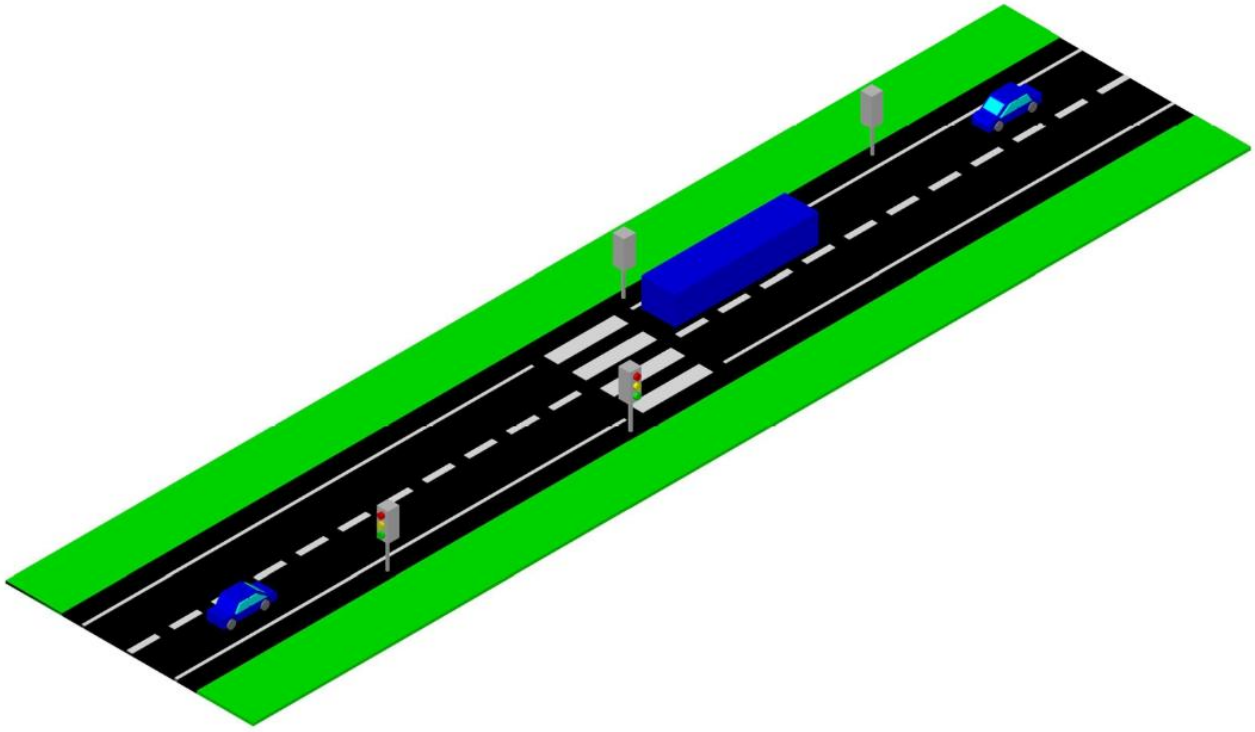
2018

Semaforo Book



sergio

Associazione ARCO

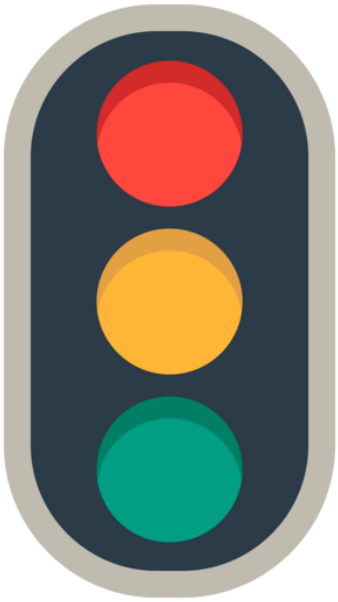
28/08/2018

Indice

Indice 2
il Progetto 4
lo Schema a Blocchi 5
il Master 7
lo Schema Elettrico 8
vers. 2 10
vers 3 11
vers. 4 11
vers. 5 12
il PCB 13
vers. 5 13
lo Slave (gestione semaforo) 14
lo Schema Elettrico 15
vers. 2.3 - FidoCAD 15
vers. 2 - EAGLE 16
vers. 3 - EAGLE 16
vers. 4 - EAGLE 17
il PCB 18
Versione 2.3 by FidoCAD 18
lo Slave (rivelatore velocità) 19
Il modulo 19
Schema elettrico 19
lo Schema Elettrico 23
Schema 1 23
schema 2 - SMD 23
il PCB 24
vers. 1 24
vers. 2 25
il diorama 26
i contatti del connettore semafori 26
il semaforo "mega" 27
la realizzazione pratica 27
il connettore 27
la tecnologia e i componenti 28
Arduino 28
4n35 28

Protocollo I2C 29
listati sperimentali 30
i File disponibili 34
i Link utili 34
i software utilizzati 34

il Progetto

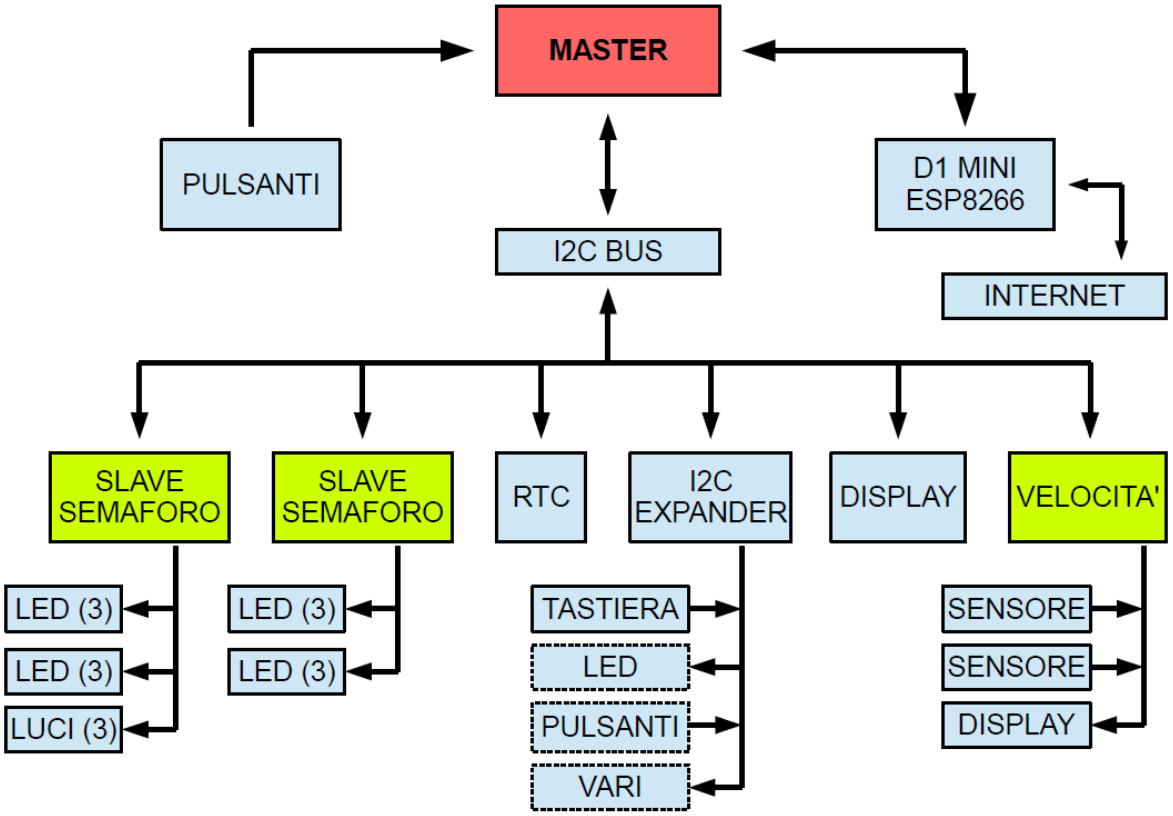


Con la serata di giovedì 11 gennaio è partita la progettazione del circuito di controllo del "Semaforo". Abbiamo convenuto che si tratta di un progetto didattico, non immaginiamo infatti che vi possano essere applicazioni effettive "su strada" della realizzazione. Sarà invece l'occasione per mettere a frutto la consolidata manualità dei nostri makers e per sperimentare la programmazione di Arduino e l'utilizzo di parecchi accessori (Shield) e delle relative librerie.
In sintesi ecco quanto è stato proposto per la sperimentazione e realizzazione:

- costruzione di un semaforo a tre luci in dimensioni reali (già in avanzata lavorazione presso la "tana")
- realizzazione di un sistema a due corpi semaforici per controllo, ad esempio, di un senso unico alternato (il secondo corpo luci sarà solo simulato con basetta led o realizzato in miniatura) che prevederà:
  - due sistemi di controllo (uno master ed uno slave)
  - scambio dati tra i due sistemi via cavo e via radio
  - check del funzionamento delle lampade all'avvio
  - possibilità di controllo manuale del ciclo
  - fascia oraria di funzionamento
  - chiamata pedonale per interruzione del ciclo
  - regolazione manuale della velocità del ciclo
  - rivelatore di prossimità per auto regolazione dei tempi di "open"
  - rivelazione di velocità elevata in avvicinamento

...
erogazione del caffè per gli sperimentatori !!!!

lo Schema a Blocchi



Inizialmente il progetto “Semaforo” ipotizzava la comunicazione fra i vari moduli, che lo avrebbero composto, sia mediante una connessione cablata (via cavo) e sia mediante una connessione wireless ovvero via “radio” (o WiFi), dopo una attenta analisi si è ritenuto, al momento, di utilizzare solo una connessione cablata (via cavo a mezzo del protocollo I2C) e di relegare la comunicazione wireless alla parte d'interfacciamento con una, eventuale, rete di computer per la gestione del progetto anche in remoto (internet).

Alla stesura del presente documento il progetto prevede i seguenti componenti:

- Un master, realizzato con un arduino e relativa scheda, che svolge la funzione di “controllore” dell'intero processo, ad esso è delegata tutta la parte gestionale dell'intero progetto. E', di fatto, il “direttore dell'orchestra”.
- Un modulo “D1 Mini” che utilizza un ESP8266 per l'interfacciamento WiFi.
- Due slave, realizzati con due arduini e relative schede, che hanno il compito di gestire, materialmente, i vari semafori (in bassa tensione mediante led) sia quelli veicolari che quelli pedonali, inoltre, sono predisposti per comandare una batteria di relè a cui può far capo un semaforo “vero”.
- Un controllore della velocità rilevata, realizzato sempre con un arduino e relativa scheda, che ha la funzione di rilevare la velocità mediante due sensori ad effetto hall (nel caso specifico si tratta di due sensori che rilevano la presenza di un campo magnetico che, sempre in questo caso, si tratta di un magnete posto sotto una “macchinina”).
- Un display lcd gestito dal “master” per interfacciarsi al mondo esterno.
- Una tastiera a matrice per poter inserire dei dati.
- Una serie di pulsanti per richiamare determinate funzionalità definite in fase di programmazione, uno di questi pulsanti è, per esempio, quello per la chiamata pedonale (il progetto ne prevede uno solo ma nulla vieta di utilizzarne più di uno collegati in parallelo).

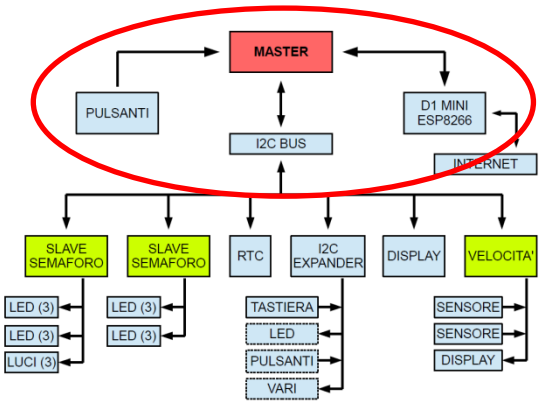
- Un modulo RTC per poter disporre di un datario ed un orologio, reali ed indipendenti dal sistema ovvero che continuano a funzionare anche se il tutto è spento.
- Un I2C espander per poter aggiungere, nel tempo, eventuali periferiche come led o pulsanti.

