca o un fornitore piuttosto che un altro; le mie valutazioni sono semplicemente in base alla mia esperienza diretta.

KY	HW	other	description	Elegoo 37 sensor kit V2	Others 37 in 1 kit *	45 in 1 kit
KY-001	HW-506	18B20	temperature-sensor-module	☑	☑	☑
KY-002	HW-513		vibration (shock) switch module	☑	☑	☑
KY-003	HW-495		Hall magnetic sensor		☑	☑
KY-004	HW-483		Key switch module (button)	☑	☑	☑
KY-005	HW-489		Infrared trasmitter module	☑	☑	☑
KY-006	HW-508		Passive buzzer module	☑	☑	☑
KY-007		HC-SR0501	Pir module	☑		
KY-008	HW-493		Laser led	☑	☑	☑
KY-009	HW-478		SMD RGB 3 colours led	☑	☑	☑
KY-010	HW-487		Opto light switch (photo interrupt)	☑	☑	☑
KY-011	HW-480		Two colours led	☑	☑	☑
KY-012	HW-512		Active buzzer module	☑	☑	☑
KY-013	HW-498		Analogue temperature module		☑	☑
KY-014		GY65 – BMP085	Atmospheric pressure sensor			
KY-015	HW-507	DHT11	Temperature & humidity sensor	☑	☑	☑
KY-016	HW-479		RGB tree colours led	☑	☑	☑
KY-017	HW-505		Mercury tilt switch		☑	☑
KY-018	HW-486		LDR – photoresistor – analogue output	☑	☑	☑
KY-019	HW-482		Relay	☑	☑	☑
KY-020	HW-501		Tilt sensor	☑	☑	☑

KY-021	HW-497		Reed switch sensor		☑	☑

KY	HW	other	description	Elegoo 37 in 1 kit	Others 37 in 1 kit *	45 in 1 kit
KY-022	HW-490		Infrared receiver	☑	☑	☑
KY-023	HW-504		Joystick	☑	☑	☑
KY-024	HW-509		Hall effect sensor module plus	☑	☑	☑
KY-025	HW-484		Reed relay – magnetic spring digital output module plus	☑	☑	☑
KY-026	HW-491		Flame sensor module plus	☑	☑	☑
KY-027	HW-499		Magic light cup module (2 pz)		☑	☑
KY-028	HW-503		Temperature sensor module plus	☑	☑	☑
KY-029	HW-477		Two colours led module		☑	☑
KY-030		MQ-2	Gas sensor module			
KY-031	HW-500		Hit (Tap/shock/vibration) sensor	☑	☑	☑
KY-032	HW-201		Obstacle avoiance detector module plus	☑	☑	☑
KY-033	HW-511		Line (tracking) follower module plus	☑	☑	☑
KY-034	HW-481		seven colours automatic flash led module	☑	☑	☑
KY-035	HW-492		Hall effect sensor		☑	☑
KY-036	HW-494		Metal touch sensor module plus	☑	☑	☑
KY-037	HW-485		Amplified microphone sound sensor (big sound) module plus	☑	☑	☑
KY-038	HW-496		Microphone sound sensor (small sound) module plus	☑	☑	☑
KY-039	HW-502		Heart pulse rate detector		☑	☑
KY-040	HW-040		Rotary encoder	☑	☑	☑
KY-050		HC-SR04	Ultrasonic sensor	☑		☑
		GY-521	accelerator/giroscope module	☑		☑
		DS-1307	RTC module	☑		
		LCD1602	LCD display	☑		
		Sen18	Water level sensor	☑		☑
			4 x 4 Membrane leyboard	☑		
			Power supply module	☑		☑
		DS1302	RTC clock module			☑
			SD card reader			☑
			Buck converter (alimentation)			☑
			Soil moisture sensor			☑

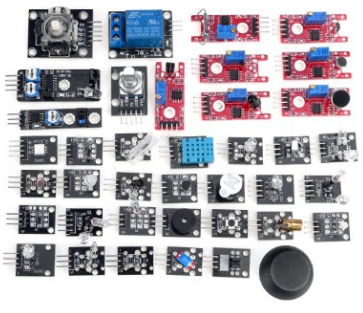
La seconda domanda è: quale kit acquistare e da chi?
Naturalmente possiamo acquistare dove preferiamo; però è bene tenere conto di alcune considerazioni generali. Se comperiamo i nostri kit da fornitori "storici", abbiamo la sicurezza di tempi certi di consegna, della possibilità di restituirli qualora avessero dei difetti, o semplicemente perché non corrispondono alle nostre aspettative; per contro il costo è leggermente superiore. Ci sono vari siti dell'Estremo Oriente che propongono dei prezzi molto competitivi, ma è necessario controllare attentamente quanto offrono, e non soltanto leggere il titolo, guardare il prezzo e presi dall'entusiasmo cliccare su "acquista subito". A volte si incorre poi in brutte sorprese, e parlo per esperienza diretta. E' bene ricordare che spesso cliccando sulle varie foto presenti sulla stessa pagina del sito, cambiano i kit e di conseguenza le offerte e i costi! Se tutto corrisponde e i prezzi sono buoni, è bene sapere che i tempi di consegna sono abitualmente più lunghi (da una decina di giorni fino a 30/40 giorni) e della difficoltà nel caso si dovesse restituire il materiale; in alcuni casi il costo della spedizione è a carico dell'acquirente e i tempi di sostituzione o di rimborso non sono sempre brevi. Attenzione a considerare anche i costi del trasporto. Se si ordina dal Regno Unito (Gran Bretagna) e da alcuni altri paesi fuori della comunità europea, a volte si devono pagare delle tasse per ritirare la merce, quindi è bene informarsi di eventuali costi nascosti ed essere informati delle varie possibilità.

Relativamente alla qualità dei kit:

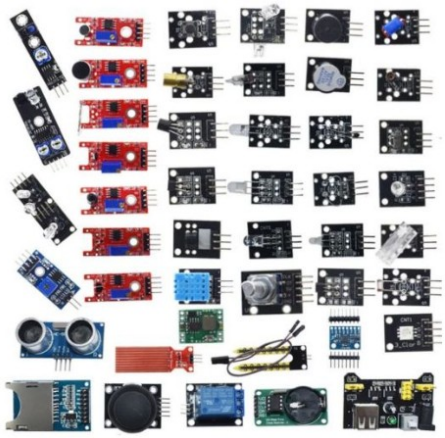
il primo che ho acquistato in ordine di tempo è stato quello della Elegoo, dal nome "37 sensor kit V2".
E' sicuramente un ottimo kit. Arriva in una bella scatola, con un cd in cui si trovano sia i datasheet dei singoli moduli e i programmi allegati vengono dettagliatamente spiegati. Inoltre ci sono alcuni moduli che danno valore al kit: per esempio il display LCD, il modulo orologio, l'accelerometro/giroscopio, il sensore PIR, un centinaio di resistenze assortite, ecc.; oggetti che non sono presenti in altri kit più economici. Il costo attualmente (febbraio 2023) si aggira intorno ai 35/38 euro.

Il kit "37 sensor kit V2" della Elegoo

Il secondo kit, sempre di 37 moduli si trova anche a 15/20 euro acquistandolo dalla Cina (Banggood, Ali Express, ecc.). Va sicuramente bene, i moduli sono perfettamente funzionanti, ma sono arrivati in una semplice busta di plastica non imbottita e alcuni moduli erano un po' acciaccati. I sensori erano in bustine singole. Non c'era alcun cd o rimando a un sito che dia qualche informazione sul loro uso, per cui bisogna sapere come usarli.


Il kit "37in 1"

Personalmente ho trovato un sito che ne dà una descrizione sbrigativa e anche un programma dimostrativo per ogni oggetto al link:https://www.instructables.com/Arduino-37-in-1-Sensors-Kit-Explained/
Al solito, questa informazione viene fornita senza alcuna garanzia relativa al sito stesso.


Uno dei kit "45 in 1"

Il terzo kit trovato su Internet alla dicitura " arduino kit 45 in 1" assomiglia molto a quello precedente, ma ha qualche modulo in più che lo impreziosiscono. I vari kit di questo tipo in vendita su Internet non sono del tutto uguali, quello che segnalo l'ho trovato su "Banggood". Altri kit cambiano per qualche componente, ma mi sembra che nessuno, per esempio, comprenda il dispay LCD 1602 molto utile in tanti progetti. Il costo si aggira mediamente tra i 22 e i 27 euro, e l'imballo e le istruzioni sono simili a quelle del kit precedente. Dovendo scegliere tra queste due ultime offerte, è sicuramente più conveniente il kit 45 in 1.

Per acquistare Arduino Nano: nel caso non si possieda almeno un Arduino Nano, è necessario acquistarlo, tenendo conto delle riflessioni generali precedenti, sempre valide quando si acquista su Internet.

Arduino Nano

Sempre per esperienza personale, consiglio di non acquistarne uno solo, per due motivi:
- il rapporto quantità/prezzo non sempre è conveniente per una singola unità;
- all'inizio è facile che un piccolo cortocircuito distrugga in un attimo il nostro prezioso microcontroller…

Tanto per non fare nomi, su Amazon si trovano tre Arduino Nano a un prezzo compreso tra i 23 e i 28 euro.
Inoltre fare attenzione: a volte vengono venduti a un prezzo inferiore, ma con i connettori da montare, come mostrato nella foto; in questo caso bisogna saper usare bene il saldatore, per evitare costosi pasticci…
Ancora per esperienza personale: non sempre è facile far colloquiare il proprio "Nano" con il computer. Ho avuto difficoltà anche con i modelli acquistati sul sito di Arduino, mentre funzionano al primo colpo quelli di marca Elegoo. Ribadisco che non ho alcun interesse a proporre questa società, ma i suoi prodotti sono sicuramente di qualità…

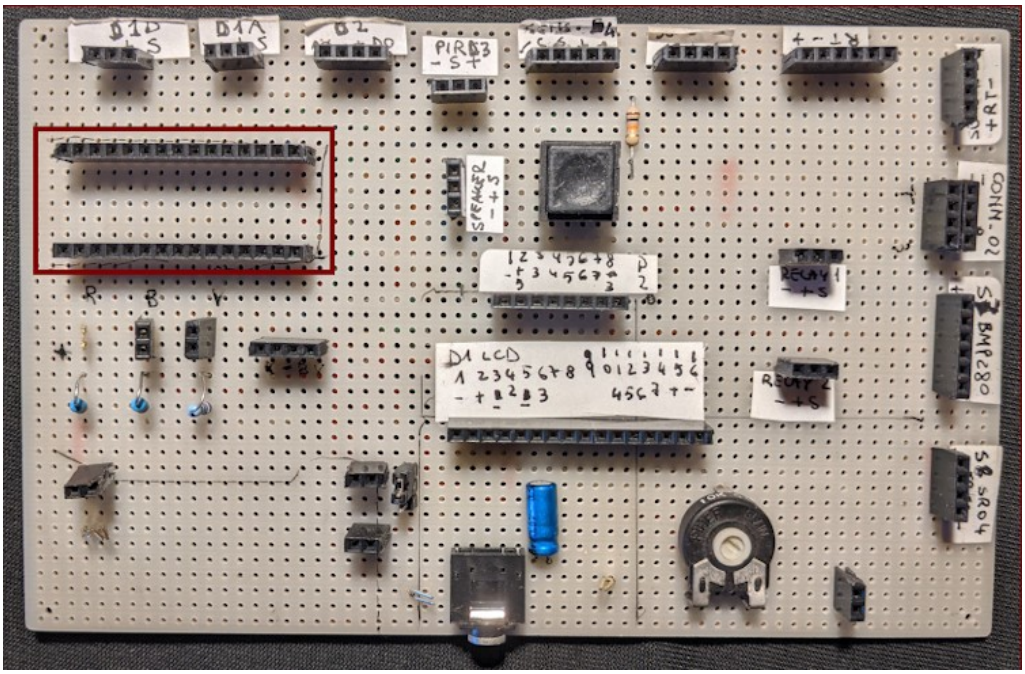
Quindi con una spesa oscillante tra i 50 e 75 euro possiamo entrare in possesso di alcuni Arduino e una bella manciata di componenti, e se il nostro budget si aggira intorno ai 100 euro, abbiamo la possibilità di acquistare ancora qualche modulo singolo per impreziosire i nostri progetti.

Nota: questi non sono consigli per acquisti, ma semplicemente la condivisione di alcune esperienze personali e dirette. Pertanto non forniscono alcuna garanzia sugli acquisti stessi, di cui non ami assumo alcuna responsabilità.

La genesi del progetto.

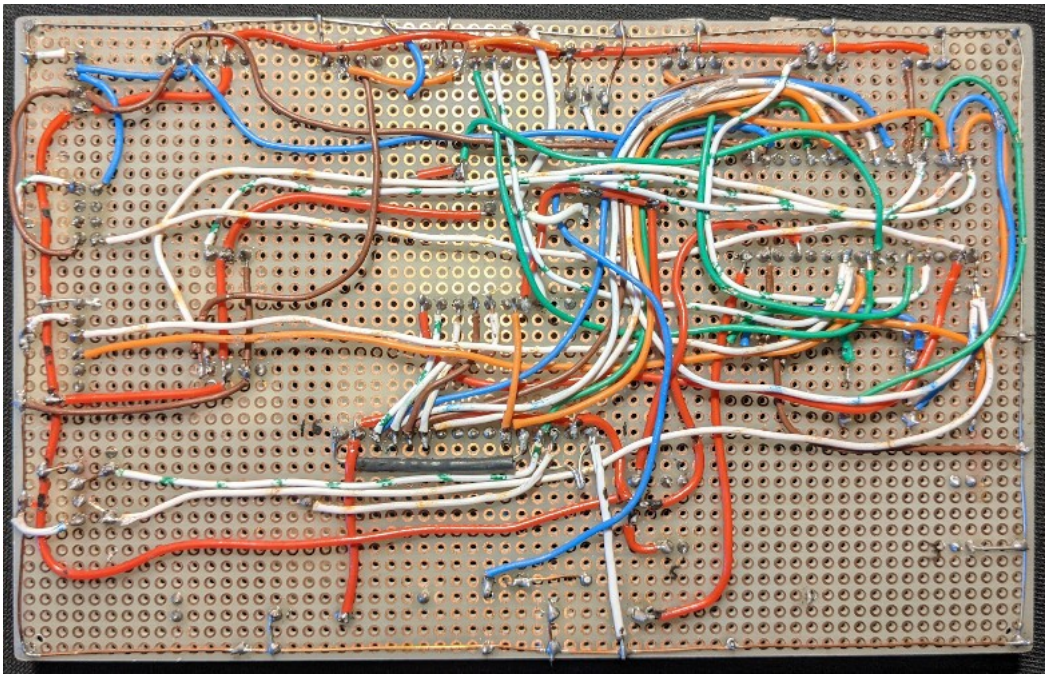
Nella primavera del 2022 sono andato in pensione, e allora ho avuto il tempo necessario per attuare un progetto che avevo in testa da tempo: creare una basetta sperimentale e didattica per Arduino Nano. Ero abbastanza scettico sulle mie capacità e la possibilità di realizzare un simile progetto; ma l'unico modo per verificarlo era di tentare. E così ho ripreso in mano il saldatore…

Prima versione. Maggio 2022



vista anteriore…

P.s.: lo zoccolo di Arduino è indicato dal rettangolo rosso scuro.

