ARDUINO

Scaricare dal web il suo ambiente di sviluppo “Arduino”.
All’interno dell’IDE nella sezione ‘file’ e in ‘esempi’ abbiamo delle porzioni di codici basi già funzionanti che possiamo usare.

ACCENSIONE E SPEGNIMENTO IN MODO CICLICO DI UN LED

```
void setup(){              //la funzione setup mi permette di inizializzare i valori
  pinMode(13, OUTPUT);     //il led del pin 13 dovrà essere erogato

}

void loop(){               //è il cuore di arduino, verrà eseguita ciclicamente

  //accendere il led, dargli il massimo della potenza (corrente: HIGH)
  digitalWrite(13, HIGH);

  //poi dobbiamo aspettare del tempo e (numero di millisecondi che vogliamo aspettare)
  //questo è il tempo in cui il led rimane acceso
  delay(1000);

  //spegnere il led e aspettare 1 secondo
  digitalWrite(13, LOW);
  //questo è il tempo in cui il led rimane spento
  delay(1000);

}
```

VARIABILI

Le variabili sono dei dati modificabili associati ad uno spazio di memoria e più questo spazio di memoria sarà pieno più le variabili avranno un valore.

```
int tempo = 1000; //dichiaro e inizializzo le variabili
int pinLed= 13;

void setup() {
  pinMode(pinLed, OUTPUT);
}

void loop() {
  digitalWrite(pinLed, HIGH);
  delay(tempo);
  digitalWrite(pinLed, LOW);
  delay(tempo);
}
```

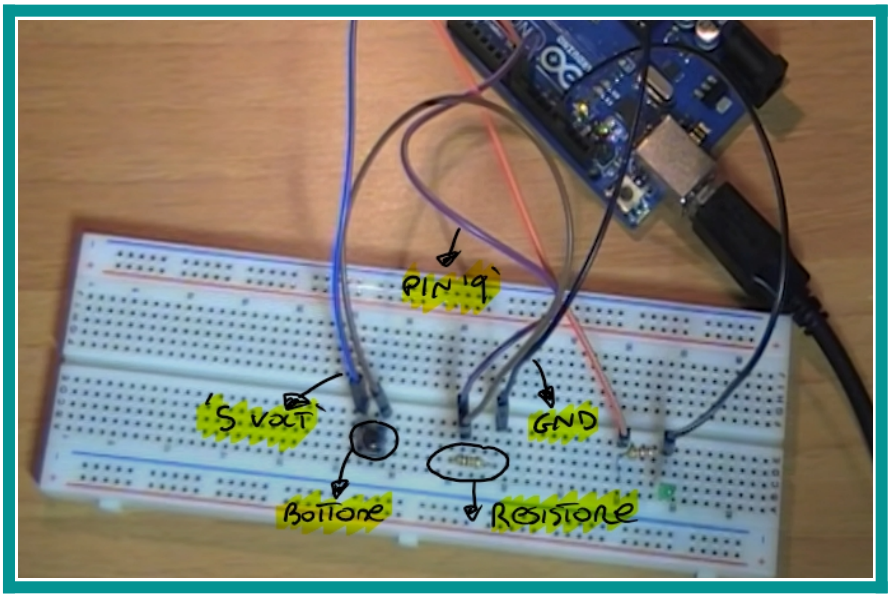
LED ESTERNO

Collego con il cavo della messa a terra (jumper) l'arduino nella sezione ***gnd*** con il piedino più corto del led inserito nella ***bredboard***; poi serve un altro cavo collegato con un'estremità ad un pin qualsiasi nell'arduino e nell'altra estremità con un buco qualsiasi della bredboard; per ultima cosa serve il ***resistore*** tra i 2 cavi nella bredboard e andrà inserito nel piedino più lungo del led.

PULSANTE

```
int pinLed=8;
int tempo=1000;
int buttonPin=9;

void setup() {
  pinMode(pinLed, OUTPUT);
  pinMode(buttonPin, INPUT);        //inizializzo il bottone

}

void loop() {
  //per vedere se il pulsante è premuto o no, utilizziamo una condizione
  if(digitalRead(buttonPin)==HIGH){        //leggo se il buttonPin è alto quindi azionato...
    digitalWrite(pinLed, HIGH);            //..accendo il led
  }

  else{              //altrimenti se il pulsante non è premuto, spengo il led
    digitalWrite(pinLed, LOW);
  }
}
```

Per collegare il **bottone** all'arduino metto il pulsante nella bredboard e lo collego con due fili al circuito. Il primo filo attacco un'estremità su un piedino del pulsante e l'altra estremità all'arduino nella voce **'5 volt'**, l'altro filo lo attacco all'altro piedino e al **resistore**, l'altro filo dal resistore al pin numero **'9'** e l'ultimo fino dal resistore al **'gnd'.**
Il resistore serve per scaricare la corrente a terra quando il pulsante non è premuto.