Come indicato in apertura l'intero sistema prevede la comunicazione fra i vari moduli (master, slave, velocità, display lcd, tastiera a matrice ect. ect.) mediante una connessione cablata che utilizza il protocollo I2C realizzando, di fatto, un bus dati I2C; questa scelta apre la possibilità, in un eventuale secondo tempo, di realizzare, ad esempio, un master con una piattaforma differente da arduino purchè abbia una "porta" I2C, una piattaforma differente potrebbe essere, per esempio, il "PI Raspberry", inoltre, il protocollo I2C avviene attraverso una trasmissione digitale, questa caratteristica potrebbe essere oggetto di uno studio per utilizzare, in sostituzione del cavo "di rame", una trasmissione mediante fibra ottica, al momento è solo ed esclusivamente una ipotesi puramente teorica.
Nello schema a blocchi è indicato quanto già sopra espresso, i sensi delle frecce indicano la direzione dell'informazione. I riquadri colorati in rosso e in giallo/verde prevedono l'uso di arduino.

il Master

Il modulo master si compone di:

- Un arduino.
- Una serie di pulsanti (5 + l'eventuale tasto di reset).
- Una “D1 Mini” della “WeMos” che utilizza un modulo ESP8266 per le comunicazioni via WiFi, il modulo monta anche un microcontrollore ed una memoria flash.
- Componentistica elettronica aggiuntiva come resistenze, led, connettori ect. ect.



per poter utilizzare in modo razionale quanto sopra si è deciso di realizzare un circuito stampato (in inglese viene chiamato PCB, printed circuit board, ovvero circuito stampato). In realtà i circuiti stampati realizzati sono più di uno questo perché, nel tempo, vi sono state evoluzioni e scelte differenti che hanno portato, appunto, a più schemi elettrici e più circuiti stampati sino a giungere a due progetti paralleli, il primo realizzato con metodi semiprofessionali, il secondo con metodi che possono essere realizzati con apparecchiature amatoriali ovvero “in casa”, ambedue i progetti svolgono le stesse funzioni, la differenza sostanziale è nelle “dimensioni”, il primo è più compatto poiché sfrutta ampiamente componentistica SMD mentre il secondo lo è di meno poiché utilizza componentistica tradizionale a foro passante più facilmente utilizzabile in ambito amatoriale. Attualmente la versione “compatta” è quella indicata come versione “4” degli schemi elettrici, la versione “amatoriale” è la versione “5”, concettualmente differiscono per la presenza di “convertitori di livello logico” direttamente sulla scheda (shield) o su schedina realizzata da terzi e per la forma di collegamento con arduino, la prima si innesta sopra ad arduino diventando di fatto una shield, la seconda utilizza dei cavetti, per comodità verrà analizzata sono una versione, la “4”, quella SMD.

lo Schema Elettrico

Arduino è una piattaforma hardware composta da una serie di schede elettroniche dotate di un microcontrollore programmabile, è una idea, e relativo sviluppo, di una società Italiana di Ivrea che ha fornito, a livello mondiale, uno strumento per la prototipazione rapida per scopi hobbistici, didattici e anche professionali, inoltre, il software a corredo è libero e l'hardware, ovvero gli schemi circuitali, sono distribuiti come "libero"; questo ha fatto sì che nel mercato si trovino arduini "originali", ovvero realizzati sotto il controllo della casa "madre", e sia arduini "cloni" che sono compatibili (non sempre, alcune volte possono avere qualche piccola differenza).

La versione di arduino che viene utilizzata è il modello "Uno" (nella release 3) ma potrebbe essere utilizzato qualsiasi modello, ovviamente la scheda nella versione compatta (shield) è specifica per quel modello (si monta fisicamente sopra) mentre la versione amatoriale si presta più facilmente ad essere utilizzata con modelli di arduino differenti o, addirittura, piattaforme differenti come, ad esempio, Pi Raspberry.

Arduino Uno dispone di 6 porte analogiche & digitali siglate come A0 ÷ A5 e 14 porte digitali siglate come D0 ÷ D13. Nel caso del modulo "master" vengono utilizzate le porte A4 e A5 (I2C) e tutte le porte digitali, le porte A0 ÷ A3 sono disponibili per utilizzi futuri, le porte A4 e A5 sono specifiche per il protocollo I2C e sono, rispettivamente, i segnali SDA ed SCL,

le porte D0 e D1 sono la porta seriale TTL (replica di quella su porta usb) e più precisamente RXD e TXD, queste porte, pur potendo essere utilizzate anche per fini differenti, in questo progetto, verranno utilizzate sempre in questa configurazione.

Le altre porte di arduino sono così utilizzate:

D2	Pulsante, funzione "programma 1", ciclo standard per il passaggio dei veicoli in modo alternato.
D3	Pulsante, funzione "programma 2", ciclo di emergenza/notturno ovvero a tutti i semafori con il solo "giallo" lampeggiante.
D4	Pulsante, chiamata "pedonale" ovvero l'attivazione della funzione specifica per gestire i semafori per il passaggio pedonale.
D5	Pulsante, funzione "Manuale/Automatica" ovvero determina se il progetto opera in base ad un programma specifico (1 o 2) oppure secondo l'orologio interno che determina quale programma utilizzare.
D6	Pulsante, funzione "Menù" per poter variare i parametri di programmazione come ad esempio i tempi dei vari cicli, quando ci si "trova" in modalità "menù" i pulsanti sopra potrebbero avere funzioni differenti.
D7 ÷ D12	Interfacciamento al modulo "D1 Mini"
D13	Segnale di Anomalia, qualsiasi circostanza che venga identificata come una anomalia (ad esempio l'assenza di comunicazione con gli slave) verrà segnalata con un led e portando questa porta a livello logico alto, in un secondo momento questo segnale potrebbe essere utilizzato da un modulo (attualmente opzionale) per la comunicazione mediante SMS su rete GSM.

L'alimentazione della scheda avviene attraverso il connettore "ALIM" a cui è presente nel pin 1 la tensione di +12V (può variare da +7V sino a +12V) che si collega, mediante un diodo switch (D1) per impedire inversioni di polarità, al pin "Vin" di arduino, questo pin fa capo all'alimentatore stabilizzato montato a bordo che fornisce i +5V per arduino e sul pin "+5V", il massimo carico che l'alimentatore stabilizzato è in grado di gestire è di circa 800 mA; sul pin 2 del connettore "ALIM" è presente la "massa" (GND) e sul pin 3, ponendolo a massa, si avrà il reset hardware sia della scheda arduino che il modulo "D1 Mini".

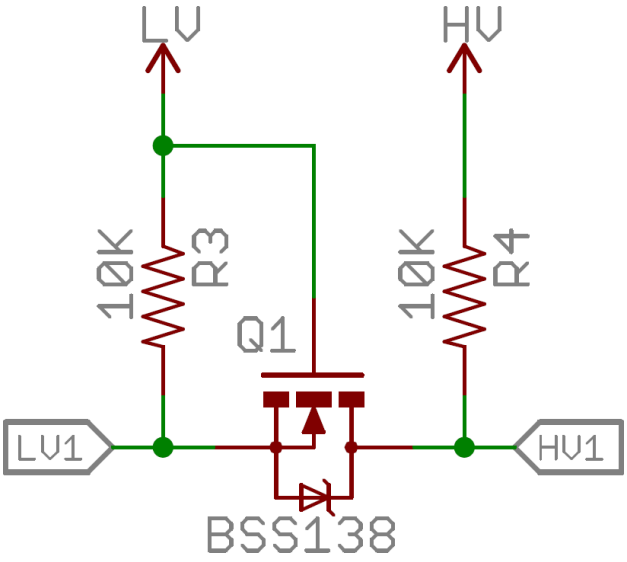
Sia per i pulsanti, che per le porte A0 ÷ A3, sono presenti delle resistenze di “pull down” da 10k (R25 ÷ R29 e R32 ÷ R35) queste garantiscono un segnale del contatto più pulito e meno sensibile ad eventuali contatti di “rimbalzo”, nel caso in cui le porte A0 ÷ A3 debbano essere utilizzate come “output” è opportuno dissaldare, o semplicemente non saldare, le rispettive resistenze di “pull down” (nelle altre varianti, a livello di c.s., questa opzione era ponticellata ovvero poteva avvenire semplicemente spostando un jumper).

Il bus dati I2C, che fa capo alle porte A4 e A5, prevedono la presenza di una coppia di resistenze di “pull up”, R30 e R31, il valore di queste resistenze può variare da un minimo di 1.5k a 4.7k in funzione della lunghezza del cavo di collegamento (più è lungo e più la resistenza dev'essere bassa), in questo progetto si è scelto per una coppia di resistenze da 1.5k. Le resistenze di “pull up” devono essere presenti solo su una scheda.

Il modulo “D1 Mini” della “WeMos” richiede una alimentazione di +3.3V che può avvenire o alimentando direttamente il modulo con una tensione di +3.3V sull'apposito pin oppure fornendo +5V, sempre sull'apposito pin del modulo, che verrà poi convertito nei +3.3V necessari, anche se non rientra fra le necessità di questo progetto il modulo può essere alimentato come arduino ovvero attraverso la porta mini usb presente.

In questo progetto l'alimentazione del modulo “D1 Mini” avviene portando i +5V, sull'apposito pin del modulo, provenienti da arduino; la linea di alimentazione a +3.3V è necessaria non solo per il modulo “D1 Mini”, che provvede autonomamente, ma anche per i “convertitori di livello logico”, nella versione “compatta” questa linea si ottiene partendo dal pin “3.3” sul modulo “D1 Mini”, nella versione “amatoriale” è opportuno utilizzare, invece, quanto fornito dalla scheda di arduino questo perchè, in assenza del modulo “D1 Mini”, se la linea a +3.3 deriva da “D1 Mini” i convertitori di livello logici si porterebbero a livello “alto” mentre se la linea a +3.3 deriva da arduino i convertitori si porterebbero a livello “basso”.

Il modulo “D1 Mini” opera, come sopra indicato, a +3.3V, arduino opera a +5V, per poter far coesistere i due sistemi, ovvero farli “colloquiare”, senza che uno dei due si “bruci” (in questo caso il modulo WiFi) si utilizzano dei “convertitori di livello logici” che possono essere già montati su una schedina commerciale o realizzati con un semplice mosfet (BSS138, Q1 ÷ Q8) e due resistenze da 10k per ogni canale (R1 ÷ R16), questa conversione avviene in modo bidirezionale ovvero alla variazione di un “lato” corrisponde automaticamente alla variazione dell'altro lato.



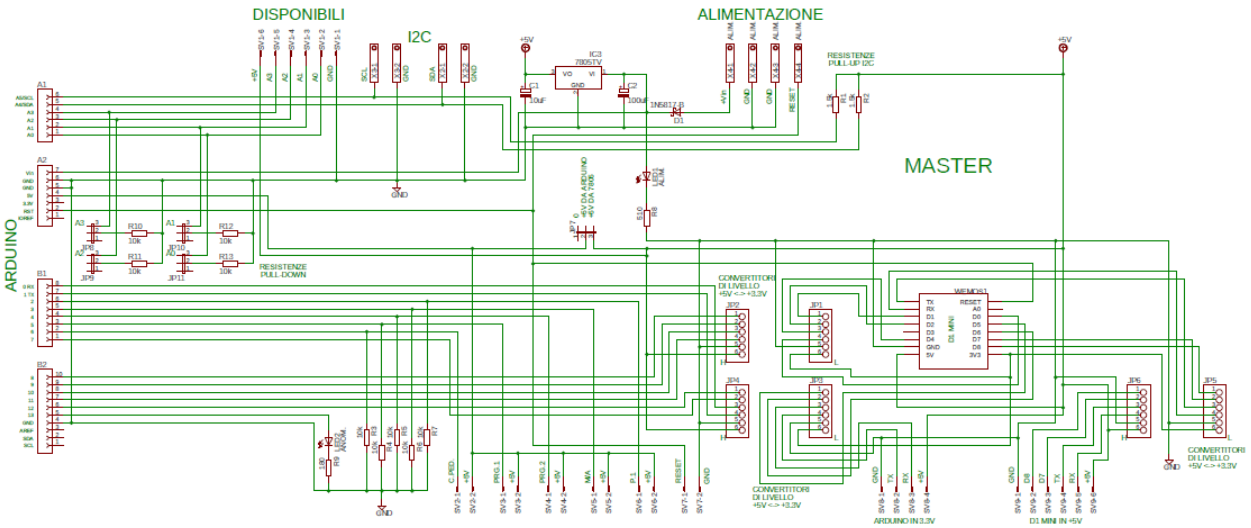
LV	Pin di alimentazione a linea di “tensione” bassa, +3.3V
HV	Pin di alimentaziona a linea di “tensione” alta, +5V
LV1	Ingresso/Uscita verso porta del modulo “D1 Mini”
HV1	Ingresso/Uscita verso porta di arduino

Oltre a interfacciare arduino con il modulo “D1 Mini”, che necessita della conversione dei livelli di logici da +3.3 a +5 (e viceversa), si prevede anche il collegamento, mediante la porta seriale TTL (D0 e D1), ad un display/monitor gestito da un “Pi Raspberry” che prevede, anch'esso, una tensione di +3.3V, per tanto, sono presenti anche due convertitori per il “Pi Raspberry” che sono replicati nel connettore “TX/RX”.