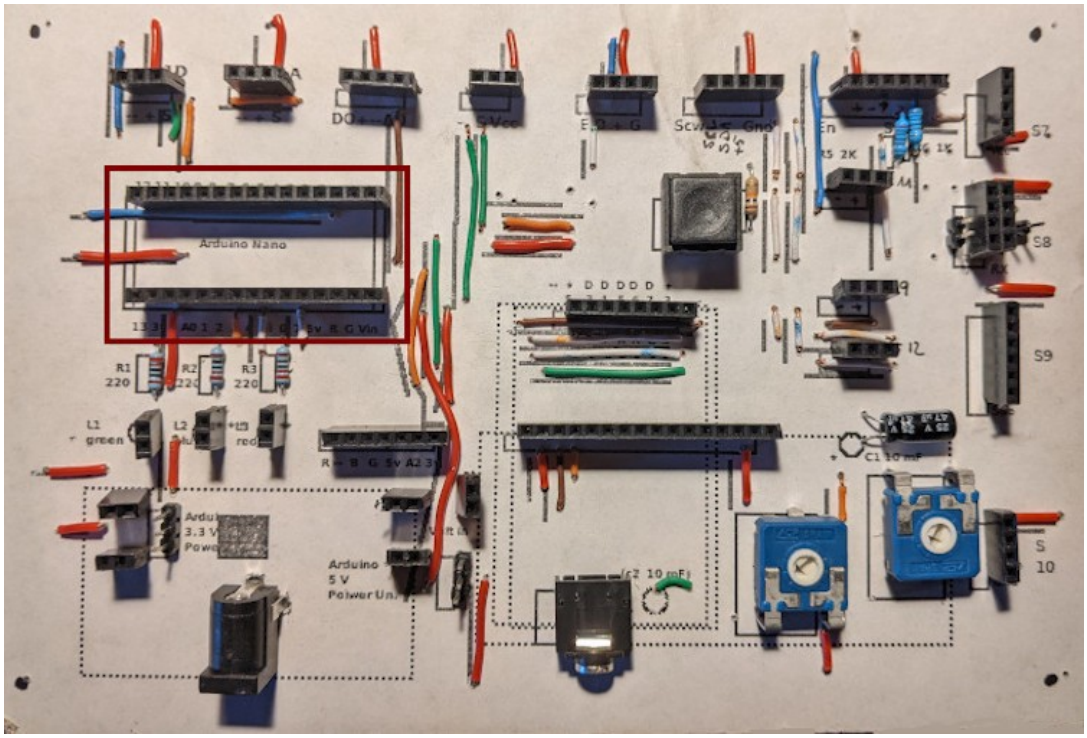


… e vista posteriore

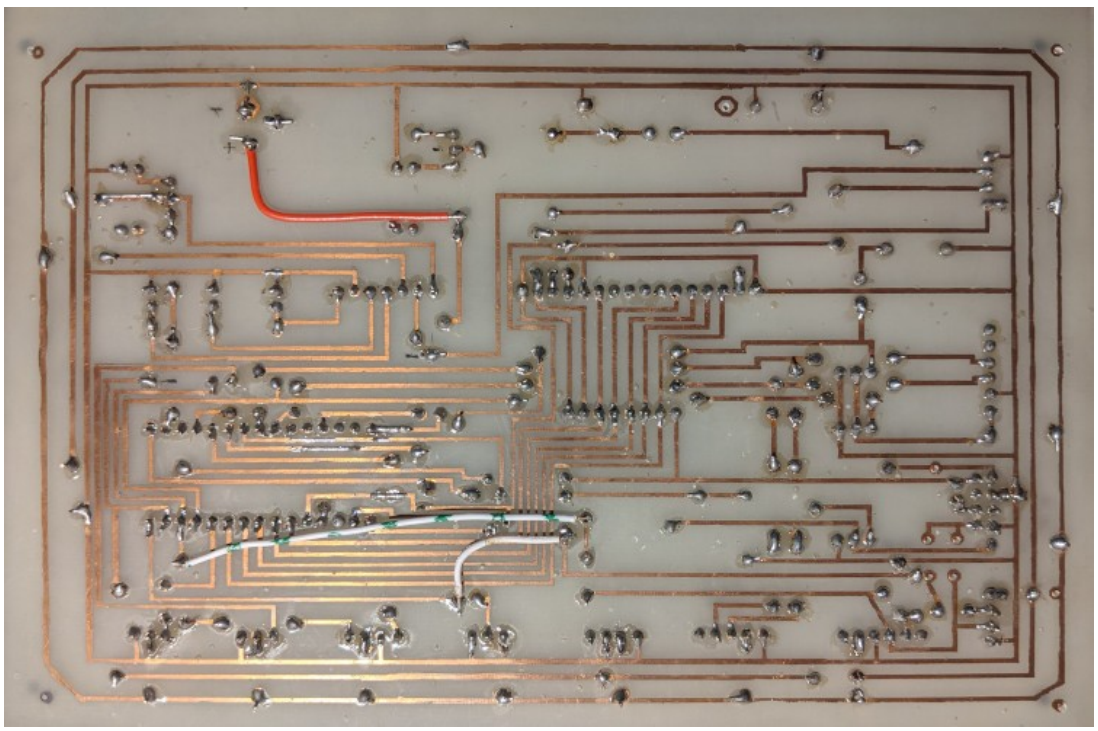
Il fatto che funzionasse, nonostante tutto il groviglio di fili, ha stupito anche me, e mi ha incoraggiato a proseguire l'esperimento.

Seconda versione, giugno 2022.

La seconda versione della bs è stata più strutturata; il cablaggio è stato ottenuto disegnando e realizzando in modo casalingo il primo circuito stampato. Intanto si è cominciato a testare alcuni programmi e tenere traccia delle informazioni con un file di word processor. Il programma usato per disegnare il circuito stampato non era proprio specifico… Usai Gimp, ma il risultato è stato comunque abbastanza soddisfacente. Si notino tutti i ponticelli necessari presenti sulla faccia superiore della basetta.



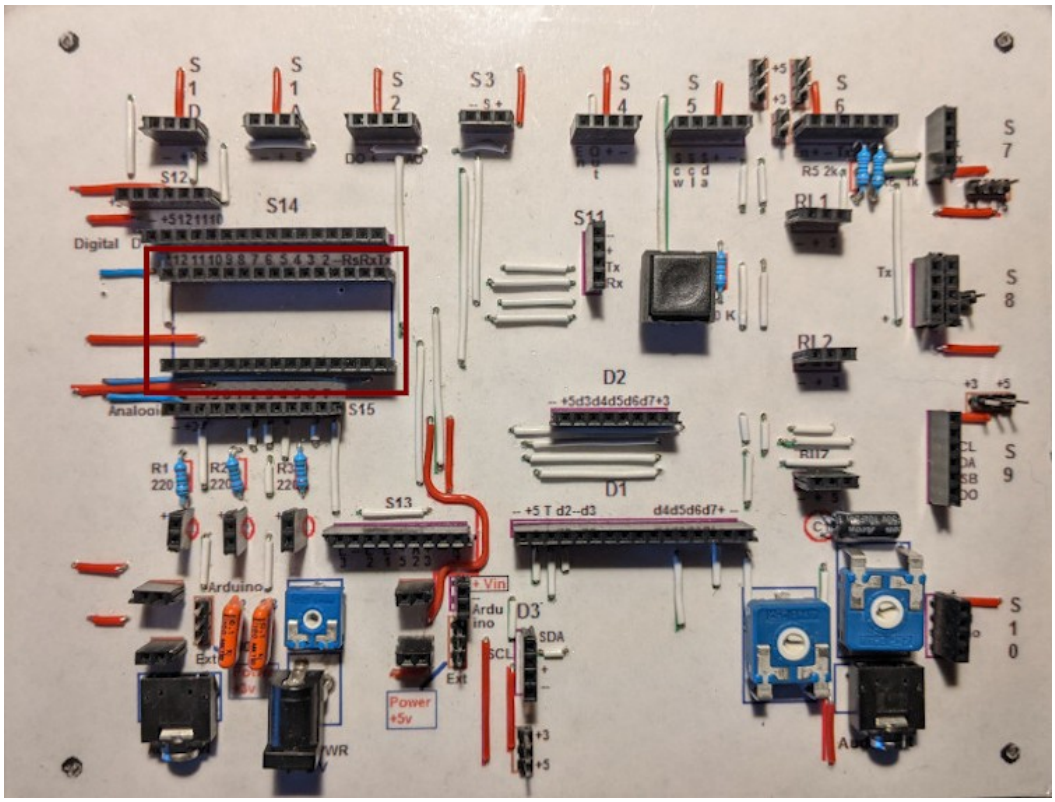
Vista superiore



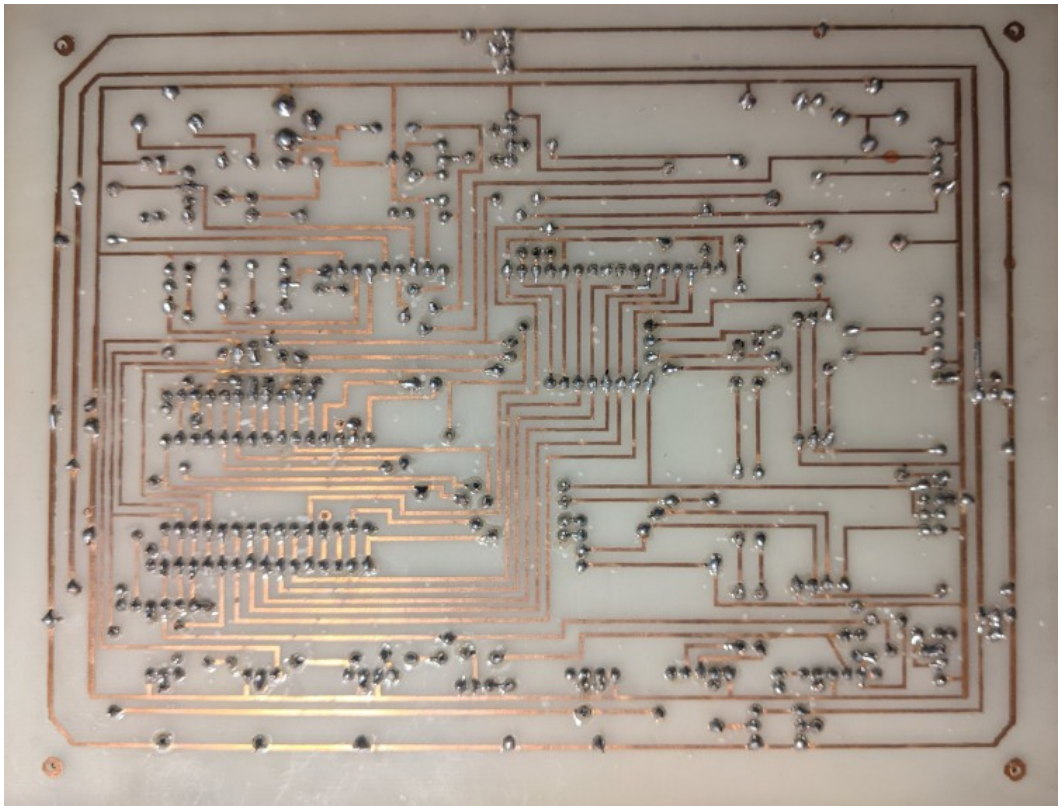
Vista inferiore

Terza versione, settembre 2022

La terza versione della bs è un perfezionamento della seconda: vengono aggiunti alcuni zoccoli tra cui quelli laterali ad Arduino; un ingresso audio, incrementando un poco le dimensioni della basetta stessa.