WHILE – DO WHILE – FOR

Programma che quando premo il pulsante spengo il led.

```
//finchè il pulsante è premuto, non eseguire nulla
void loop() {
  if(digitalRead(buttonPin)==HIGH){                 //se il pulsante è premuto...
    digitalWrite(pinLed, !digitalRead(pinLed));     //...leggo e cambio lo stato del led
    while(digitalRead(buttonPin)==HIGH){ //finchè il pulsante è premuto, non fare nulla, in questo modo dopo aver cambiato lo stato del pin, l'arduino aspetterà che il pulsante venga rilasciato
    }
  }
}
```

Programma che quando premo il pulsante, il led lampeggia.

```
void loop() {
  while(digitalRead(buttonPin)== HIGH){          //finchè il pulsante è premuto...
   digitalWrite(pinLed, !digitalRead(pinLed));   //...gli assegno lo stato contrario
   delay(tempo);
  }
}
```

```
void loop() {
  if(digitalRead(buttonPin)==HIGH){                 //se premo il pulsante...
    for(int i=0; i<3; i++){                         //lo faccio lampeggiare 3 volte
      digitalWrite(pinLed, HIGH);
      delay(tempo);
      digitalWrite(pinLed, LOW);
      delay(tempo);
    }
  }
}
```

COMUNICAZIONE SERIALE

La comunicazione seriale è la comunicazione che avviene tra l'arduino e il computer tramite cavo usb o jumper tra arduini. E' una comunicazione digitale quindi avviene solo tra 1 e 0.

```
int tempo=200;
int pinLed=8;
int buttonPin=9;

void setup() {
  pinMode(pinLed, OUTPUT);
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);   //Serial=comunicazione seriale a 9600 baud; baud=unità di misura che indica
                          il numero di simboli che vengono inviati al secondo.

}

void loop() {
  //vogliamo farci inviare il carattere 'a' ogni secondo
```

```
  Serial.print("a");
  delay(1000);
}
```

PROGRAMMA TEMPO DI REAZIONE

Programma che mi accende il led in un tempo casuale da 5 a 10 secondi e quando si accende devo premere il pulsante il più velocemente possibile e il programma mi stampa quanto tempo ci ho messo a spegnere il led dopo aver premuto il pulsante.

```
int tempo=200;
int pinLed=8;
int buttonPin=9;
int a;

void setup() {
  pinMode(pinLed, OUTPUT);
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);

}

void loop() {
  delay(random(5000,10000));                //in un tempo che varia da 5 a 10 secondi...
  digitalWrite(pinLed, HIGH);               //...accendo il led
  a=0;
  while(digitalRead(buttonPin)==LOW){       //finchè il pulsante si trova nello stato basso
    a++;
    delay(1);     //1 millisecondo
  }

  Serial.print("Hai aspettato: ");
  Serial.print(a);
  Serial.print(" millisecondi");
  digitalWrite(pinLed, LOW);            //spegno il led

}
```

OUTPUT ANALOGICO CON PWM

Un segnale analogico può essere di 2 tipi:

- **Vero:** quindi un voltaggio che non è né 0 volt né 5 volt ma qualcosa nel mezzo per esempio 2,3,4 volt.
- **Simulato:** questo segnale può essere fornito in output ad uno dei suoi dispositivi da arduino per esempio una lampadina, un led o un motore attraverso un procedimento chiamato ‘pwm’.

Il digitale può assumere il valore 0-1, l’analogico può assumere diversi valori 0-1-2-…-255 (0 spento, 255 il led sarà acceso al massimo, quindi più il valore è alto più luminoso sarà).