Il LED1 segnala la condizione di anomalia, la resistenza R37 è stata calcolata per avere circa 20mA, il LED2 indica la presenza di tensione, analogo dimensionamento è avvenuto per la resistenza R36.

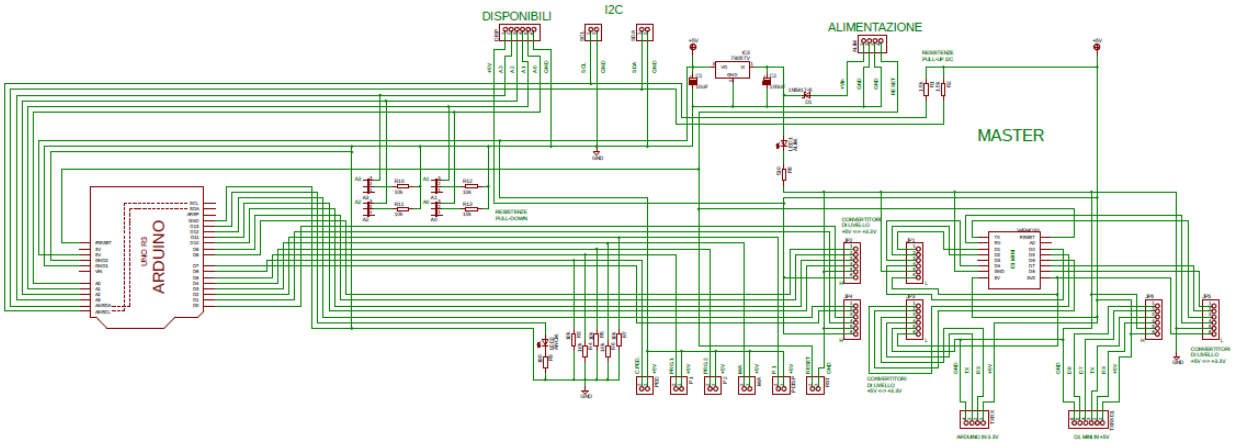
vers. 2

by zg



vers 3

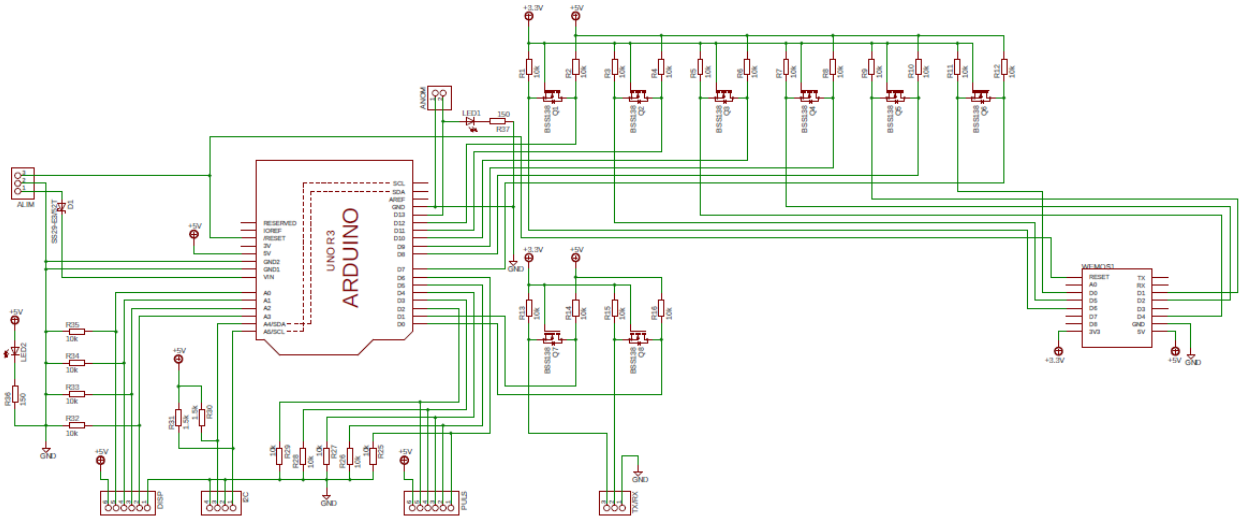
by zg



vers. 4

SMD

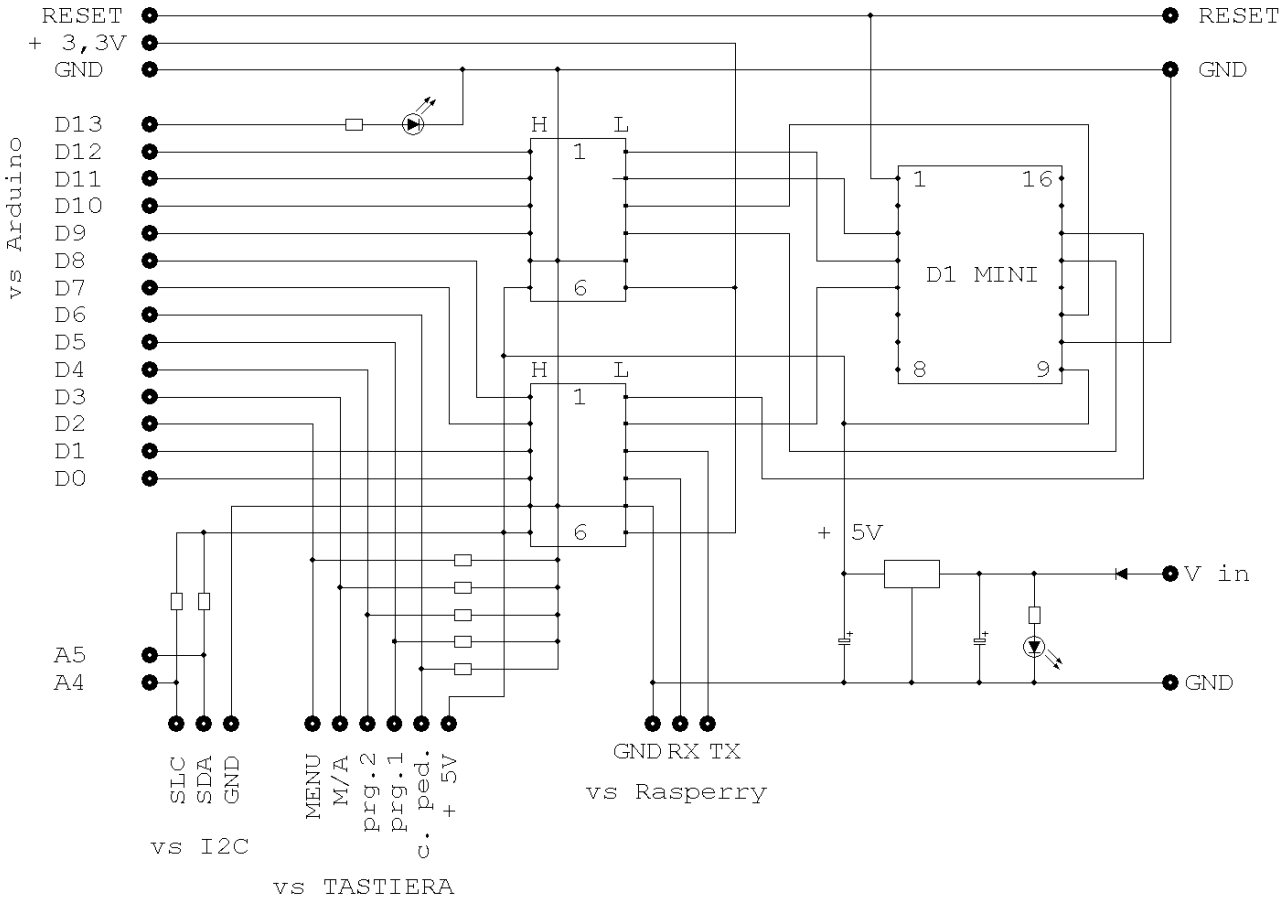
by zg



vers. 5

by gs

Master 1.1

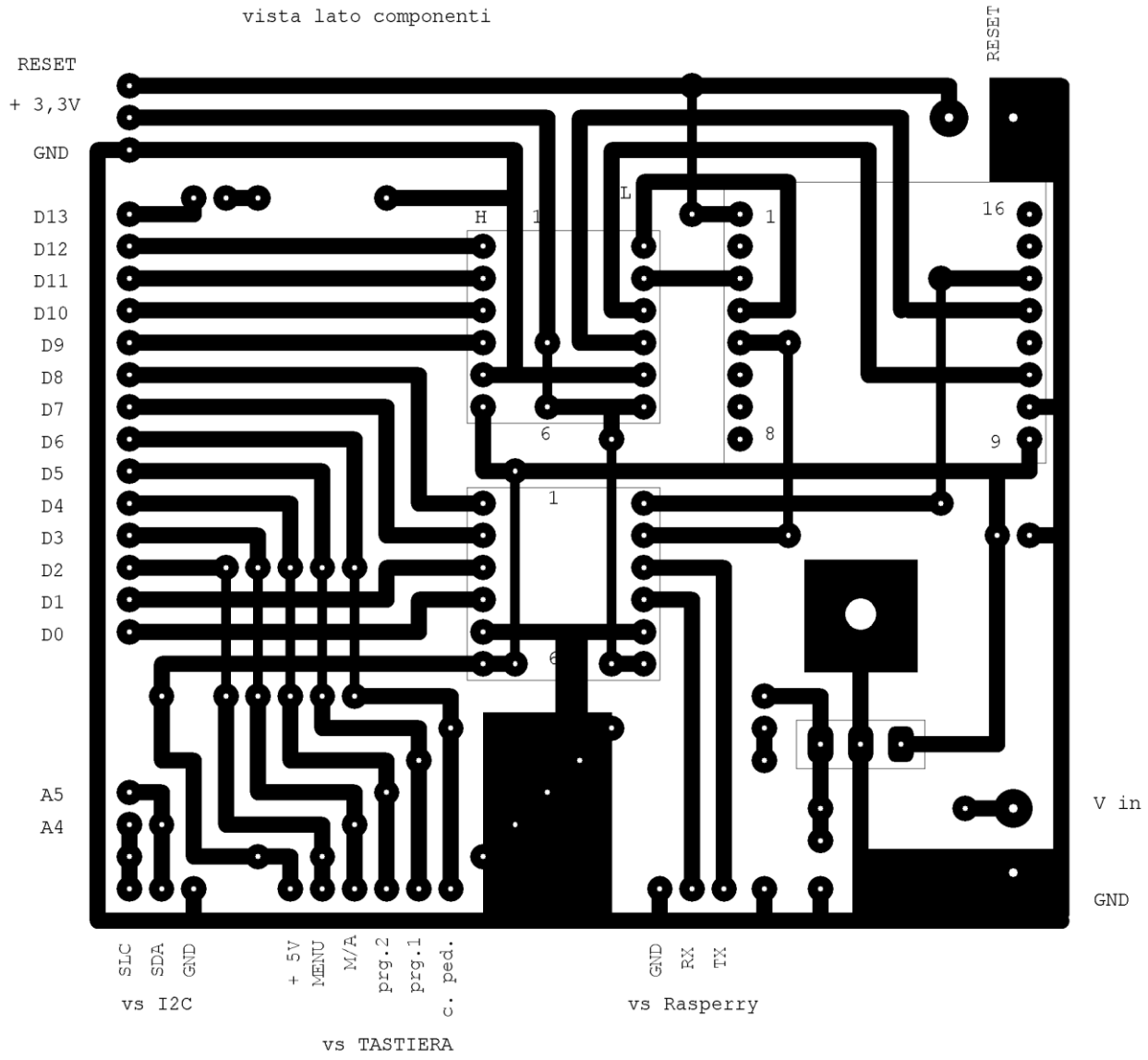


il PCB

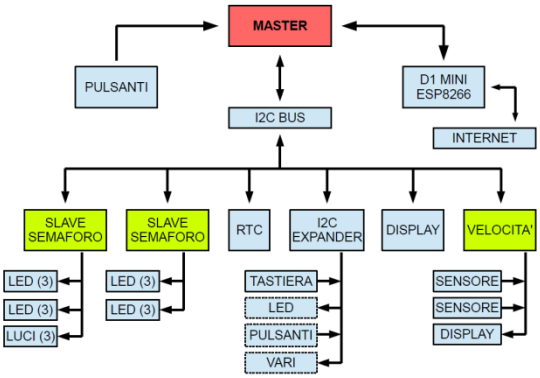
vers. 5

by gs

fidocad

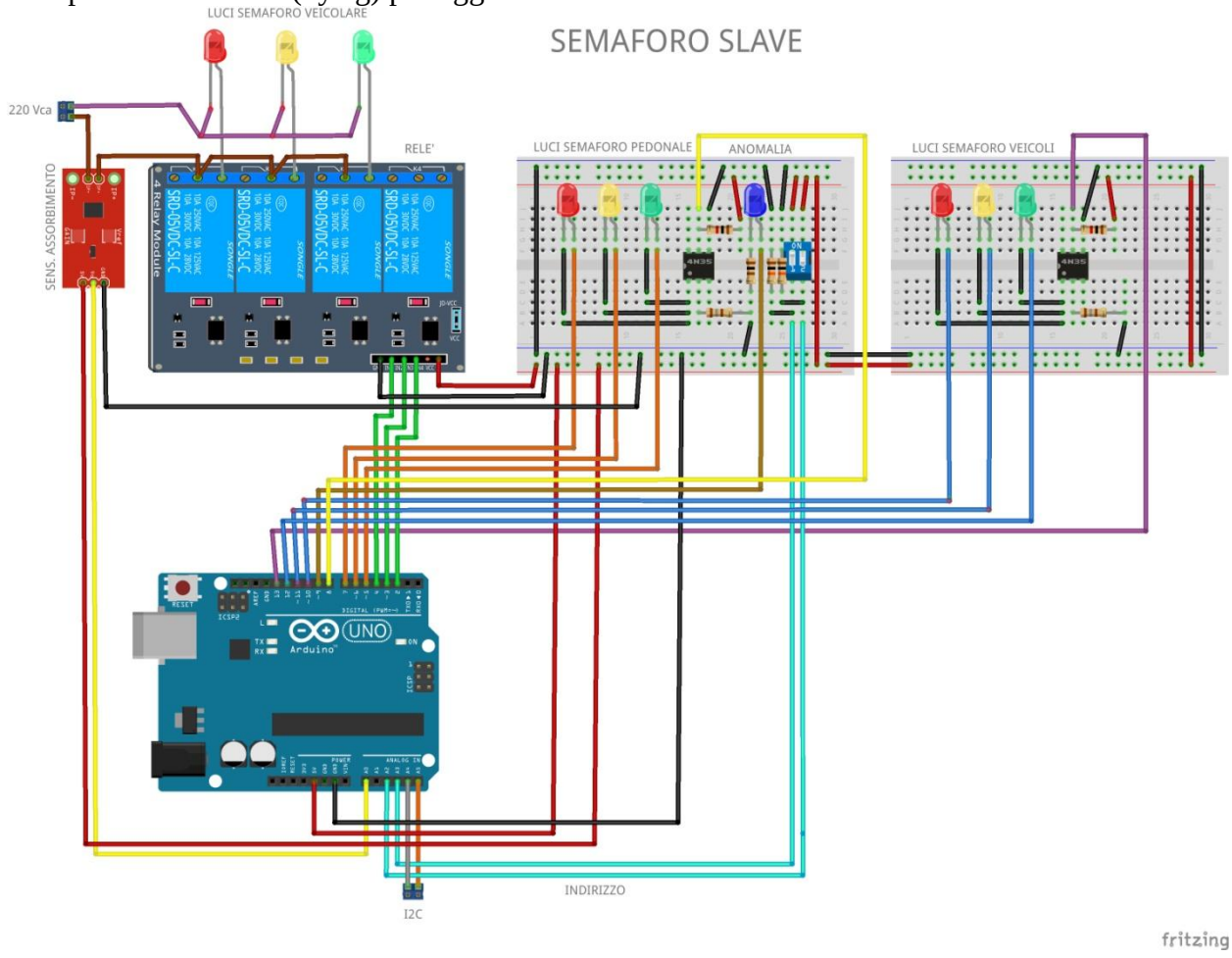


lo Slave (gestione semaforo)



Il modulo slave è il modulo che materialmente gestisce i due semafori (pedonale e veicolare) ed, eventualmente, in remoto un sistema semaforico alimentato a 220V (o ad un altro voltaggio).
A titolo puramente di verifica della fattibilità è stato in un primo momento sviluppato, la coppia di slave, mediante l'utilizzo di una bredboard, un arduino, un modulo relè, un sensore di misurazione dell'assorbimento della corrente di