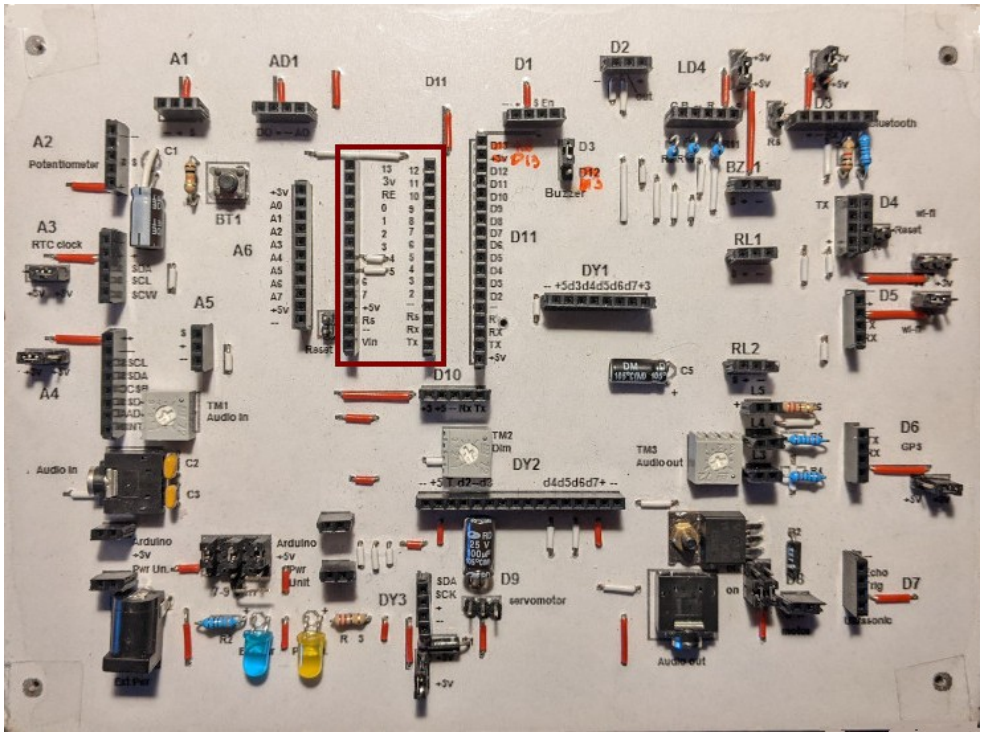
Vista superiore



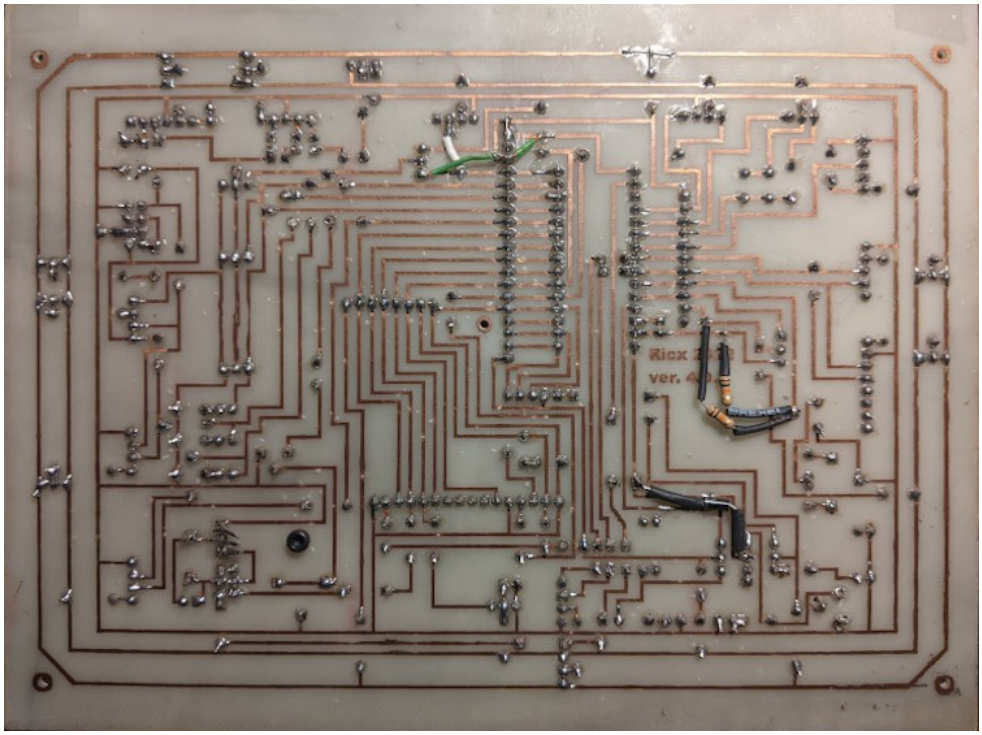
Vista inferiore

Quarta versione, dicembre 2022/ febbraio 2023

Con la quarta versione si è ridisegnato radicalmente la *bs*: Arduino non è più in posizione laterale, ma in alto, quasi centrale; le porte analogiche sono state spostate a sinistra, mentre quelle digitali a destra, ottenendo una certa razionalizzazione dello stampato, arricchendolo di nuove possibilità e aumentando ancora un poco le dimensioni della bs, ora grande quasi quanto un foglio A5 (la metà di un A4).



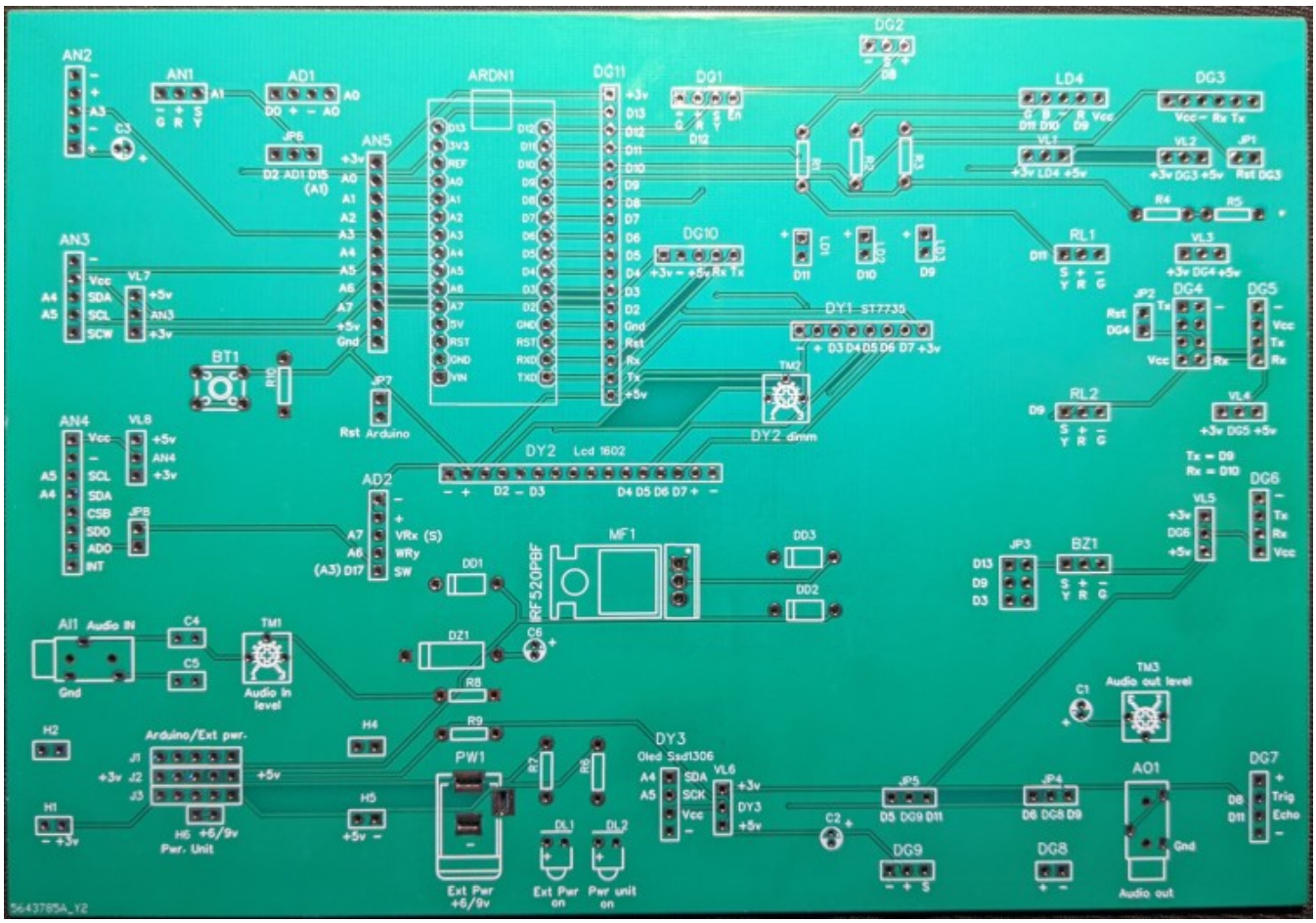
Vista superiore



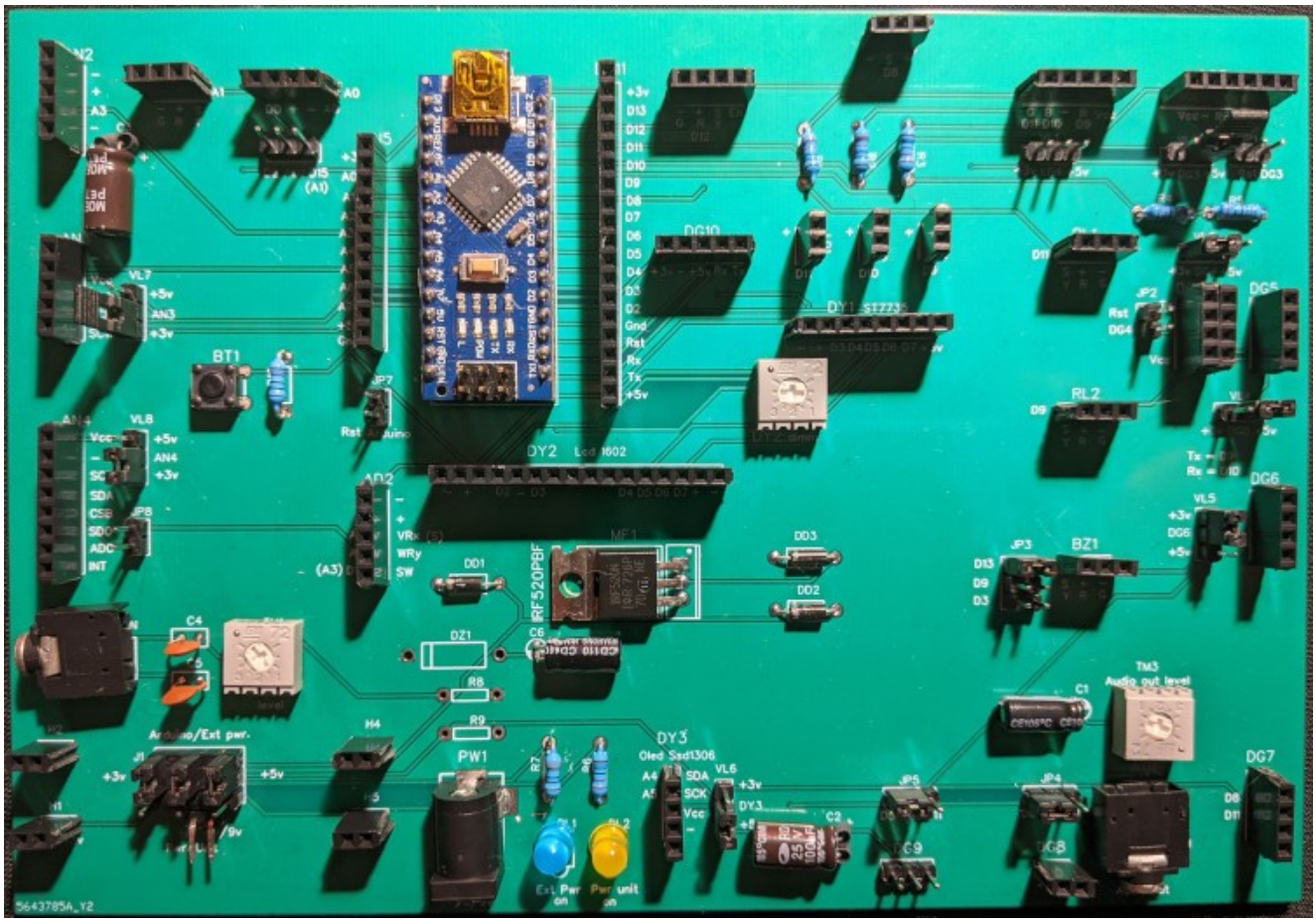
Vista inferiore. Si notano alcune modifiche "volanti".

Quinta versione, marzo 2023

Ormai il progetto era maturo, ma l'esecuzione della basetta casalinga, il trasferimento del tracciato del circuito sulla basetta ramata, seguito dal passaggio nell'acido, l'applicazione del foglio con la serigrafia, la foratura (centinaia di fori!), poi la saldatura dei ponticelli, degli zoccoli e dei componenti richiedeva un numero elevato di ore, e la realizzazione del progetto diventava problematica se eseguita da altre persone, e quindi di diffusione piuttosto difficoltosa. Allora, anche su incoraggiamento di mio figlio e tenendo come base le ulteriori evoluzioni della quarta versione della *bs*, ho ridisegnato integralmente il progetto con un programma gratuito presente in rete, dal nome di Easy Eda, con cui ho preso rapidamente confidenza e ho realizzato il progetto attuale. Dopo aver verificato che non ci fossero errori nel progetto, ho ottenuto i file gerber per la realizzazione pratica della *bs*. Utilizzando sempre Easy Eda ho inviato i file per la creazione della basetta a una ditta a loro collegata, dal nome di JLCPCB. In poco più di una settimana ho ricevuto le basette, costruite in modo impeccabile e in una confezione eccellente.
P.s.: ritengo che i file gerber siano uno standard, e possano essere inviati per la realizzazione pratica anche ad altre ditte specializzate.