PROGRAMMA:

```
//per scrivere in analogico si utilizzano i pin nell'arduino con il simbolo '~'
int pinLed=6;

void setup() {
  pinMode(pinLed,OUTPUT);
```

```
}
void loop() {
  analogWrite(pinLed, 255); //il led sarà acceso al massimo

  //transizione da luminosità minima a massima
  for(int i=0; i<=255; i++){
    analogWrite(pinLed,i);
    delay(10);
  }

}
```

INPUT ANALOGICO E FUNZIONE MAP

Ci serve un ***potenziometro***, è un resistore variabile. Questo è formato da 3 ‘gambe’ che andranno inserire nella bredboard, la gamba destra andrà collegata con la terra ‘gnd’, la gamba sinistra la collego ai 5 volt e alla gamba centrale andrà in uno dei pin analogici (A0-A1-...-A5).

PROGRAMMA:

```
//andrà a leggere il valore del voltaggio e impostare la luminosità del led
int potenziometro; //da 0 a 1023
int valore; //da 0 a 255

void setup() {
  Serial.begin(9600);

}

void loop() {
  potenziometro = analogRead(A0);          //A0 su cui abbiamo collegato il segnale del potenziometro
  Serial.println(potenziometro);

  valore= map(potenziometro,0,1023,0,255); //funzione map: (valore che vogliamo convertire, i valori
                                           che possiede tale valore (da 0 a 1023), i valori a cui lo
                                           vogliamo convertire (da 0 a 255))

  analogWrite(9, valore);                  //9 è il pin a cui ho collegato il led
  delay(30);

}
```

STRINGA DA SERIALE

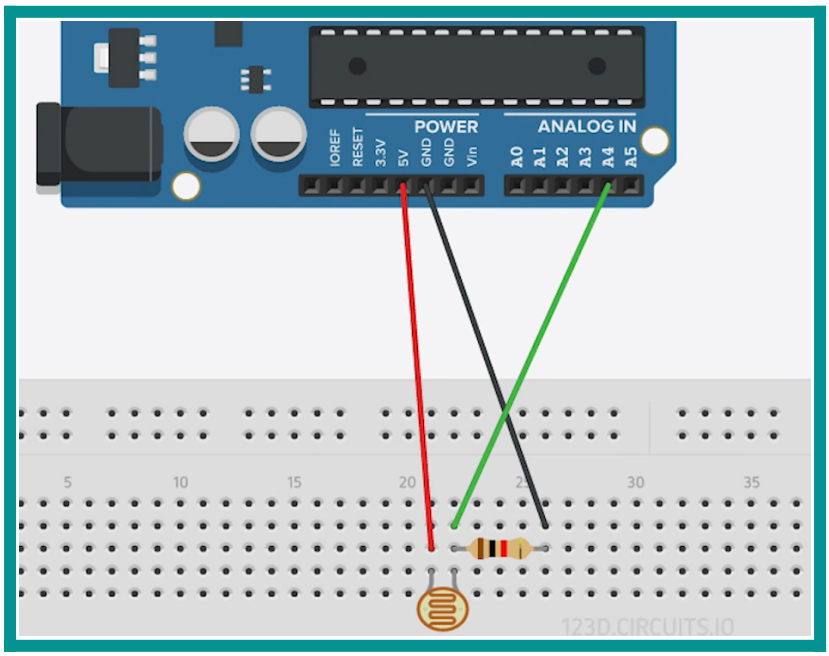
Programma in cui scriviamo un carattere e lo inseriamo in una stringa e poi accendiamo o spegniamo il led in base se abbiamo scritto accendi o spegni.

```
char c;
String stringa;

void setup() {
  Serial.begin(9600); //apro la comunicazione seriale
  pinMode(13, OUTPUT);
}
```

```
void loop() {
 //controlla se il computer sta inviando qualcosa all'arduino
 if(Serial.available()){
   stringa="";                  //inizilizzo la stringa come vuota, dove poi andrò a inserire i caratteri
   do{                          //continiamo a leggere e inserire il carattere della stringa fino...
    if(Serial.available()){     //metto Serial.available() così non mi stampa caratteri in più
      c=Serial.read();
      stringa+=c;
    }
   }while(c !='\n');            //...a quando non arriviamo a fine riga

   Serial.print(stringa);
   if(stringa=="accendi")
    digitalWrite(13, HIGH);
   else if(stringa=="spegni")
    digitalWrite(13, LOW);

 }
}
```

COSTANTI E FOTORESISTENZE

La fotoresistenza è un componente elettronico la cui resistenza è inversamente proporzionale alla quantità di luce che lo colpisce. Si comporta come un normale resistore, ma il suo valore in ohm diminuisce a mano a mano che aumenta l'intensità della luce che la colpisce.