un carico e della componentistica varia quali led, resistenze e fotoaccopiatori. Analogamente al modulo master, anche per i due slave, abbiamo una evoluzione che utilizza componenti discreti, ed integrati, con montaggio su foro passante sino alla realizzazione di una "shield" per arduino con componentistica a montaggio superficiale (SMD), come per il modulo master si è sviluppato un ulteriore circuito, avente comunque le stesse funzionalità, che possa essere realizzato in ambiente amatoriale con la tecnica della foto incisione o della fresatura. A differenza del modulo master, nello slave, sono presenti dei jumper per il settaggio dell'indirizzo dello stesso (a livello circuitale è possibile avere sino a 4 moduli) e per l'abilitazione, o la disabilitazione, della verifica del funzionamento dei semafori, pedonale e veicolare, a bassa tensione ovvero se effettivamente si "accendono" o se sono guasti.

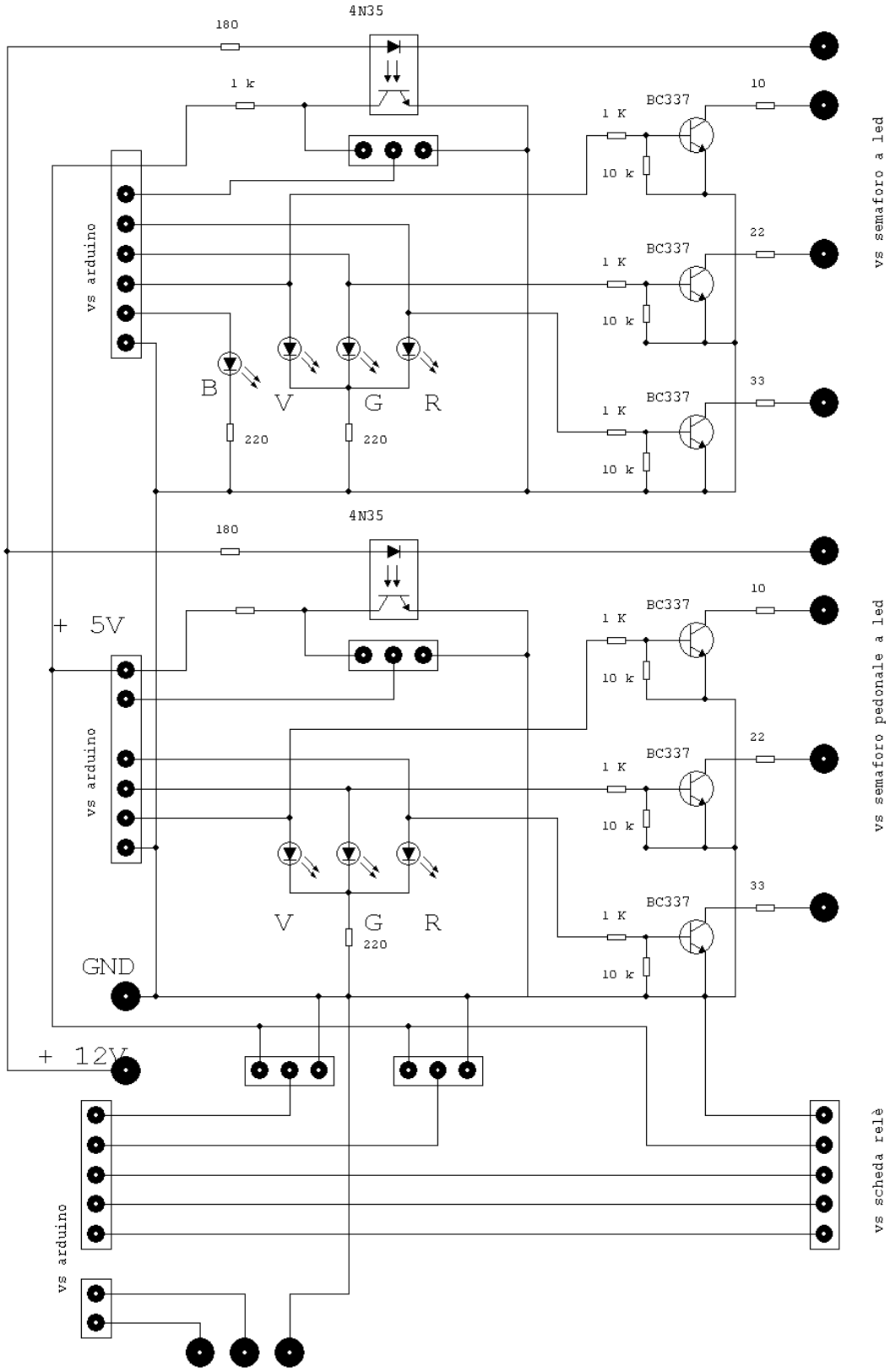
Prototipo su breadboard (by zg) poi oggetto di verifiche /modifiche