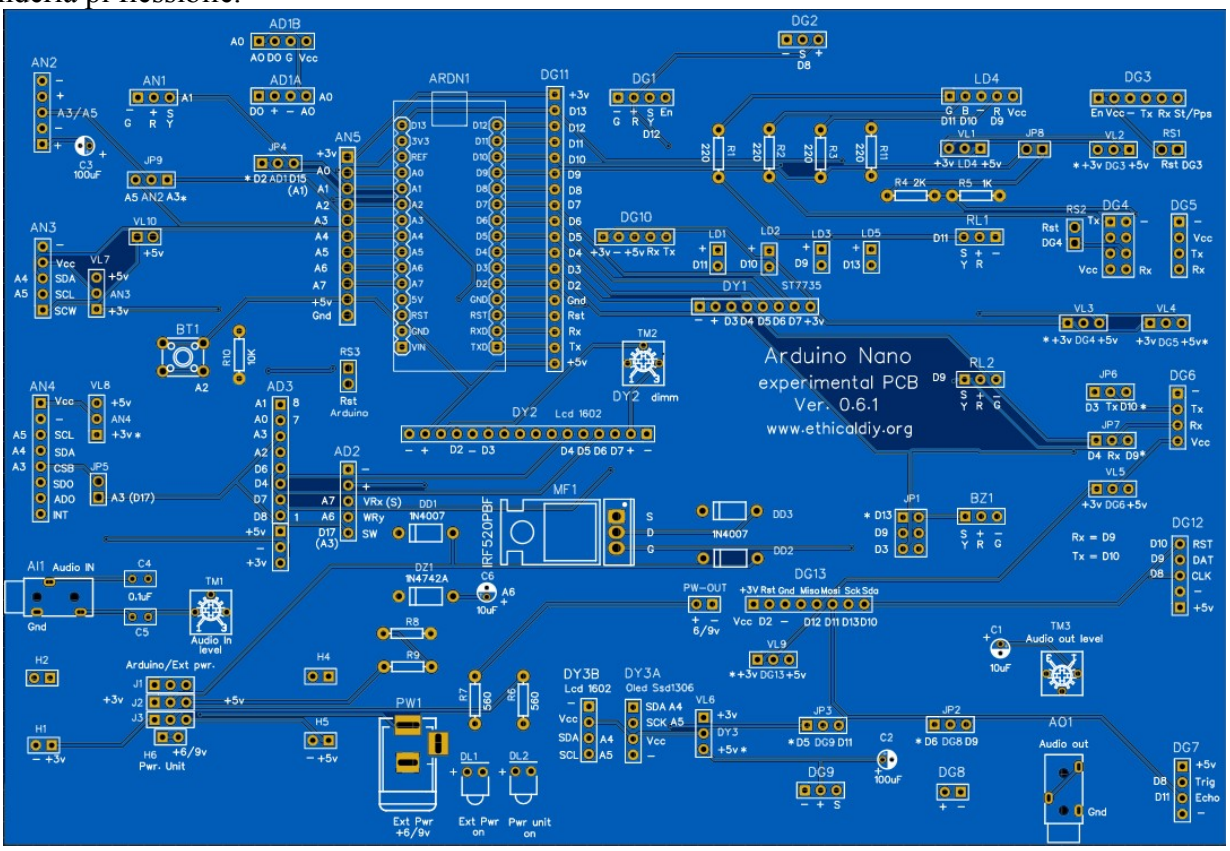
La quinta versione della basetta nuda, in tutto il suo splendore...

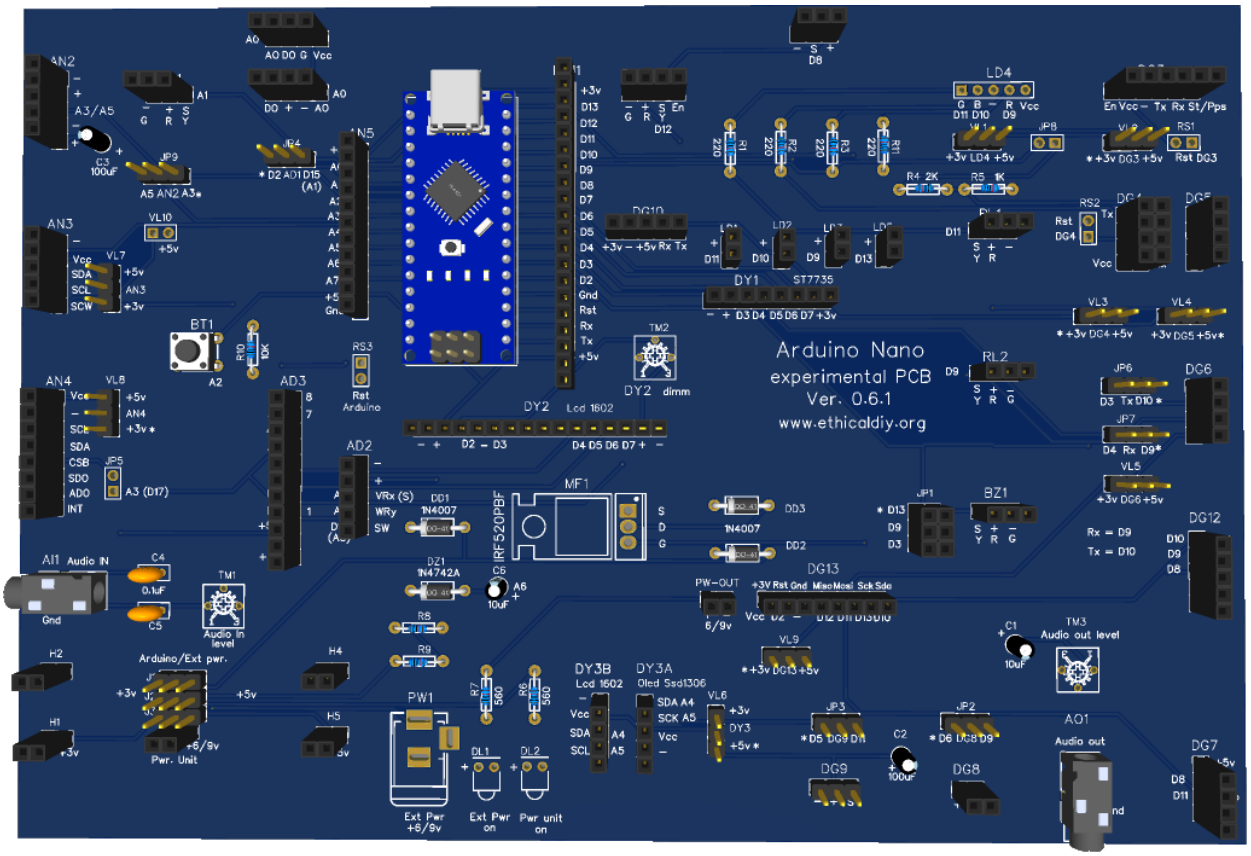


… ed eccola montata, pronta a una miriade di esperimenti e nuovi progetti.

Sesta (e per ora ultima) versione, maggio 2023

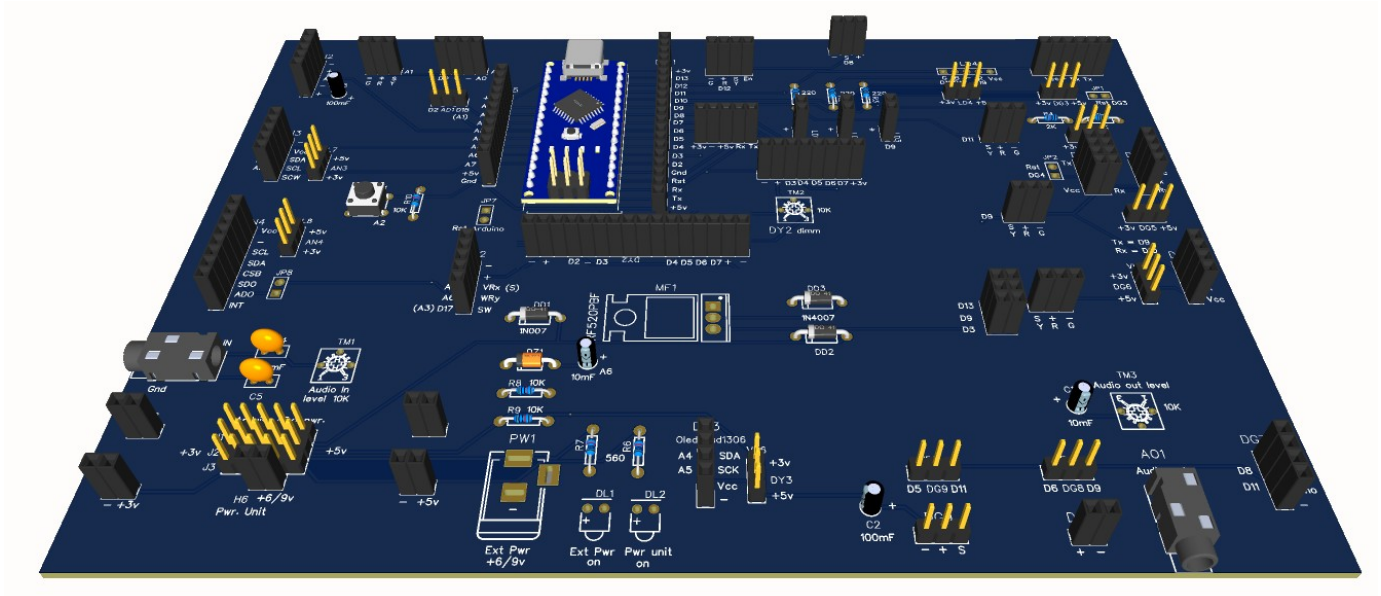
Dopo aver testato positivamente la quinta versione della bs, che è anche la prima realizzata professionalmente, sono state apportate alcune migliorie ad essa, aggiunti alcuni zoccoli per renderla pi flessibile.





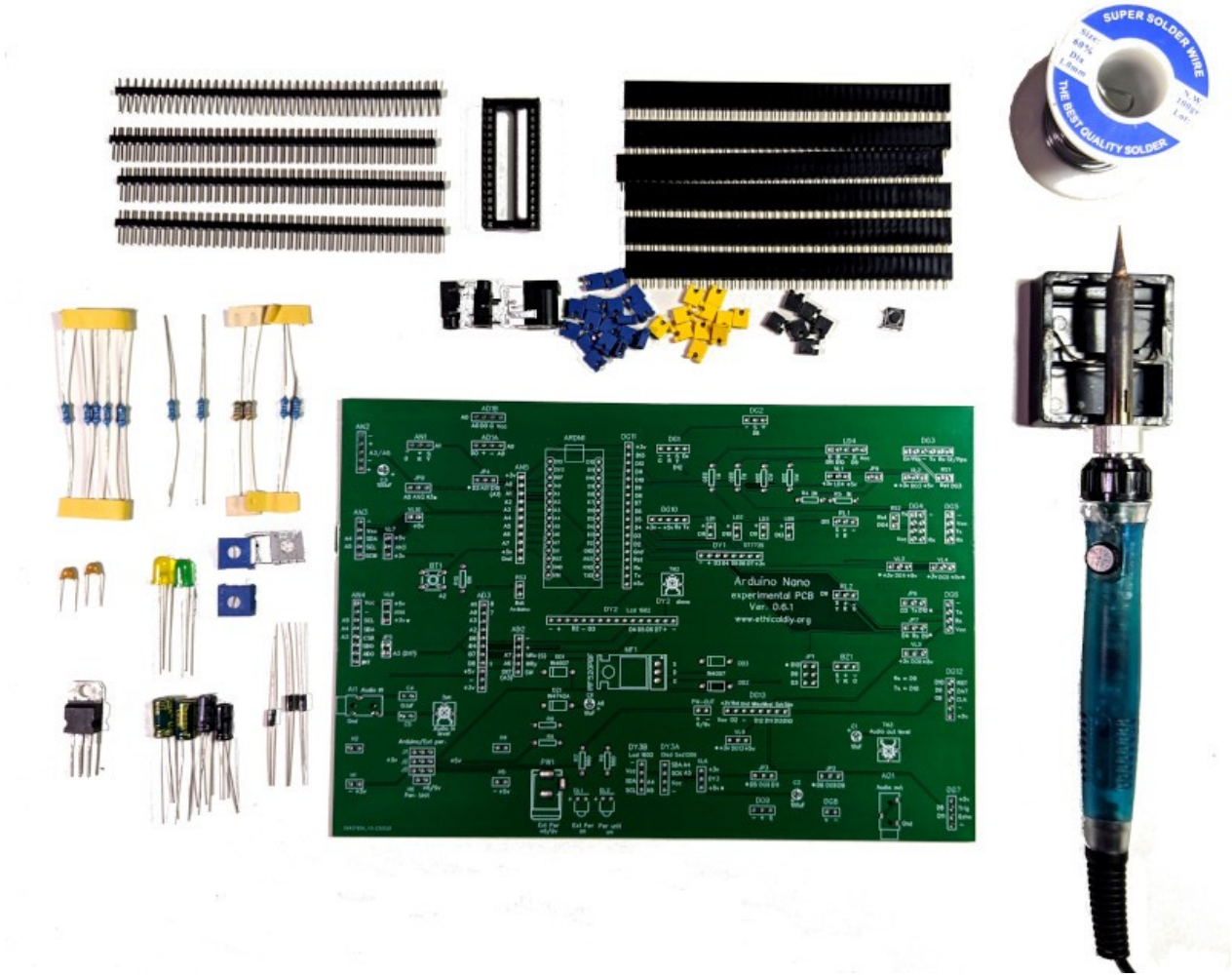
Sezione VI:

Montare la basetta sperimentale



Il montaggio della basetta sperimentale versione 0.6.1

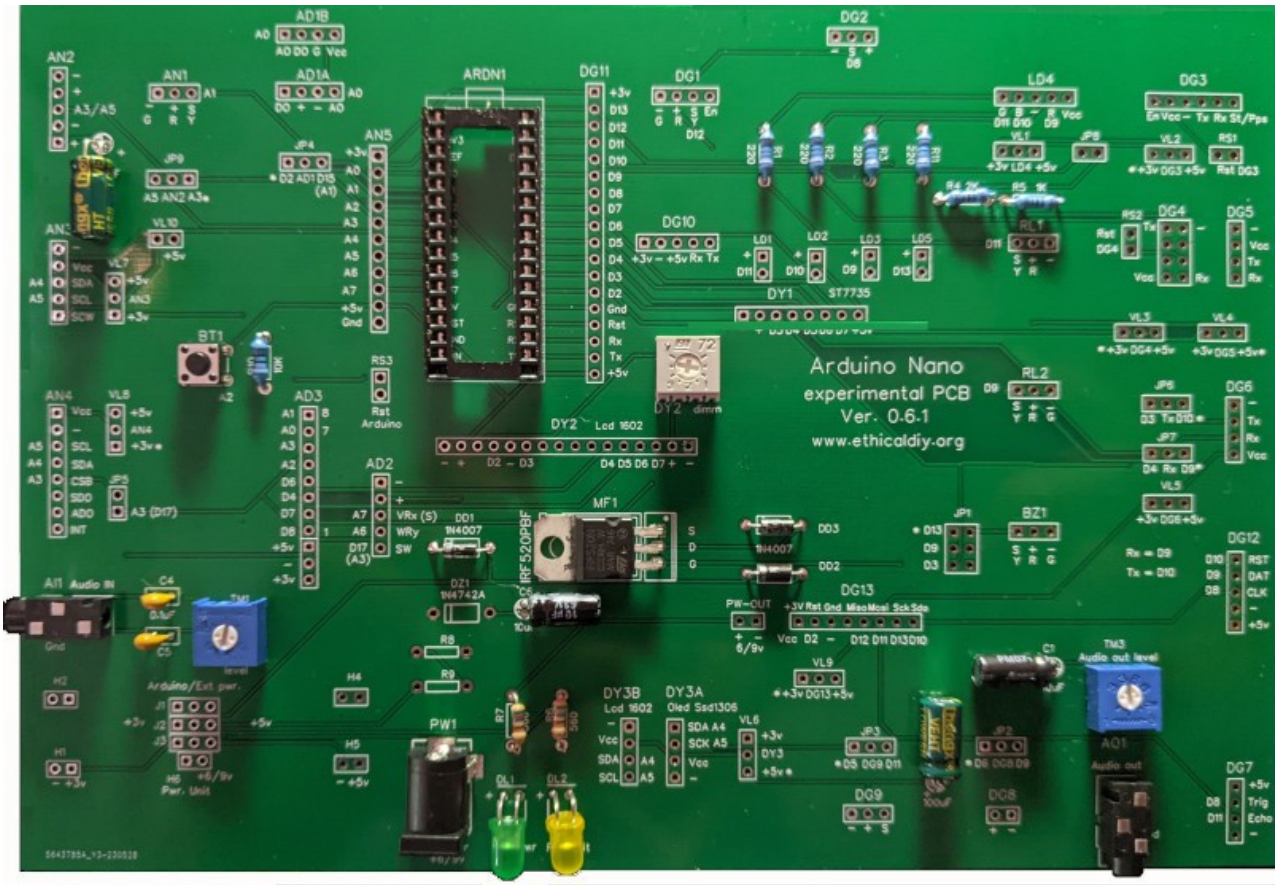
Immagine dei componenti richiesti e degli attrezzi necessari per montare la basetta sperimentale.



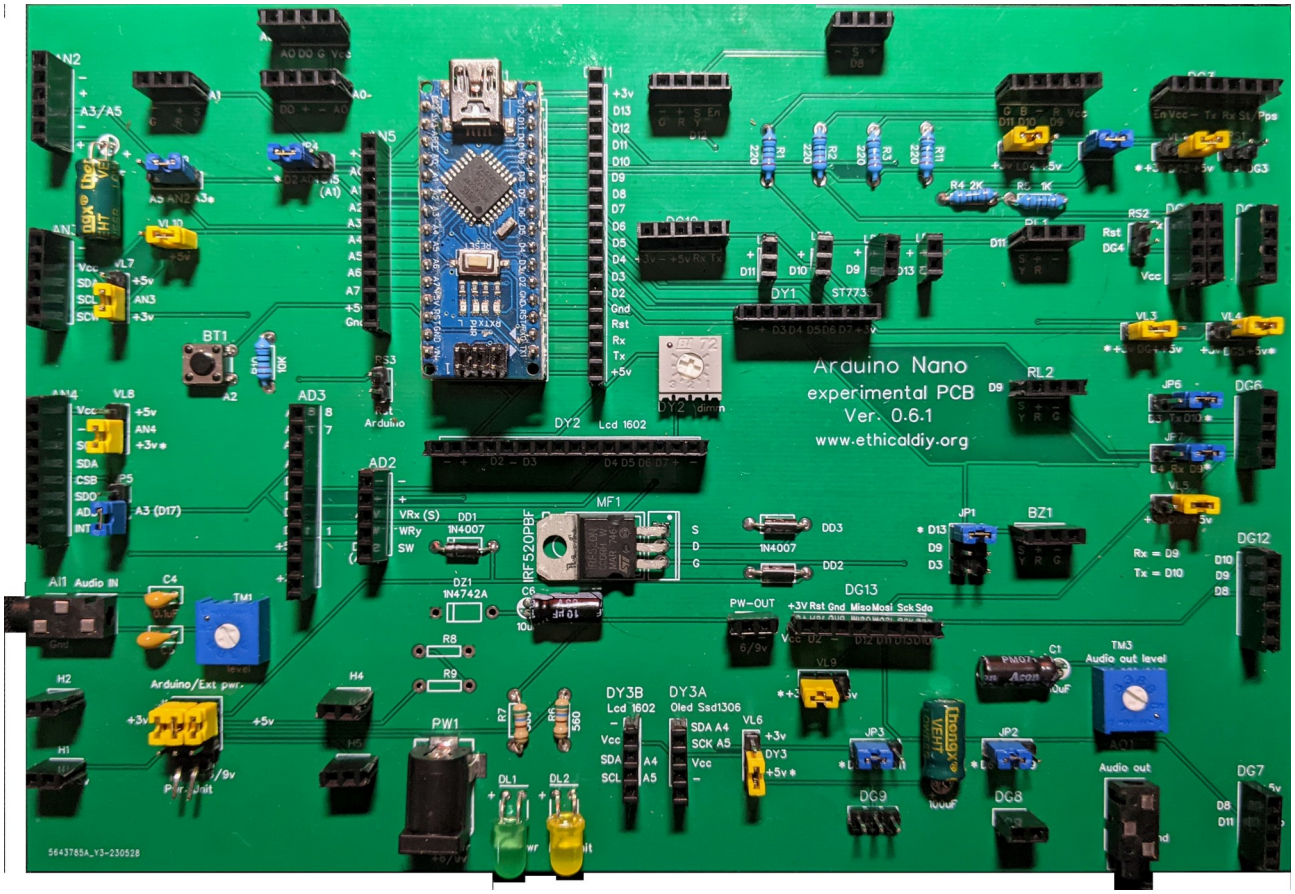
Nell'immagine precedente si vede il piccolo numero di componenti necessari per montare la bs. Gli attrezzi per l'assemblaggio sono veramente pochissimi: un saldatore e lo stagno adatti per saldare piccoli componenti come integrati, Sono necessari anche un tronchesino e una piccola pinza. Potrebbero essere utili, ma non indispensabili, un tester e un accessorio noto come "terza mano", che permette di tenere la basetta sollevata (vedi immagine).

Un tester, la “terza mano”, un tronchesino e una piccola pinza diritta

Operazione 1: montare i componenti



Si comincia a montare i componenti più piccoli, come i componenti elettronici e lo zoccolo per Arduino



… e poi si procede montando i connettori, sia maschi che femmine.
Ed ecco la nostra basetta sperimentale in tutto il suo splendore, con anche Arduino Nano a bordo!

2: alcune considerazioni

Per assemblare con successo la nostra *bs*, è necessario dare alcune indicazioni:

- ci sono circa 350 punti di saldatura per completare l'assemblaggio della basetta, e il passo tra i vari pin è quello standard, ovvero 2,54 mm. Per cui è necessario un buon saldatore, con la punta sottile e dello stagno adatto, anch'esso sufficientemente fine e di qualità. Naturalmente è necessaria anche un pò di manualità e di esperienza nella saldatura…
- E' necessario saldare molti connettori, su cui si inseriranno i vari moduli per testare i nostri programmi. E' possibile acquistarli su internet già tagliati di misura, ovvero da 3, 6, 8… 16 pin. Però in questo modo si rischia di non trovarli tutti, o di spendere parecchio. Consiglio di acquistare le strisce standard, da 40 connettori e poi con un tronchesino di tagliarli di misura. In questo modo si è autonomi e si spende di meno.
- I connettori che iniziano per Adx, Anx, Dgx, Hx, sono tutti femmina, tranne DG9 che è maschio, perché la maggior parte dei servomotori hanno un connettore femmina.
- Tutti i connettori che iniziano perVLx, JPx, sono maschi, e servono per eseguire dei settaggi. Relativi alla scelta di tensioni, dell'uso delle porte di Arduino, o per resettare dei moduli.
- **Ricordarsi di inserire i jumper sui connettori VLx, JPx, <u>altrimenti la scheda sembrerà non funzionate!</u> Va posta una cura particolare ai connettori J1, J2, J3, posti in basso a sinistra della scheda. Essi servono per alimentare Arduino e i moduli. (vedi immagini)**

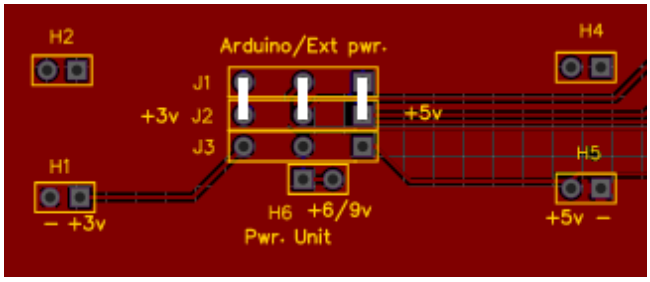


Fig. 1

Alimentare Arduino e i moduli attraverso la USB e/o alimentatore esterno (ext pwr). Dai test effettuati, questi due tipi di alimentazione sono sufficienti per la totalità dei progetti assemblati

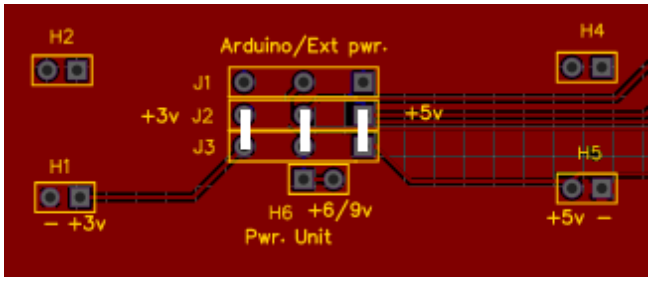


Fig. 2

Alimentare Arduino e i moduli attraverso un alimentazione esterna con modulo tipo "Power unit" della Elegoo (vedi il paragrafo "Alimentazione di Arduino").
Questa scelta permette di alimentare anche moduli piuttosto "energivori", ma non si è mai dimostrata strettamente necessaria.

3: la lista dei componenti

codice su bs	componente	codice commerciale	descrizione
AI1, AO1	PJ-320		jack standard 3,5 mm
C1, C6	10 µF 25v		elettrolitico
C2, C3	100 µF 25v		elettrolitico
C4, C5	0,1 µF		ceramico
DD1, DD2, DD3	1N4007		diodi
DL1	Led verde		led
DL2	Led giallo		led
DZ1	1N4742A		diodo zener 4,7 v
MF1	IRF520		Mos-fet
R1, R2, R3, R11	220 Ω 1/8 W		resistenze
R4	2 KΩ 1/8 W		resistenze
R5	1 KΩ 1/8 W		resistenze
R6, R7	560 Ω 1/8 W		resistenze
R8, R9, R10	10 KΩ 1/8 W		resistenze
TM1, TM2, TM3	10 KΩ 1/8 W	Trimpo 3362 500R	trimmer resistivi
PW1	KLDX-0202-B	KLDX-0202-B	connettore di alimentazione
BT1	Dip tact switch 6x6x4.3 mm	KA-6x6_TH	switch
'---	barrette connettori femmine *	HDR-F-2.54_1x40	necessarie 5 barrette
'---	barrette connettori maschi *	HDR-M-2.54_1x40	Necessarie 2 barrette
Jumper	Jumper cap 2.54mm – height 5 mm		Necessari 20 pz.
ARDN1	zoccolo per IC da 30 pin		zoccolo per Arduino Nano

Note sui componenti

Su internet si trovano anche le barrette delle dimensioni corrette per gli zoccoli, per esempio da 4 connettori, ma sicuramente sono più care e bisogna comprarne parecchie, e di varie dimensioni. Consiglio di acquistare le barrette standard, da 40 connettori, e poi tagliarle della giusta misura con il tronchesino.

Per i componenti che potrebbero creare dei fraintendimenti, sono stati inseriti i codici commerciali con cui sono stati acquistati per montare la basetta sperimentale.

Per esperienza personale, è più conveniente acquistare un certo numero di componenti,piuttosto che il singolo componente, specialmente se si comprano su interrnet.