lo Schema Elettrico

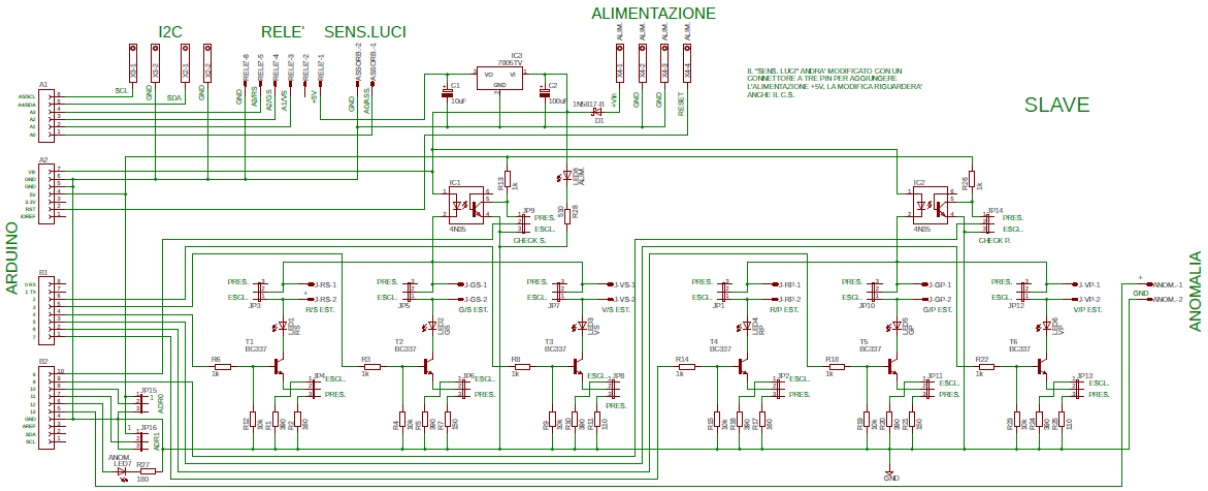
vers. 2.3 - FidoCAD

by gs



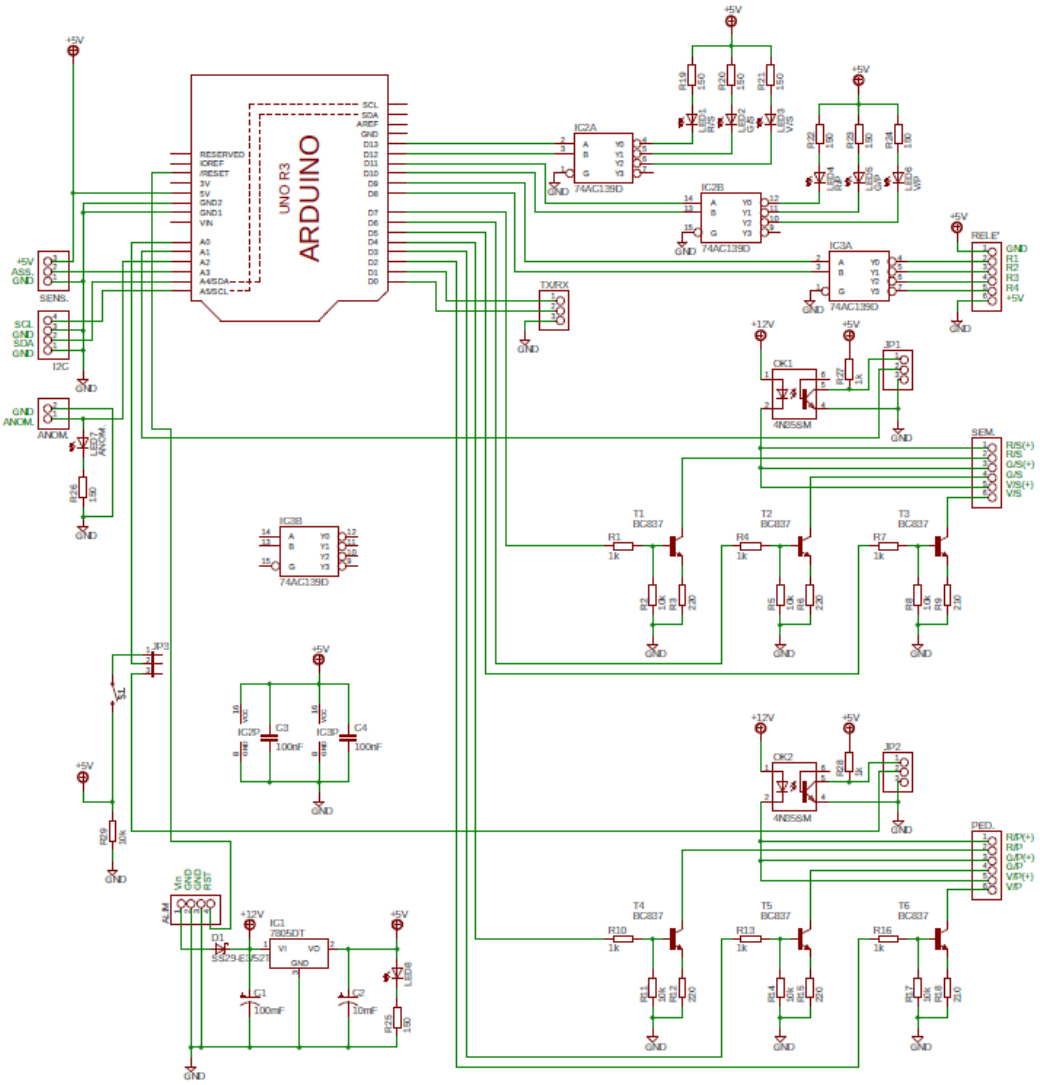
vers. 2 - EAGLE

by zg



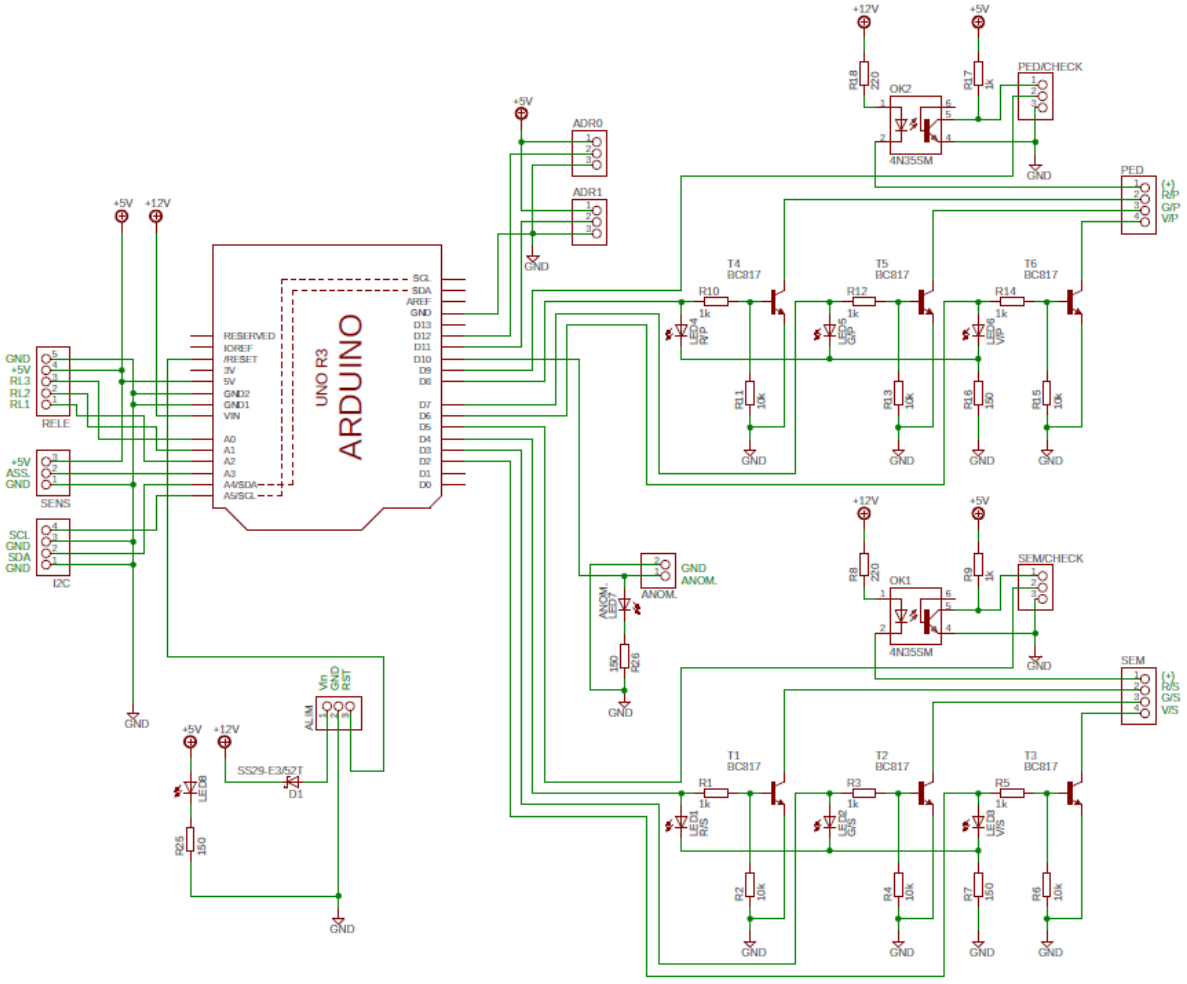
vers. 3 - EAGLE

by zg



vers. 4 - EAGLE

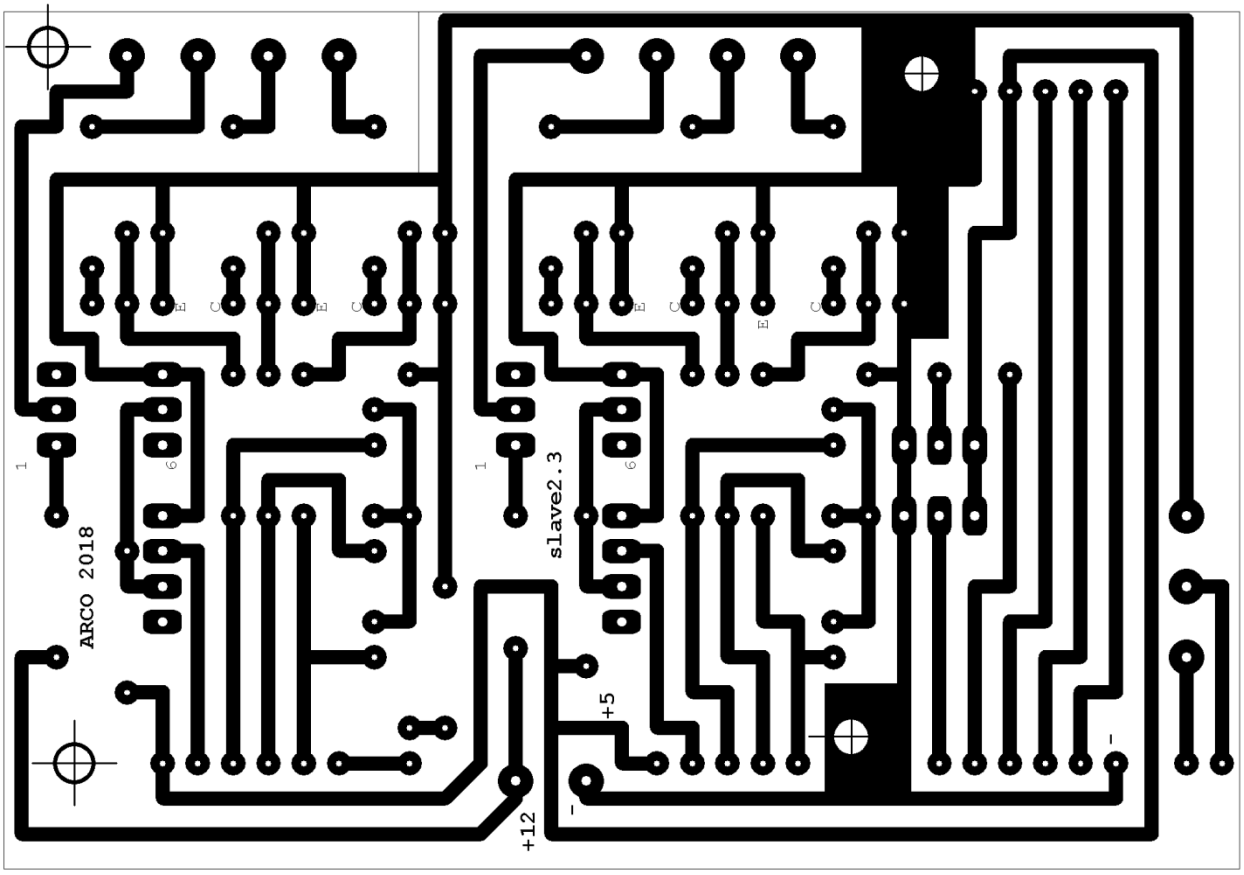
by zg



il PCB

Versione 2.3 by FidoCAD

sbroglio manuale - dimensioni cm 6,3 x 8,9 - by gs



lo Slave (rivelatore velocità)

Il modulo rilevazione di velocità si compone di:

- Un arduino.
- Componentistica elettronica aggiuntiva come resistenze, led, connettori ect. ect.

per poter utilizzare in modo razionale quanto sopra si è deciso di realizzare un circuito stampato (in inglese viene chiamato PCB, printed circuit board, ovvero circuito stampato). In realtà i circuiti stampati realizzati sono più di uno questo perché, nel tempo, vi sono state evoluzioni e scelte differenti che hanno portato, appunto, a più schemi elettrici e più circuiti stampati sino a giungere a due progetti paralleli, il primo realizzato con metodi semiprofessionali, il secondo con metodi che possono essere realizzati con apparecchiature amatoriali ovvero "in casa", ambedue i progetti svolgono le stesse funzioni, la differenza sostanziale è nelle "dimensioni", il primo è più compatto poiché sfrutta ampiamente componentistica SMD mentre il secondo lo è di meno poiché utilizza componentistica tradizionale a foro passante più facilmente utilizzabile in ambito amatoriale.
La visualizzazione della velocità avviene sia sul display a bordo scheda che su un display remoto.