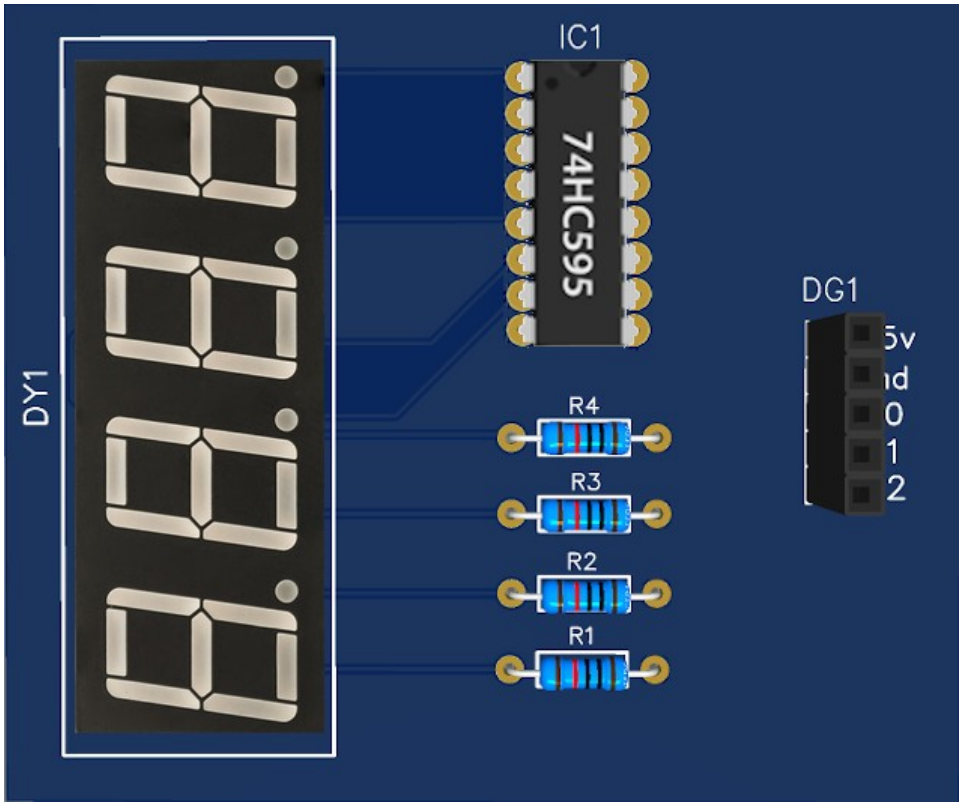
SEZIONE VII:
le schede accessorie

Montaggio della basetta per Dispaly 5641AS con IC 7HC595

Il display 5641AS, se collegato direttamente ad Arduino Nano, richiede ben 8 porte digitali e otto resistenze da 220 Ohm, lasciando poco spazio per altri moduli e/o sensori.

Utilizzando un integrato 7HC595, si usano solamente 3 porte digitali di Arduino, per cui diventa molto più flessibile.

Il montaggio di questa basetta è molto semplice, in quanto sono necessari il display, l'integrato, quattro resistenze e alcuni connettori.



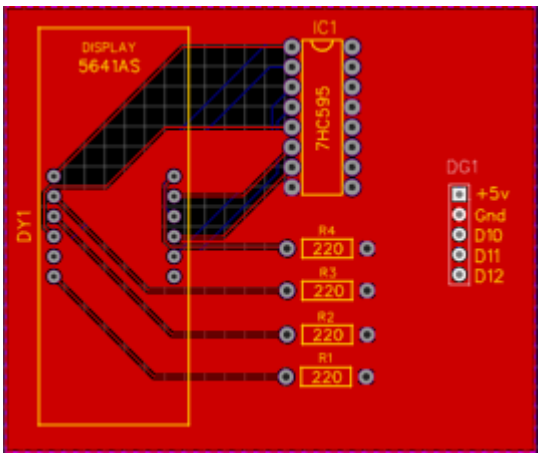
Ecco il render 3d della basetta montata (ingrandita)

Si consiglia di montare l'integrato su di uno zoccolo standard da 16 pin, per non bruciare i delicati circuiti interni saldandolo direttamente sulla basetta.

Il connettore da 5 pin va collegato al connettore DG11 della bs, rispettando sia la polarità di alimentazione che i collegamenti alle tre porte digitali, rispettivamente: D10, D11, D12

Componenti:

- Display 4 digits 5641AS
- Integrato 7HC595
- Zoccolo per integrato a 16 pin (opzionale)
- R1, R2, R3, R4: resistenze 220 Ohm, 1/8 W
- zoccolo DG1 da 5 pin: HDR-F-2.54_1x5

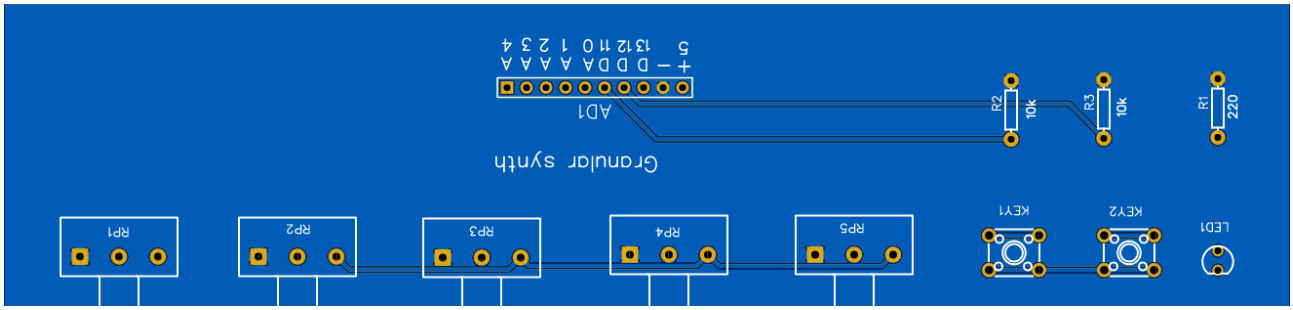


Ed ecco invece la basetta da montare nelle sue reali dimensioni 1:1

Clicca qui per andare al progetto.
Clicca qui per i file ***gerber***
Clicca qui per la pagina del ***download***

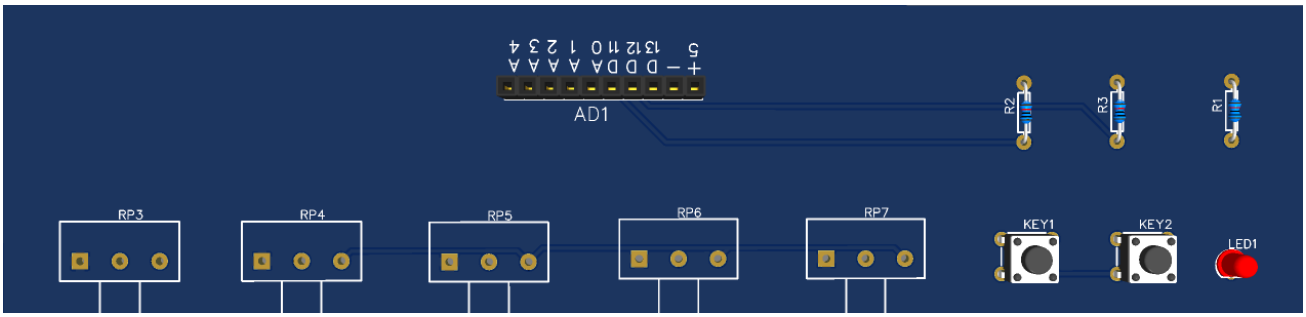
Il montaggio della scheda per il granular synth

Il granular synth richiede un collegamento con cinque potenziometri, ed eventualmente un pulsante, per variare il modo della regolazione della frequenza, e di un led che pulsa in sintonia con la frequenza del suono. I potenziometri richiedono molti fili, e il sistema, specialmente se si desidera usarlo con una certa frequenza, diventa molto complesso. Per questo motivo ho pensato di sviluppare una basetta atta allo scopo. Nella pagina dedicata a questo progetto, si vede la foto di una basetta costruita in modo tradizionale, ricalcando le tracce sulla superficie ramata, poi immergendola nel cloruro ferrico per asportare il rame in eccesso, forata, ecc. In questa sezione invece si trova la versione più professionale, preparata da una ditta specializzata.
Ecco un immagine della basetta:



Un lato di ogni potenziometro è collegato a massa, l'altro lato è connesso a +5 volt. Il connettore centrale va collegato rispettivamente da A0 ad A4. Il pulsante può essere connesso alla porta digitale D12 e il led alla porta D13.

Lo schema prevede anche due pulsanti. Per **granular_synth_1** non servono, mentre per **granular_synth_2** ne è sufficiente uno. Il secondo è libero, e può servire per eventuali successivi sviluppi.
Ecco invece il render della scheda con i componenti. Purtroppo per il programma Easy Eda non ho trovato l'immagine dei potenziometri. Perciò useremo la fantasia…



Per ultimo, la lista dei componenti:

- Potenziometri, da RP3 a RP7: 10 Kohm, lineari, per montaggio su basetta. **Clicca qui** per i potenziometri presso Ali express. ***Questo link non intende sollecitare l'acquisto***.
- R1: 220 Ohm, 1/8 W
- R2, R3: 10 Kohm, 1/8 W
- Key1, Key 2: pulsanti da basetta, 6x6x4,3 mm; cod. commerciale: KA-6x6_TH
- Led1: un led (colore a piacere)
- AD1: connettore femmina a 10 poli, cod. commerciale: HDR-F-2.54_1x10

Sarà poi necessario un cavetto a 10 poli, con connettori maschio/maschio, per collegare la basetta dei potenziometri a quella principale, rispettando la sequenza delle porte indicate.

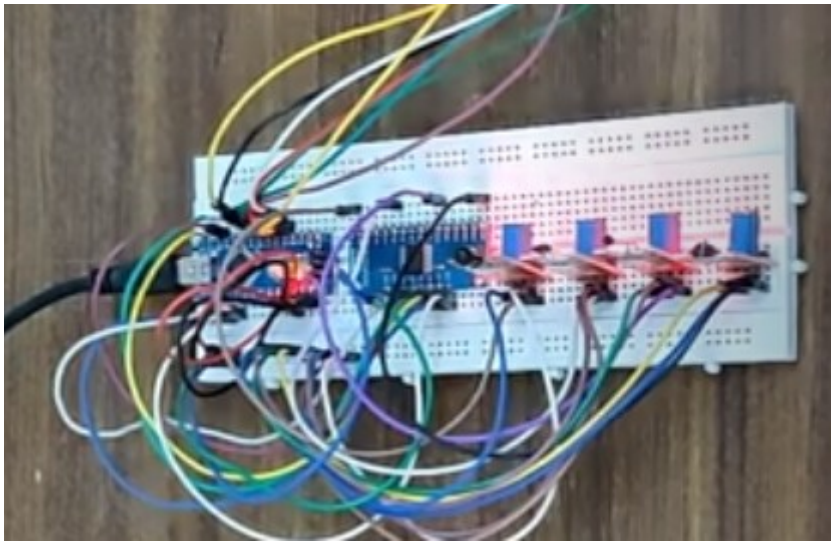
Clicca qui per andare al progetto
Clicca qui per scaricare il file "***gerber***"

Clicca qui per la pagina del ***download*** (per costruire/richiedere la basetta)

Montaggio della scheda per il Multiplexer CD7HC067

Il multiplexer può essere molto utile, perché permette si collegare 16 ingressi/uscite ad Arduino, occupando solo 6 porte del microcontroller piuttosto che sedici!

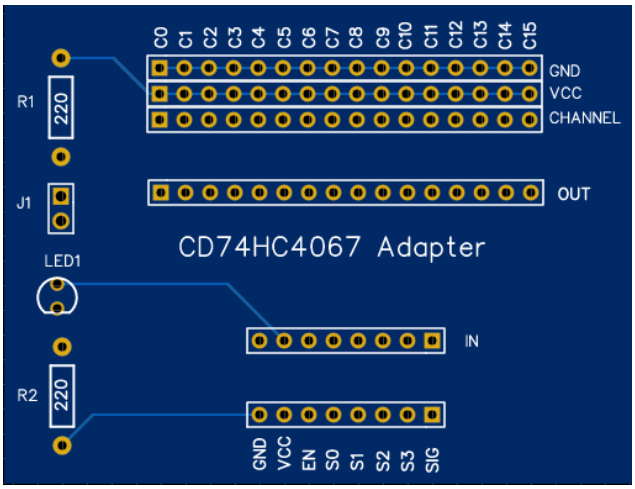
Però collegare 16 moduli in ingresso o uscita al multiplexer, oltre che i cavi necessari per abbinarlo ad Arduino, possono creare un bell'ingorgo di cavi.



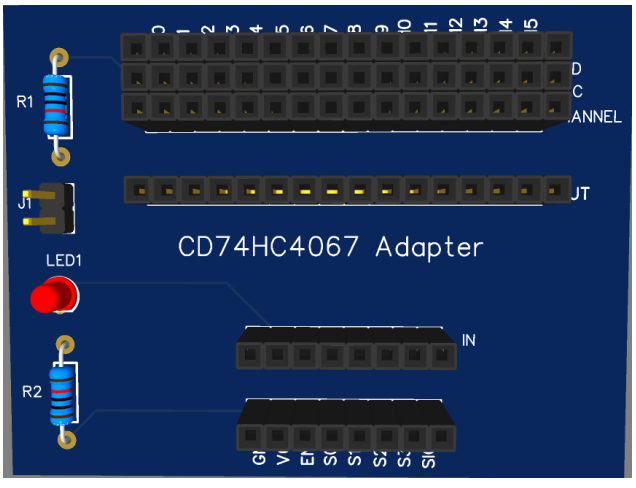
La massa di cavi necessari per collegare quattro sensori...

Nel paragrafo relativo ai programmi relativi ad esso, è stata utilizzata una basetta sperimentale, perfettamente funzionante ma la cui realizzazione non è alla portata di tutti, e così ne è stata realizzata una pi professionale, che inviando i file "gerber" a una ditta specializzata, se ne potrà ottenere a un prezzo competitivo e in pochi giorni una versione su cui è necessario montare solo una manciata di componenti.

Ecco l'immagine della basetta, all'incirca delle dimensioni reali:



La basetta da montare...



… e nella simulazione 3D

Come si vede nella parte alta, sono presenti tre file da sedici connettori ciascuna, numerati da 0 a 15. Nella fila superiore, tutti i connettori sono collegati alla massa; in quella centrale, sono collegati a 5 v; a quella inferiore corrispondono i sedici ingressi (o uscite). Si ricorda che non si possono usare insieme ingressi e uscite, ma bisognerà decidere in sede di progetto quale sarà l'utilizzo.

Si è usata questa configurazione, perché la maggior parte dei moduli di ingresso (potenziometri, trimmer, sensori di temperatura e tutti quelli che possono essere direttamente collegati agli zoccoli **DG1** e **AN1** usano questa configurazione.

Per i primi test, i moduli stessi possono essere addirittura inseriti direttamente nei connettori corrispondenti; in fase di realizzazione o quando siano molti, collegarli con cavetti a tre connettori. Anche alcuni led di test (sono riuscito a collegarne direttamente otto) possono essere inseriti direttamente sui connettori, alternandoli, usando per esempio C0, C2, ecc., collegando il cavo più lungo (anodo) al la fila di connettori inferiori, quelli contrassegnati con "Channel); mentre il cavo pi corto (catodo, alla fila superiore, contrassegnata con "GND".