Schema elettrico

Arduino Uno dispone di 6 porte analogiche & digitali siglate come A0 ÷ A5 e 14 porte digitali siglate come D0 ÷ D13. Nel caso del modulo "slave" vengono utilizzate le porte A4 e A5 (I2C) e tutte le porte digitali ad esclusione della porta D13, le porte A4 e A5 sono specifiche per il protocollo I2C e sono, rispettivamente, i segnali SDA ed SCL, le porte D0 e D1 sono la porta seriale TTL replica di quella su porta usb. Le altre porte di Arduino sono così utilizzate:

A0 Disponibile per usi futuri
A1 Disponibile per usi futuri
A2 Ingresso per rilevatore passaggio 2
A3 Ingresso per rilevatore passaggio 1

D2 Pilotaggio anodo comune 3
D3 Pilotaggio anodo comune 2
D4 Pilotaggio anodo comune 1
D5 Ingresso porta D circuito integrato 7447
D6 Ingresso porta C circuito integrato 7447
D7 Ingresso porta B circuito integrato 7447
D8 Ingresso porta A circuito integrato 7447
D9 Disponibile per usi futuri
D10 Disponibile per usi futuri
D11 Disponibile per usi futuri
D12 Disponibile per usi futuri
D13 Segnale di Anomalia, qualsiasi circostanza che venga identificata come una anomalia (ad esempio l'assenza di comunicazione con il master) verrà segnalata con un led e portando questa porta a livello logico alto, in un secondo momento questo segnale potrebbe essere utilizzato da un modulo (attualmente opzionale) per la comunicazione mediante SMS su rete GSM.

L'alimentazione della scheda avviene attraverso il connettore “ALIM” a cui è presente nel pin 1 la tensione di +12V (può variare da +7V sino a +12V) che si collega, mediante un diodo switch (D1) per impedire inversioni di polarità, a un alimentatore stabilizzato che fornisce i +5V per arduino, il massimo carico che l'alimentatore stabilizzato è in grado di gestire è di circa 1A; sul pin 2 del connettore “ALIM” è presente la “massa” (GND) e sul pin 3, ponendolo a massa, si avrà il reset hardware.

La logica di funzionamento è la seguente, ai due ingressi di passaggio sono collegati due sensori ad effetto “hall” che sono in grado di trasformare una variazione di campo magnetico in un impulso elettrico che viene gestito da Arduino, al variare, in sequenza, dello stato dei sensori è possibile determinare una velocità poiché è nota la distanza fra i due ed il tempo con il quale questa doppia variazione avviene; questa velocità è convertita in formato BCD e visualizzata sui due display a sette segmenti e inviata al “master” mediante il bus I2C.

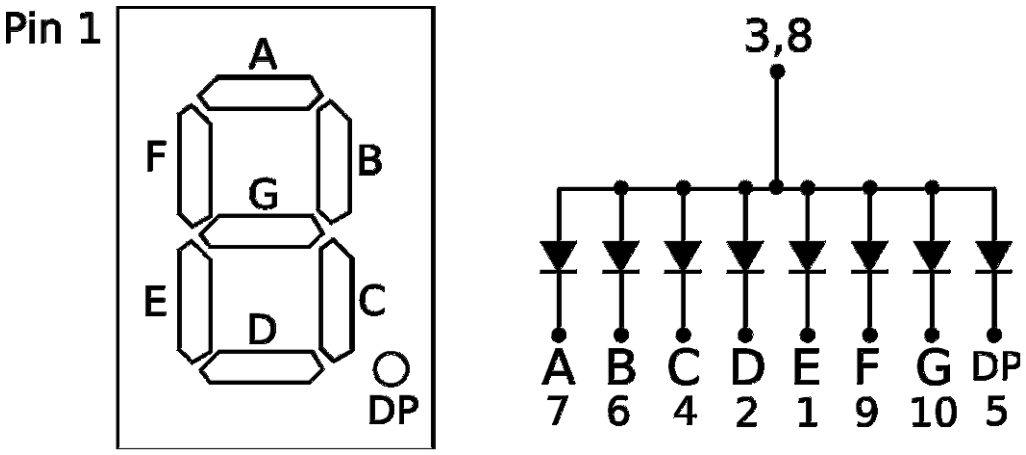
Il formato BCD è una codifica che prevede l'uso di quattro canali binari, attraverso questi quattro canali è possibile codificare sino a 16 numeri (0-15).

Decimale	Codifica BCD			
	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Una volta che un numero è codificato in formato BCD è possibile utilizzare un driver BCD per display led come, appunto, il 7447.

L'integrato 7447 è un driver per display led a 7 segmenti con anodo comune, mediante gli ingressi BCD è possibile convertire il “numero” espresso, appunto dalla codifica BCD, in una serie di segnali elettrici che possano far accendere un display led a 7 segmenti.

Un display, di esempio il circuito monta un display triplo, avente anodo comune ha uno schema come segue



Il driver 7447 converte i segnali BCD nel display a sette segmenti come segue:

Numero	Ingressi BCD				Segmenti su display						
	D	C	B	A	a	b	c	d	e	f	g
0	L	L	L	L	L	L	L	L	L	L	H
1	L	L	L	H	H	L	L	H	H	H	H
2	L	L	H	L	L	L	H	L	L	H	L
3	L	L	H	H	L	L	L	L	H	H	L
4	L	H	L	L	H	L	L	H	H	L	L
5	L	H	L	H	L	H	L	L	H	L	L
6	L	H	H	L	H	H	L	L	L	L	L
7	L	H	H	H	L	L	L	H	H	H	H
8	H	L	L	L	L	L	L	L	L	L	L
9	H	L	L	H	L	L	L	H	H	L	L
10	H	L	H	L	H	H	H	L	L	H	L
11	H	L	H	H	H	H	L	L	H	H	L
12	H	H	L	L	H	L	H	H	H	L	L
13	H	H	L	H	L	H	H	L	H	L	L
14	H	H	H	L	H	H	H	L	L	L	L
15	H	H	H	H	H	H	H	H	H	H	H

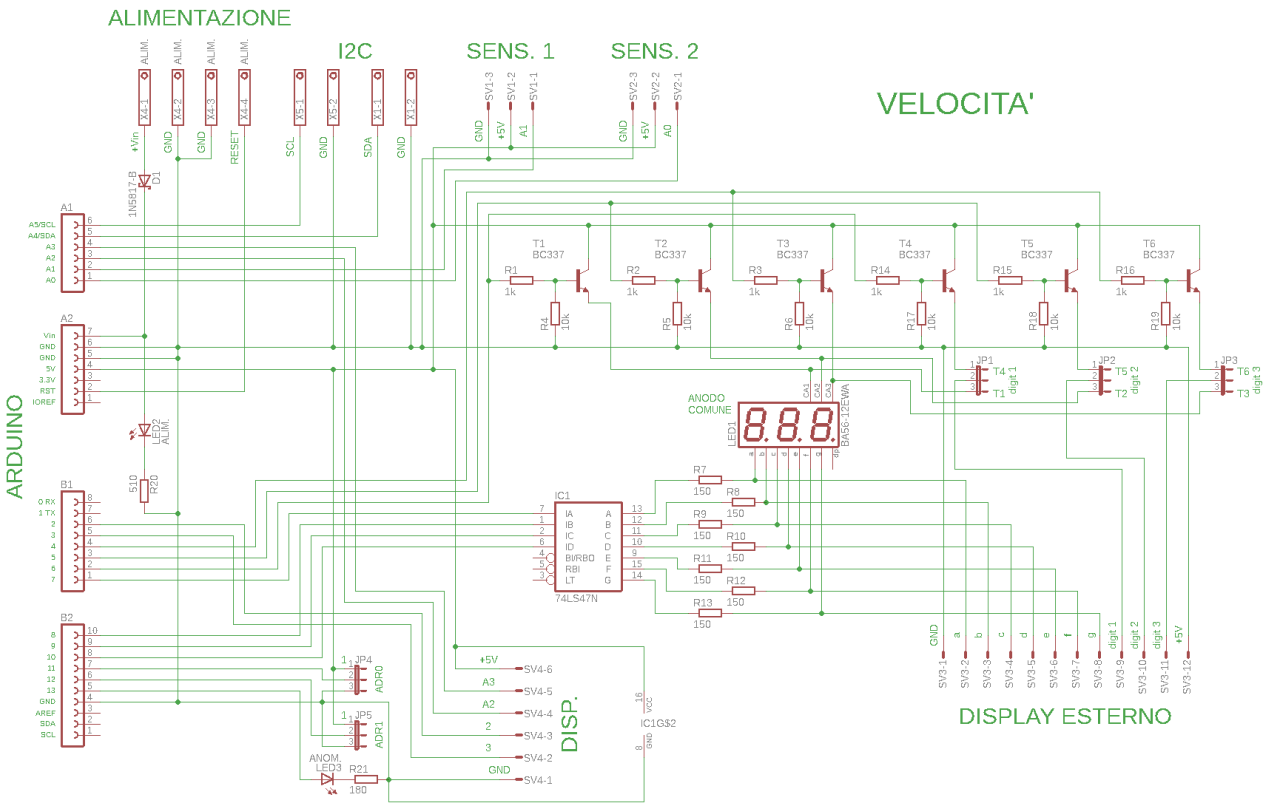
Essendo il display ad anodo comune il segmento si illuminerà nel momento in cui sull'uscita del driver 7447 è allo stato logico basso (low – L).

Il display utilizzato ha, oltre anodo comune per singola cifra, anche i catodi dei vari led sono in comune, per tanto, la piedinatura del display prevede i 7 catodi, i 3 anodi ed il catodo del "punto" (dot point), per illuminare il numero "123" su un display così cablato si utilizza il metodo del "multiplexer" che è, di fatto, una illusione ottica; il nostro occhio non riesce a percepire variazioni d'immagini se questo avvengono con una frequenza superiore ai 20/25 Hz, per tanto, se io accendo le singole cifre del display con un ciclo completo ad una frequenza superiore ai 25 Hz, ai miei occhi, vedrò come se tutte le cifre fossero accese.

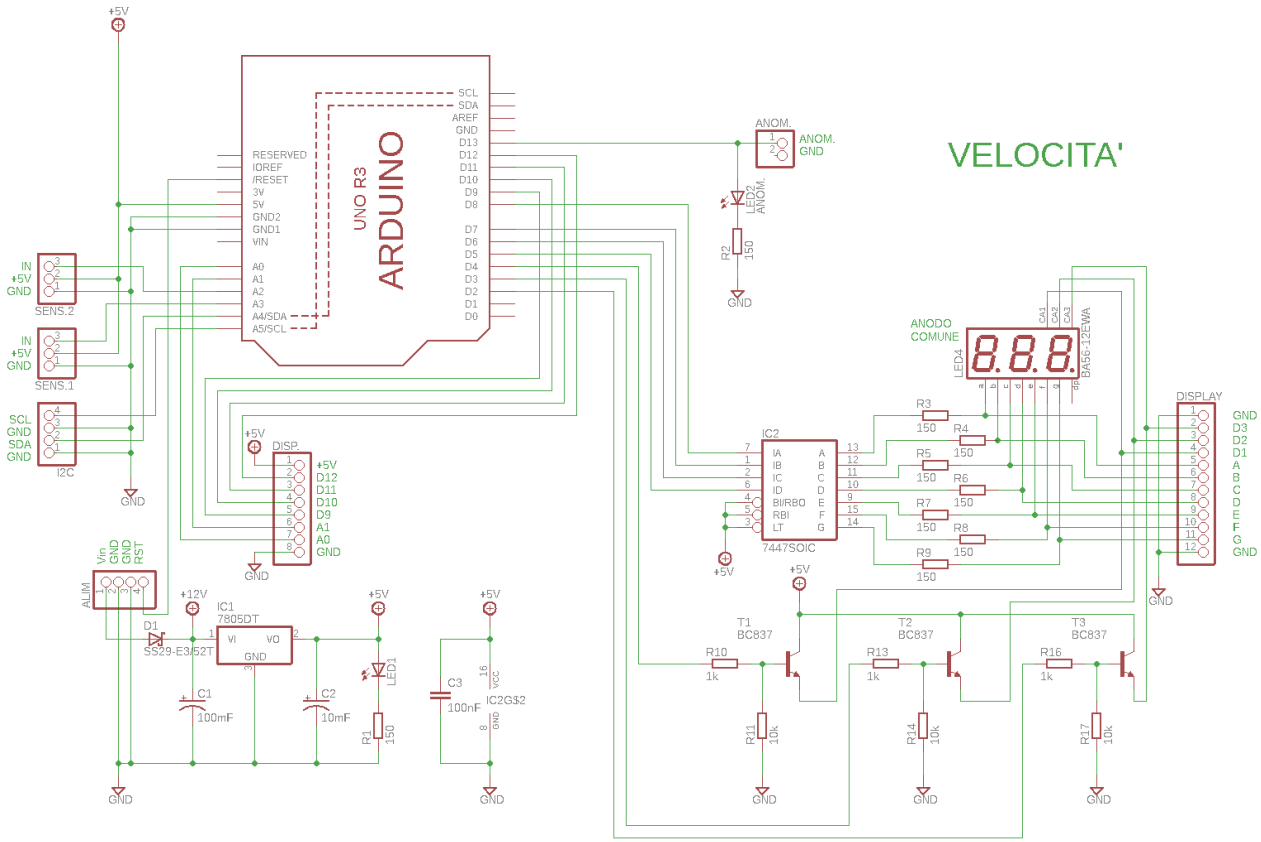
L'effetto “multiplexer” è ottenuto mandando sull'integrato 7447 la cifra da visualizzare in formato BCD e mandando in conduzione il transistor corrispondente lasciando gli altri due transistor aperti.

lo Schema Elettrico

Schema 1

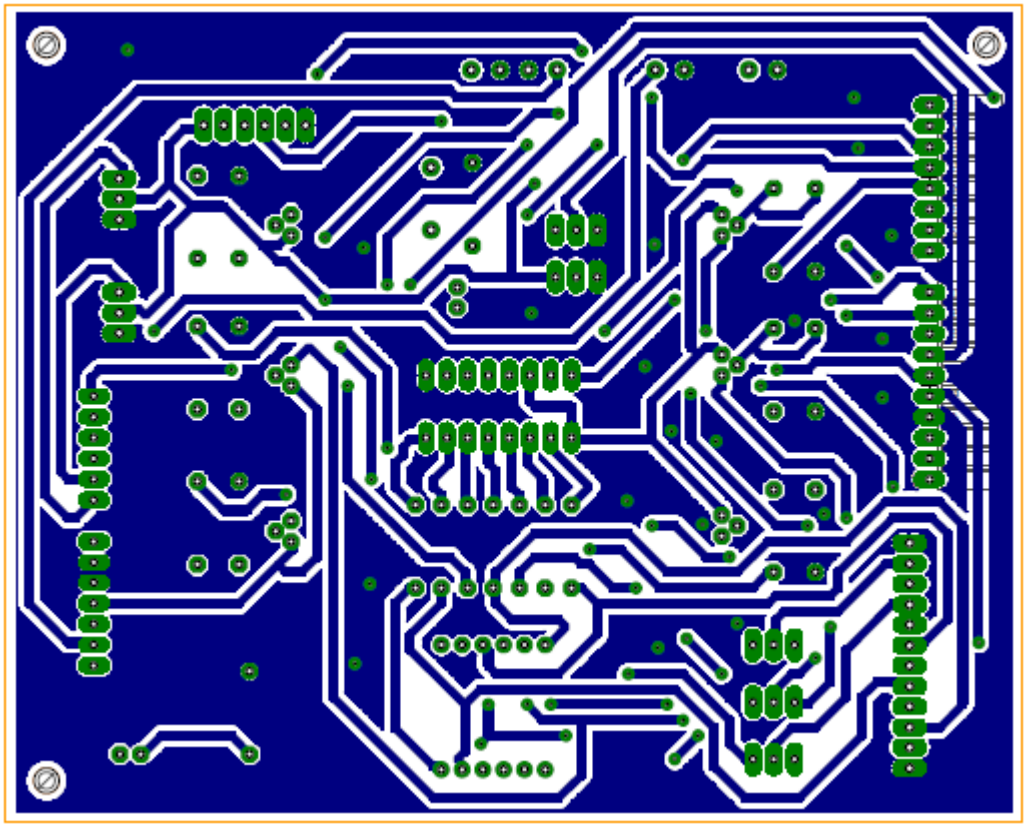


schema 2 - SMD

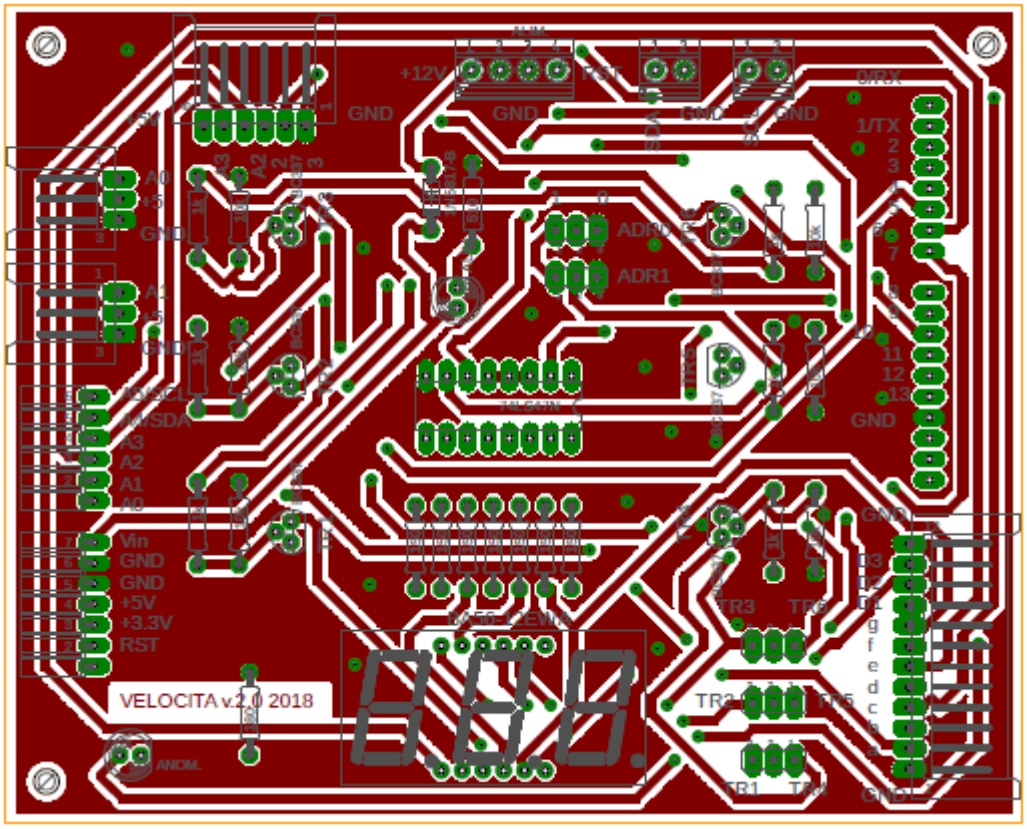


il PCB

vers. 1

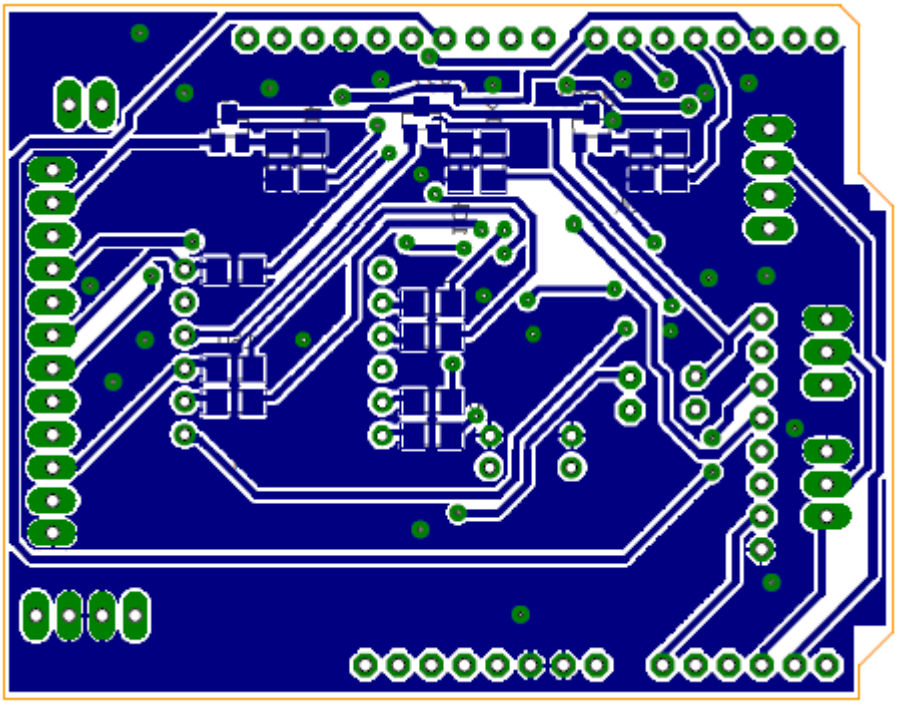


bottom

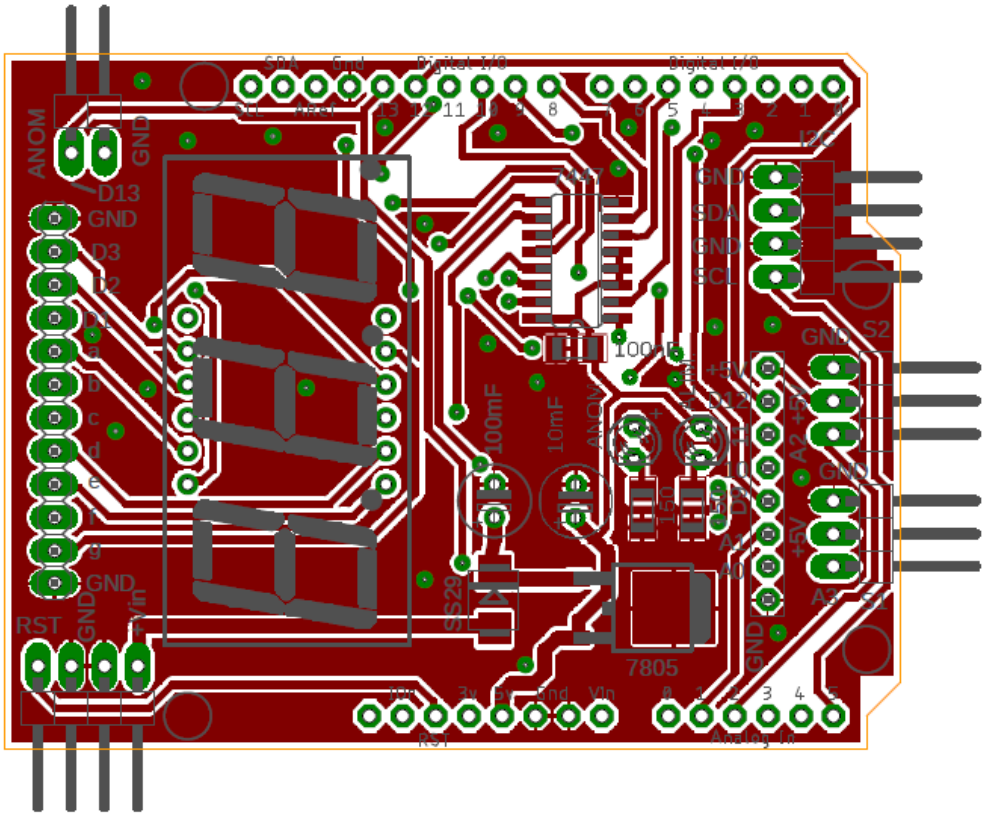


top

vers. 2



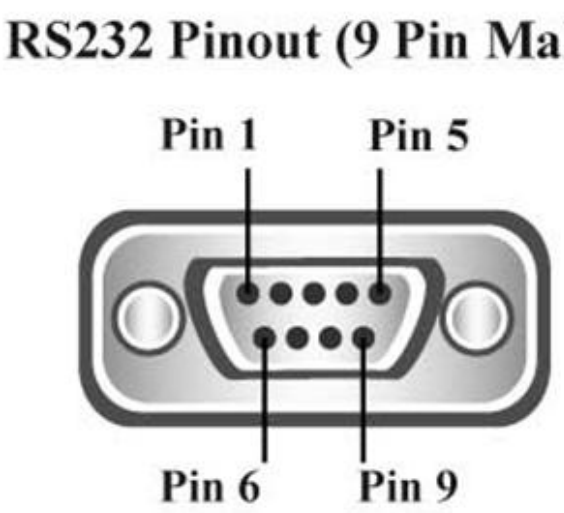
bottom



top

il diorama

i contatti del connettore semafori



1 VERDE

2

3 GIALLO

4

5 ROSSO

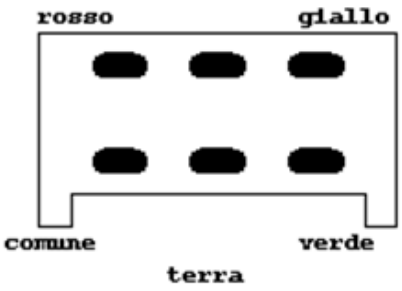
6

7 - 8 comune + (filo blu)