Il multiplexer sarà connesso ai due connettori centrali, quello a 16 pin contrassegnato "OUT" e quello a otto contrassegnato "IN". Poiché il multiplexer ha i terminali asimmetrici (16 in uscita e 8 in ingresso), non è possibile invertire erroneamente il suo posizionamento sulla basetta.

Il connettore in basso, a otto poli, è quello che sarà collegato ad Arduino, o sullo zoccolo **DY2** o su **DG11**, in base alla configurazione prescelta.

Ecco la configurazione delle porte per il collegamento multiplexer con Arduino:

Collegamento a DG1:			**Collegamento a DY2**		
GND	→	--	GND	→	--
VCC	→	+	VCC	→	+
EN	→	D11	EN	→	D2
SO	→	D2	SO	→	D3
S1	→	D3	S1	→	D4
S2	→	D5	S2	→	D5
S3	→	D10	S3	→	D6
SIG	→	D13	SIG	→	D7

Restano da spiegare solo i componenti posti in verticale sulla sinistra della basetta: Il multiplexer non ha alcun led che ne indichi il collegamento ad Arduino. Il led "led1", alimentato attraverso la resistenza "R2" ne indica il collegamento al microcontroller.

Come spiegato all'inizio di questo paragrafo, si possono collegare sia moduli in ingresso che in uscita, alimentati a 5 volt. Ma i led, alimentati a questa tensione, bruciano rapidamente; la resistenza "R1" è stata inserita in questo progetto proprio per evitare il problema. Ma la tensione non sarà troppo bassa, se si desiderano collegare dei moduli che funzionano a 5 v? In questo caso è sufficiente cortocircuitare la resistenza, inserendo il cappuccio di collegamento sul jumper "J1"; in questo modo essi riceveranno la giusta tensione di alimentazione.

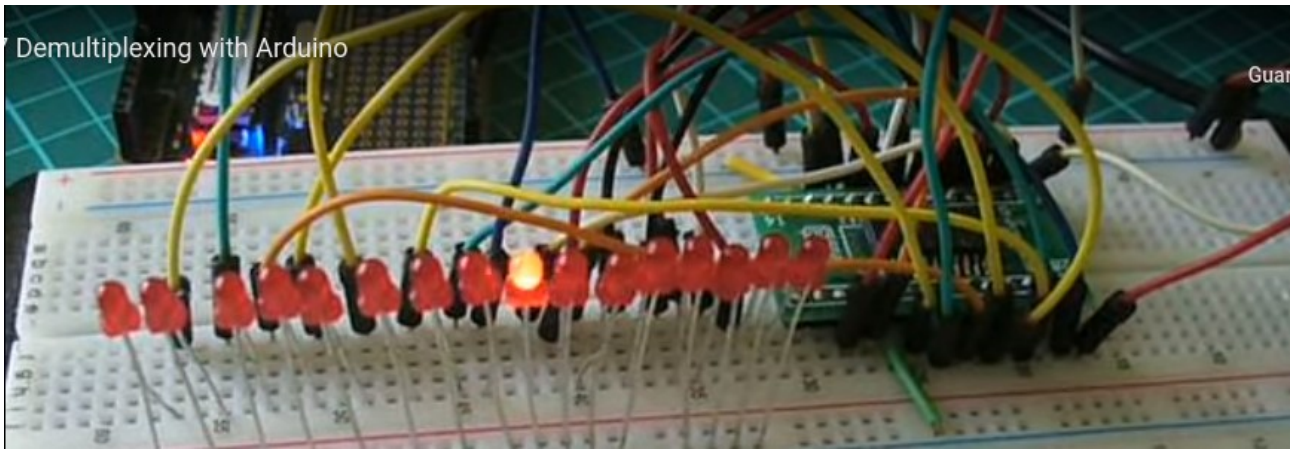
Clicca qui per andare alla pagina del progetto

Clicca qui per scaricare il file "***gerber***" (per costruire/richiedere la basetta)

Clicca qui per la pagina del ***download***

Montaggio della scheda multiled (da abbinare al multiplexer)

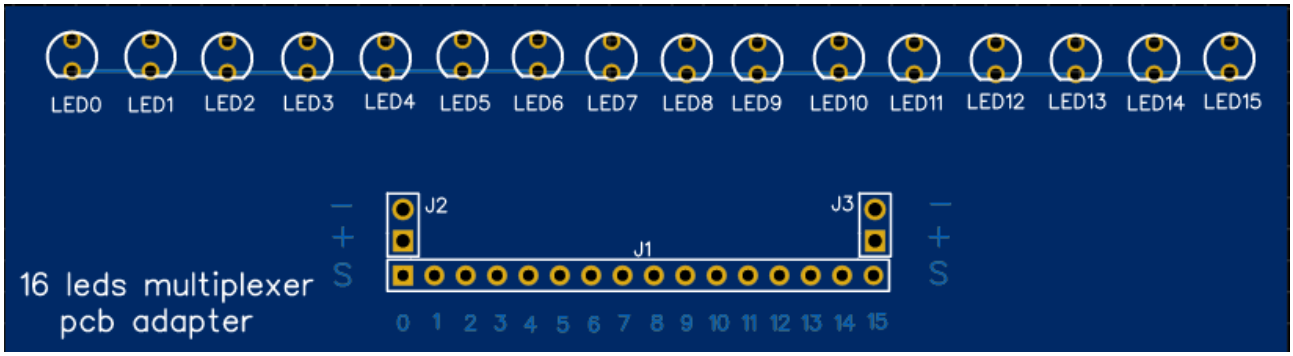
Per collegare una batteria di sedici led al multiplexer, sono necessari almeno diciassette cavetti (16 per i led, 1 per la massa), massa tale che possono creare qualche confusione.



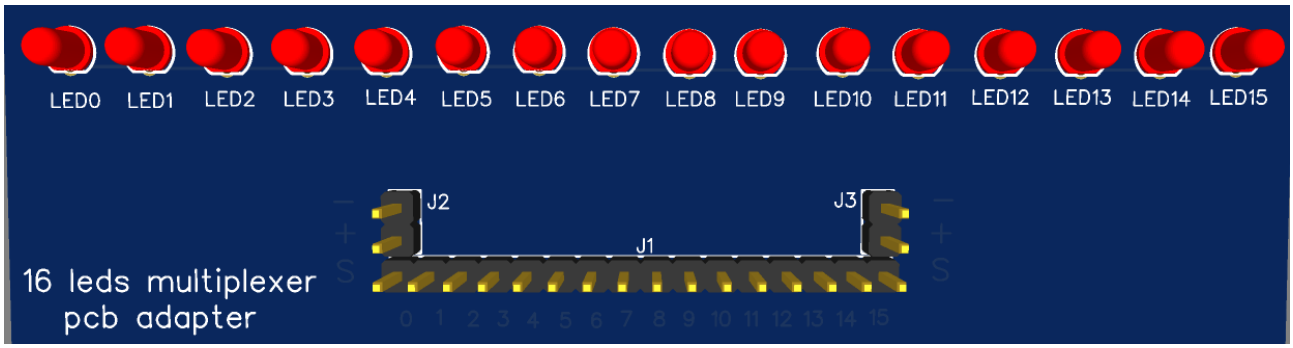
I cavi necessari per far accendere 16 led con una breadboard…

Un sistema semplice e ordinato, consiste nel costruire una basetta dedicata, utile specialmente se si desidera usare questa configurazione più volte.

Ecco l'immagine della semplice basetta creata ad hoc:



basetta "nuda"



versione 3d della basetta con 16 led, senza lacun cavo

Attenzione: nella simulazione 3d, i connettori “maschi” sono stati inseriti nella parte superiore; mentre nella realtà devono essere inseriti sulla faccia inferiore, come i piedini saldati alla basetta di Arduino Nano. In questo modo, la basetta con i led può essere inserita direttamente su quella del multiplexer; infatti la posizione dei piedini corrisponde perfettamente. Le due file superiori di piedini non è completa, in quanto i piedini dalla massa (quelli indicati con “-”), sono comuni a tutti i led; mentre la seconda, quella relativa ai +5v non verrà utilizzata ai fini dell’alimentazione.

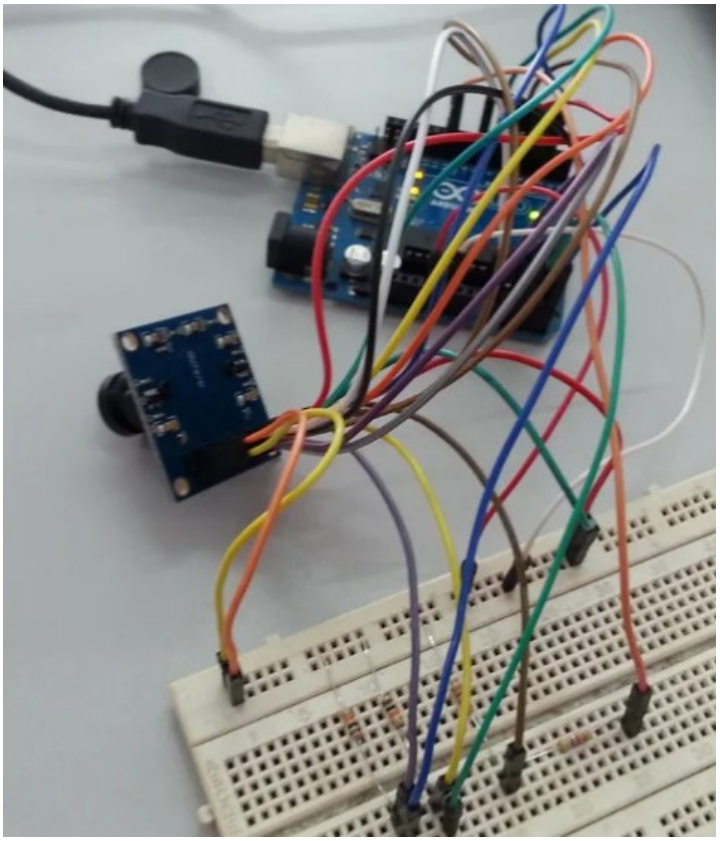
Clicca qui per andare alla pagina del progetto

Clicca qui per scaricare il file “***gerber***” (per costruire/richiedere la basetta)

Clicca qui per la pagina del ***download***

Montaggio della scheda per la camera OV7670

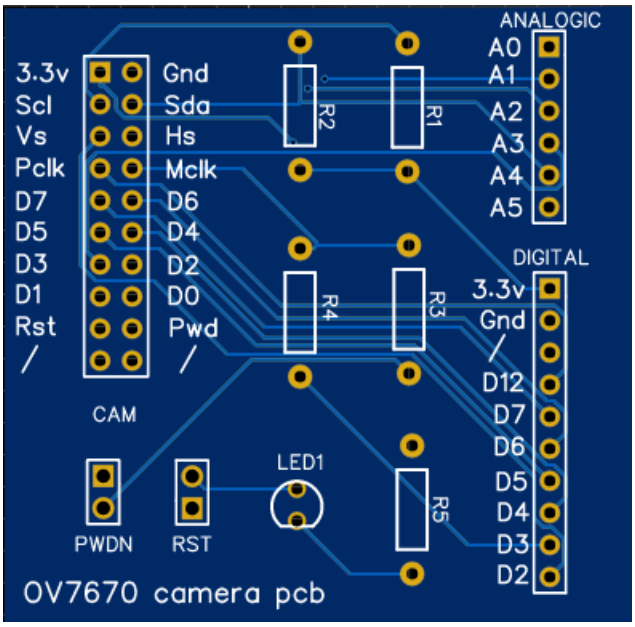
Il collegamento della Cam OV7670 ad Arduino dà molte soddisfazioni, ma non è del tutto semplice.



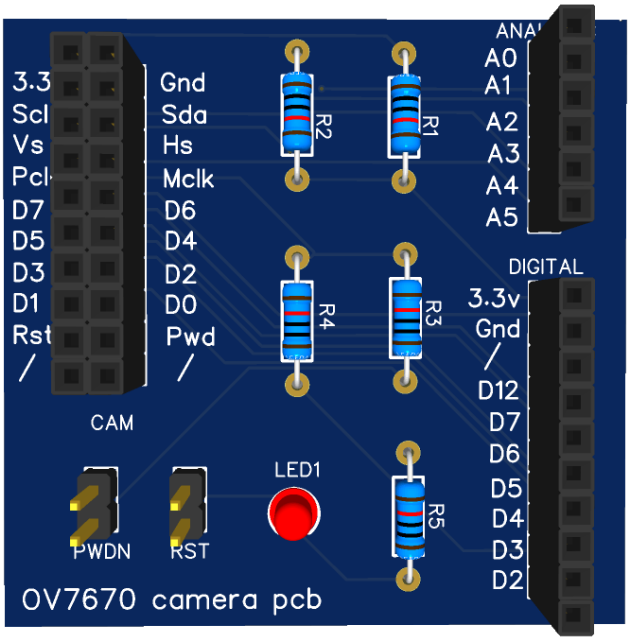
Il collegamento della Cam ad Arduino

La piccola telecamera in oggetto ha due file di piedini, alcuni dei quali andranno collegati a porte digitali, altre a quelle analogiche; inoltre, essendo alimentata a 3,33 v, alcune connessioni devono essere adattate alle giuste tensioni inserendo delle resistenze per evitare di bruciare la camera stessa.
Naturalmente è possibile strutturare il circuito su di una breadboard, ma richiede tempo e pazienza, e una volta che si sia stato disassemblato, richiederà nuovamente pazienza per essere riconfigurato.
Con una basetta dedicata invece, il collegamento della cam ad Arduino è un'attività di pochi secondi.

Ecco le immagini della basetta ideata:



La basetta nuda...