il semaforo "mega"

la realizzazione pratica

il connettore



maschio fisso su semaforo

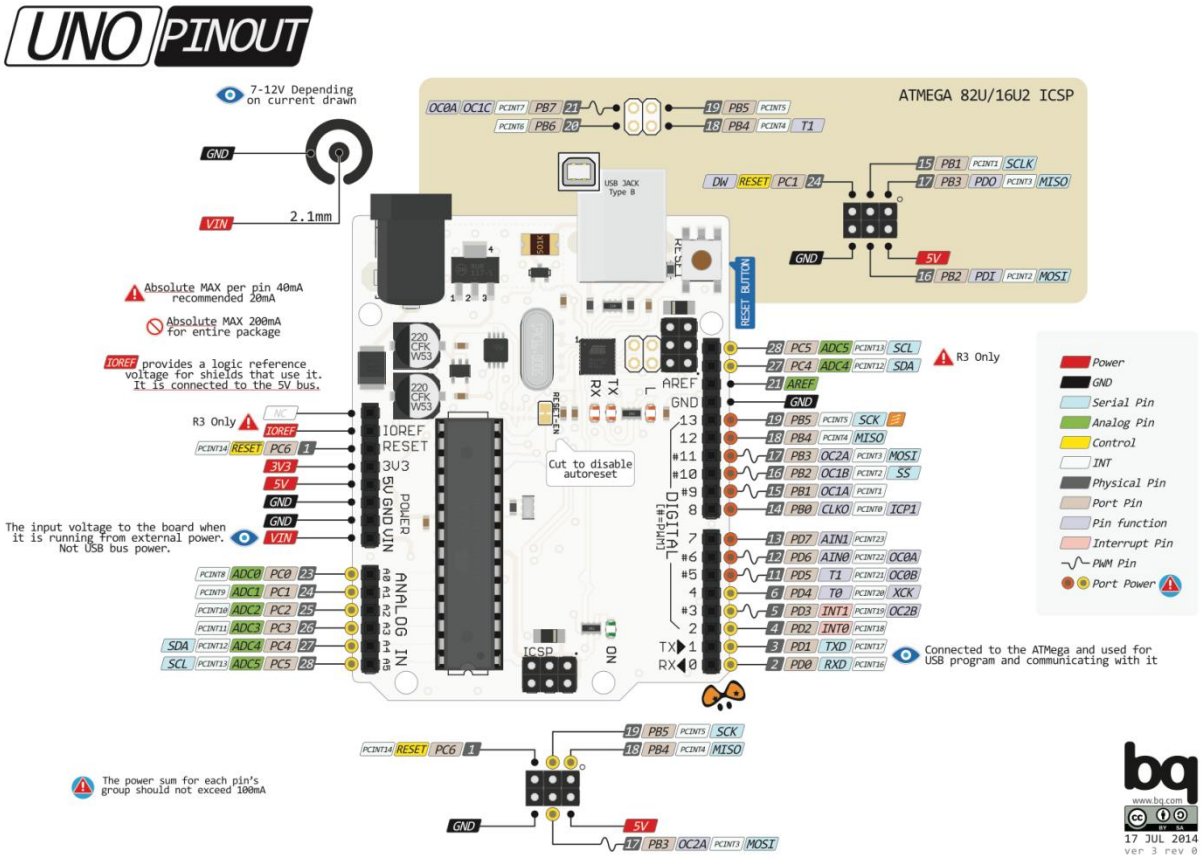
vista fronte

Cavo di collegamento

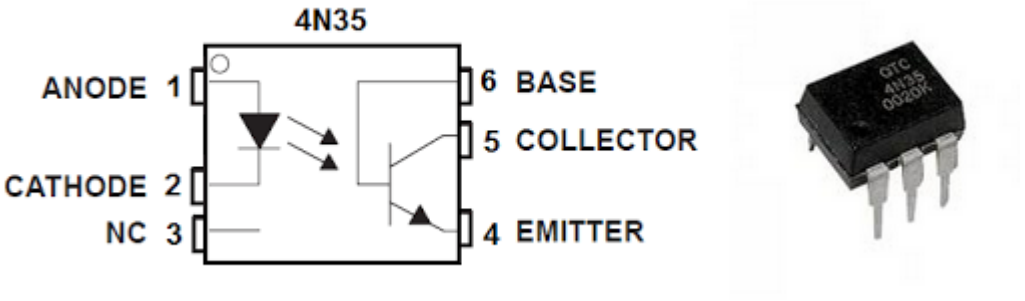
Luce / pin	colore cavo
rosso	blu
giallo	grigio
comune	marrone
terra	giallo/verde
verde	nero

la tecnologia e i componenti

Arduino



4n35



LA LIBRERIA I2C PER ARDUINO (Wire.h)
LE FUNZIONI

begin()/begin(address)
Questa funzione permette di inizializzare la libreria e attribuire il ruolo di Master o Slave alla scheda Arduino. Se non si specifica un indirizzo, Arduino acquisirà il ruolo di Master sul bus, mentre se si indica un indirizzo, sarà quello a cui risponde come Slave. Con questo meccanismo, è quindi anche possibile far comunicare fra loro delle schede Arduino che possono assumere entrambe i ruoli.

requestFrom(address, count)
Se Arduino è stato configurato come Master, con questa funzione può richiedere al dispositivo Slave con indirizzo address di inviargli un numero count di bytes, che poi andranno letti effettivamente con la funzione read().Per poter leggere i byte, questi dovranno essere anche disponibili nel buffer di comunicazione, da verificare con available().

beginTrasmission(address)
Sempre come Master, con questa funzione Arduino inizia la procedura di trasmissione di dati al dispositivo caratterizzato sul bus dall'indirizzo address. Si tratta di un processo che prevede anche il write() dei byte e l'effettiva trasmissione degli stessi quando il pacchetto da trasmettere viene indicato come completo attraverso la funzione endTrasmission().

write()
Con questa funzione si scrive in un buffer un byte alla volta; questo permette di creare l'intero vettore di byte da trasmettere in modo sincrono al dispositivo destinatario senza che ci siano pause fra i byte. Il protocollo i2c prevede infatti che la comunicazione avvenga con una cadenza precisa, senza pause fra i byte di un medesimo pacchetto. L'inizio della scrittura è determinato da beginTrasmission().

endTrasmission()
Quando viene invocata questa funzione, la libreria provvede a inoltrare i byte scritti nel buffer di trasmissione al dispositivo Slaveil cui indirizzo address è stato definito con la fuinzione beginTrasmission(address). Questa funzione restituisce un byte che ha i seguenti significati:
- 0: trasmissione andata a buon fine
- 1: troppi dati per le dimensioni del buffer di comunicazione
- 2: ricevuto un NACK errore di trasmissione dell'indirizzo
- 3: ricevuto un NACK sulla trasmissione dei dati
- 4: altro errore

available()
La funzione va utilizzata per avere come risultato un byte che indica il numero di byte pronti per essere letti dal buffer di ricezione con la funzione read(). I dati possono essere stati ricevuti come Master a seguito di una requestFrom(address) oppure come Slave all'interno della gestione con onReceive(). Il byte permette di dimensionare il ciclo con il quale verrà letto il buffer di ricezione.

read()
Questa è la funzione con cui vengono letti i byte ricevuti come Master o come Slave. In entrambi i casi, i byte sono prelevati dal buffer di ricezione il cui contenuto può essere

sempre controllato con available(). Inserendo la funzione read() in una struttura while(available()) continueremo a leggere dati finche il buffer di ricezione ne contiene, dato che solo quando il valore restituito da available() va a zero si ha l'uscita dal loop.

onReceive(handler)
Quando Arduino è configurato come Slave, deve aspettarsi comunicazioni dal MAster in qualsiasi momento. Per questo, grazie a onReceive(handler) è possibile definire la funzione "handler" che viene chiamata quando vengono ricevuti dati dal Master. Alla funzione viene passato solo un byte che è il numero di byte ricevuti nella trasmissione dal Master. Una volta impostatala funzione nello sketch con una sintassi del tipo void myHandler(int numBytes), questa verrà chiamata immediatamente alla ricezione dei dati, interrompendo l'esecuzione dello sketch.

onRequest(handler)
Simile alla precedente, questa funzione definisce la funzione handler da chiamare quando lo Slave riceve una richiesta di dati da parte di un Master. Se Arduino è configurato come Slave, deve rispondere alle richieste del Master e lo fa attraverso il gestore (handler) definito con questa funzione. Come arriva la richiesta, l'esecuzione del codice passa al gestore che deve onorare la richiesta per non creare un errore sul bus i2c

listati sperimentali

MASTER

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#define i2c_expander 0x20
#define i2c_slave 0x10
LiquidCrystal_I2C lcd(0x27,20,4); //dichiara il display "lcd" di tipo 20x4 all'indirizzo 0x20
void setup() {
Wire.begin();
Serial.begin(9600);
lcd.begin(); // Inizializza lcd
lcd.backlight();
lcd.setCursor(0 ,0);
lcd.clear();
lcd.print("Setup");
delay(1000);
lcd.clear();
lcd.clear();
lcd.print("I2C_EXPANDER SERIE");
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00000000);
Wire.endTransmission();
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b10000000);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b01000000);
Wire.endTransmission();
delay(500);
```

```
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00100000);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00010000);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00001000);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00000100);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00000000);
Wire.endTransmission();
delay(500);
lcd.clear();
}
void i2c_expander_test() {
lcd.clear();
lcd.print("CHECK I2C_EXPANDER");
lcd.setCursor(0,2);
Wire.requestFrom(i2c_expander,1);
if(Wire.available()){
int i2c_dato = Wire.read();
if (bitRead(i2c_dato, 1) == 1) {
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00001000);
Wire.endTransmission();
lcd.print("TASTO SX PREMUTO");
delay(2000);
}
if (bitRead(i2c_dato, 0) == 1) {
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00000100);
Wire.endTransmission();
lcd.print("TASTO DX PREMUTO");
delay(2000);
}
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b00000000);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(i2c_expander);
Wire.write((byte)0b10000000);
Wire.endTransmission();
delay(500);
Wire.beginTransmission(i2c_expander);
```

```
Wire.write((byte)0b00000000);
Wire.endTransmission();
delay(500);
}
lcd.clear();
}
void i2c_slave_test() {
lcd.clear();
lcd.print("TX MASTER -> SLAVE");
lcd.setCursor(0,2);
int x=0;
int y=0;
for (x==0; x<=3; x++) {
Wire.beginTransmission(i2c_slave);
Wire.write(x);
Wire.endTransmission();
delay(100);
Wire.requestFrom(i2c_slave, 1);
if (Wire.available()) {
lcd.print(x);
lcd.print("->");
int y=Wire.read();
lcd.print(y);
lcd.print(" ");
}
}
delay(5000);
}
void loop() {
lcd.clear();
i2c_expander_test();
i2c_slave_test();
lcd.clear();
}
```

SLAVE

```
#include <Wire.h>
#define i2c_slave 0x10
int I2C_dato = 0; // dato I2C
void setup() {
Serial.begin(9600);
Wire.begin(i2c_slave);
delay(100);
//eventi per la ricezione del dato
//e per la richiesta del dato
Wire.onReceive(RX_I2C);
Wire.onRequest(TX_I2C);
}
void RX_I2C() {
int x = 0;
if(Wire.available()){
x = Wire.read();
```

```
I2C_dato = x+1;
Serial.println(x);
Serial.println(I2C_dato);
}
}
void TX_I2C() {
Wire.write(I2C_dato);
Serial.println(I2C_dato);
Serial.println("");
}
void loop() {
// put your main code here, to run repeatedly:
}
```

i File disponibili

- arduino
- programmazione wiring
- Protocollo di comunicazione I2C
- schema slave 2.3-gs
- schema slave 2-zg
- schema slave 3-zg
- schema slave 4-zg
- pcb slave 2.3-gs
- datasheet 4n35
- cavo di collegamento
- connettore semafori

i Link utili

- https://www.logicaprogrammabile.it/comunicazione-tra-arduino-tramite-i2c/

i software utilizzati

- FreeCAD
- FidoCAD
- Eagle