… e nella simulazione 3D

Come si può vedere, sulla sinistra appare la doppia fila di connettori su cui collegare la OV7670. Nella parte centrale, sono inserite le quattro resistenze necessarie per il collegamento della camera ad Arduino, ed in basso un led e la sua resistenza per adeguare l'assorbimento, utili per indicare che la scheda è collegata ed attiva. Sull'estrema destra sono presenti i due connettori di uscita, uno per le porte digitali e una per quelle analogiche. Perciò con due ordinati connettori a più cavetti potremo collegare la camera ad Arduino.
Sulla sinistra, sotto i connettori per la telecamera, sono presenti due jumper, uno per dare/togliere corrente all'apparecchio e il secondo per resettarla. Per il normale funzionamento devono essere entrambi ponticellati.

Ecco il valore delle resistenze necessarie:

- R1, R2: 10 KOhm, ¼ W;
- R3: 1 KOhm, ¼ W;
- R4: 680 Ohm, ¼ W;
- R5: 100 Ohm, ¼ W.

Clicca qui per andare alla pagina del progetto

Clicca qui per scaricare il file "***gerber***" (per costruire/richiedere la basetta)

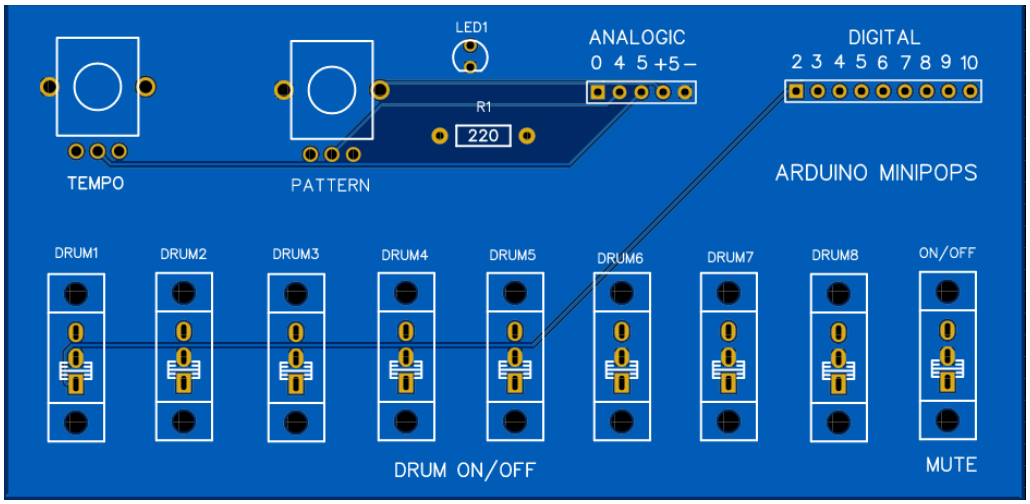
Clicca qui per la pagina del ***download***

Montaggio della scheda per Minipops (drum machine)

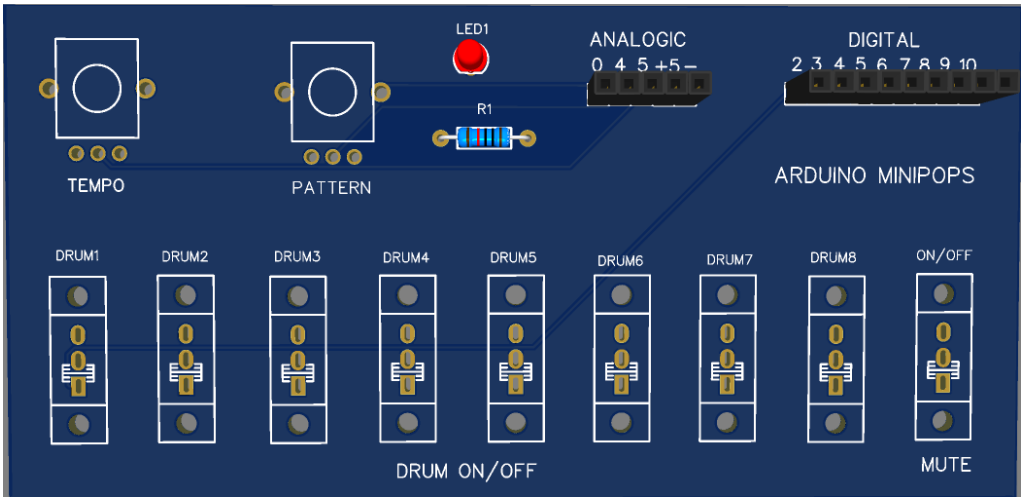
Il programma Minipops è un eccellente esempio di come un progetto elettronico realizzato negli anni ‘70 - ‘80 del secolo scorso, complesso e costoso, possa essere realizzato con una spesa irrilevante con Arduino.

Probabilmente nessuno al giorno d’oggi userebbe questa drum machine per un brano musicale, e per testarla è sufficiente inserire un paio di potenziometri sulla bs. Ma se fossimo appassionati di strumenti musicali vintage, questo progetto dal costo di pochi euro potrebbe fare al caso nostro.
Il Miniposp originale montava due potenziometri, uno per i pattern e uno per il tempo, otto switch per attivare/disattivare i vari strumenti del drum set, un nono interruttore per il mute dello strumento e un terzo potenziometro per il volume, che noi in questo progetto non implementiamo.

Ecco le immagini della basetta per il Minipops:



La basetta nuda…



… e nella versione 3D. Purtroppo Easy Eda nel suo database non possiede le immagini dei potenziometri e degli switch usati.

Il montaggio della scheda

In alto sono posti i due potenziometri che sono il cuore del progetto, infatti sono relativi al tempo (battiti al secondo) e il pattern sonoro (rumba, cha cha cha, bossa nova, ecc). Di fianco si trova un led che indica semplicemente lalimentazione della scheda, associato alla sua resistenza. Infine si trovano i connettori per i collegamenti sia analogici che digitali con Arduino. Le porte di riferimento sono indicate dalla serigrafie.

Nella fila inferiore sono stati inseriti gli otto switch che attivano/disattivano i vari componenti del drum set e il nono serve per attivare/disattivare il sequencer.

La lista dei componenti:

Name	Designator	Footprint	Quantity	Manufacturer Part
HDR-F-2.54_1x9	DIGITAL	HDR-F-2.54_1X9	1	HDR-F-2.54_1x9
HDR-F-2.54_1x5	ANALOGIG	HDR-F-2.54_1X5	1	HDR-F-2.54_1x5
LED-TH-3mm_R	LED1	LED-TH_BD3.0_RED	1	204-10SDRD/S530-A3-L
RK097N	Potentiometer	RK097N	2	RK097N
220 Ohm ¼ W	R1	R_AXIAL-0.4	1	
HX SS12F15G5	DRUM1/./DRUM8,ON/OFF	SW-TH_HX_SS12F15G5	9	HX SS12F15G5

Uno degli switch HX ss12F15G5

I potenziometri RN09N

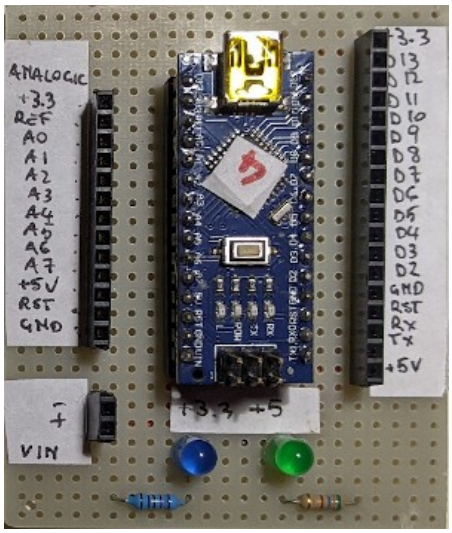
Clicca qui per andare alla pagina del progetto

Clicca qui per scaricare il file "***gerber***" (per costruire/richiedere la basetta)

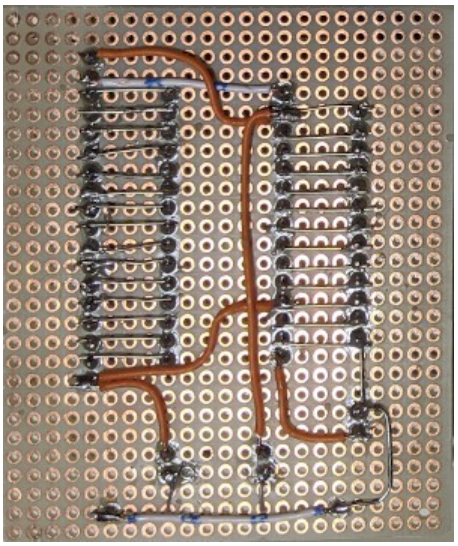
Clicca qui per la pagina del ***download***

Una scheda veramente di base per Arduino Nano

C'è già il progetto per una basetta sperimentale per Arduino Nano. E' necessario realizzarne un'altra? In effetti non è indispensabile, ma può essere utile…
Questa idea di una scheda veramente basica, mi è venuta quando ho avuto la necessità di un collegamento volante con Arduino Nano, che a differenza di Arduino Uno, ha tutti i connettori maschio, e non sempre si hanno i cavi con i capicorda corretti, e quindi si rischiano contatti, o peggio cortocircuiti. Ne ho viste offerte dai "soliti" siti in Internet, però invece di fornire delle connessioni femmina, usano dei connettori a vite, adatti più per scopi operativi che di test. Così ho pensato di realizzarne una come piace a me.
Naturalmente la prima versione come prototipo, è stata realizzata con una basetta millefori.

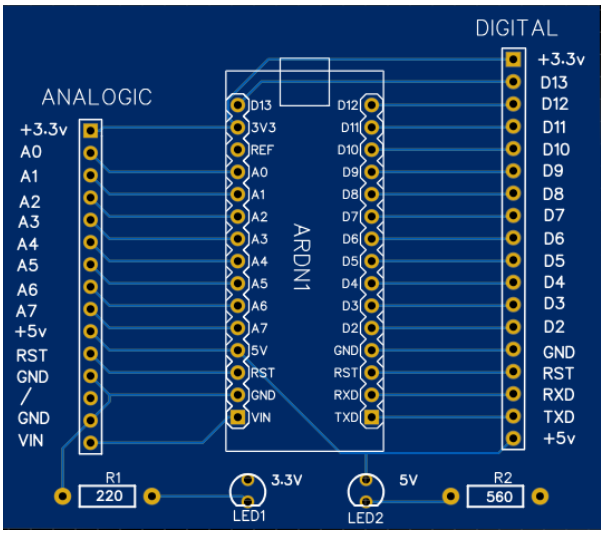


Vista frontale

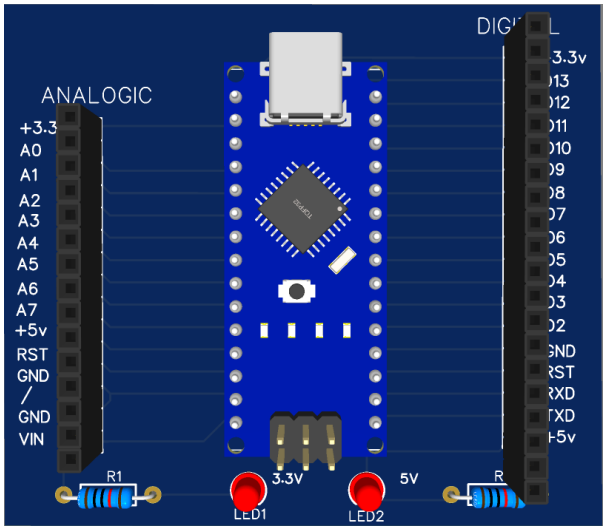


Vista posteriore

Ed eccola invece nella versione definitiva:



Vista 2D



Vista 3D

Come si vede, il progetto è estremamente semplice: al centro c'è lo zoccolo a trenta piedini per Arduino Nano, sulla sinistra tutte le porte analogiche, comprese le due alimentazioni e la massa; sulla destra sono presenti tutte le porte digitali (compresa D13, che sul microcontroller è a sinistra, e compaiono anche le due alimentazioni, a 3,3v e a 5v, oltre che alla massa. Si è preferito porre le alimentazioni su entrambi i lati, sia per ridondanza che per praticità.
Un altro piccolo tocco personale: due led, alimentati attraverso alle solite resistenze che limitano ad essi la corrente per evitare di bruciarli rapidamente. Led 1 indica la presenza della tensione di 3,3v; Led2 quella di 5v.

Non sarebbe stato sufficiente un singolo led, a indicare il funzionamento? Ho pensato che potrebbe essere utile specialmente per Arduino Nano 33 Iot e BLE. Infatti le nuove versioni di Nano forniscono nativamente solo 3,3v, e solo con un artifizio erogano anche i 5v. Con i due led siamo in grado di stabilire immediatamente come siano stati settati.

Lista dei componenti:

- n.1 zoccolo per integrato da 30 pin;
- n. 1 connettore femmina HDR-F-2.54_1x15
- n. 1 connettore femmina HDR-F-2.54_1x18
- n. 2 led
- n. 1 resistenza 220 Ohm, ¼ W (R1)
- n. 1 resistenza 560 Ohm, ¼ W (R2)

Clicca qui per scaricare i file gerber

Clicca qui per lo schema elettrico

I componenti per montare la pesa con Arduino

La precisa pesa che possiamo realizzare con la cella di carico e l'amplificatore di segnale HX711, non può essere realmente usata se non ha una base e una vaschetta su cui porre gli oggetti da pesare. Perciò abbiamo realizzato la base, il sostegno per la vaschetta e la vaschetta con una stampante 3D.

La cella di carico, le quattro viti, gli otto dadi, la base, il supporto e la vaschetta, costruiti con una stampante 3D

La pesa montata. La distanza tra la base e la cella di carico e tra quest'ultima e il supporto della vaschetta è regolata in base a quanto profondamente sono inserite le viti. I controdadi servono a evitare che si cambino i rapporti tra le distanze.

Il montaggio di per sé è molto facile e intuitivo.

Clicca qui per visualizzare i programmi relativi alla pesa.

Clicca qui per i file per realizzare i tre componenti con la stampante 3D