PROGRAMMA:

Legge il valore della luce nella stanza e me lo stampi in seriale, se il valore letto in seriale è sotto alla soglia 280 (esempio) si accende un led, altrimenti rimane spento

```
int luce;
int soglia=280; //oppure #define soglia 280 = constante: valore che non cambia mai

void setup() {
```

```
  Serial.begin(9600);
  pinMode(8, OUTPUT);
}

void loop() {
  //3 istruzioni che mi vanno a leggere quanta luce c'è nella stanza
  luce=analogRead(A3);
    if(luce<soglia)
      digitalWrite(8,HIGH);
    else
      digitalWrite(8,LOW);
  Serial.println(luce);
  delay(1000);
}
```

SENSORI ULTRASUONI

Il sensore ultrasuoni va inserito nella bredboard, e dispone di **2 altoparlanti**, uno per trasmettere e uno per ricevere. Dall'altoparlante che trasmette, lui emette un suono ad ultrasuoni, il suono si propaga e 'colpisce' il primo oggetto che trova nel suo cammino, quindi l'eco orna indietro e andrà a finire nell'altoparlante per ricevere. Il sensore poi ha 2 pin:

- **trigger:** vuol dire mettere in funzione, quando viene azionato, il sensore emetterà il suono.
- **echo:** in questo pin il sensore ci restituirà un segnale digitale nel momento in cui arriva l'eco.

PROGRAMMA:

```
#include <NewPing.h>        //includo la libreria installata

#define PIN_TRIGGER 12      //pin dove andrò a dare il segnale perchè venga emesso il suono
#define PIN_ECHO 11
#define MASSIMO 100         //massima distanza di lettura 100cm

NewPing sonar(PIN_TRIGGER, PIN_ECHO, MASSIMO); //nome sensore

void setup() {
  Serial.begin(9600);
}

void loop() {
  int lettura=sonar.ping_cm();  //_cm perchè mi darà il risultato in centimetri
  Serial.print(lettura);
  delay(50);
}
```

COMUNICAZIONE SERIALE TRA ARDUINI

Per la comunicazione seriale tra arduini abbiamo due pin, uno per trasmettere chiamato ***TX*** e uno per ricevere chiamato ***RX***.
Comunicazione: *TX → RX* e *RX → TX*

//ARDUINO MEGA

```
void setup() {
  Serial.begin(115200); //seriale per il computer
  Serial1.begin(115200); //seriale per l'altro arduino, questo arduino ha più seriali
}

void loop() {
  if(Serial.available()){ //se c'è qualcosa da leggere
    char a = Serial.read();
    delay(10);
    Serial1.write(a);

    //comunicazione
    while(!serial1.available){

    } //mentre la seriale non è disponibile, aspetta, non fare nulla

    char b=Serial1.read();
    Serial.print(b);

  }

  delay(100);

}
```

//ARDUINO UNO

```
void setup() {
   Serial.begin(115200); //seriale per il computer

}
void loop() {
 if(Serial.available()){
    delay(100);
    Serial.write(a);
  }
  delay(50);

}
```

COMUNICAZIONE I²C

La comunicazione I²C può avere massimo 128 dispositivi collegati e ogni dispositivo ha un proprio indirizzo. Viene utilizzato lo schema ***master-slave***, in cui c'è il capo (master) che fornisce le indicazioni e da i compiti agli schiavi (slave) e possono essere altri arduini oppure sensori, chip.
Vengono utilizzati i cavi: ***SDA*** (serial data, pin A4 nell'arduino) e ***SCL*** (serial clock, dà il ritmo alle conversazioni, pin A5).
Comunicazione: *SDA → SDA e SCL → SCL*

Il programma mi stampa una volta al secondo ***L'altro ha scritto: ciao***

//master che richiede informazioni

```
#include <Wire.h>

void setup() {
  Wire.begin();
  Serial.begin(9600);

}
```

//slave che fornisce informazioni

```
#include <Wire.h>

void setup() {
  Wire.begin(3); // Wire.begin(INDIRIZZO SCHIAVO)
  Wire.onRequest(manda); //quando mi viene chiesto qualcosa, eseguo la funzione
```

```
void loop() {
  Wire.requestFrom(3,4);
// Wire.requestFrom(INDIRIZZO SCHIAVO, QUANTI BYTE DEVE RICHIEDERE)
  Serial.println("L'altro ha scritto: ");
  while(Wire.available()){
    char a =Wire.read();
    Serial.print(a);
  }

  Serial.println();
  delay(1000);
}
```

```
}

void loop() {
  delay(100);

}

void manda(){
  Wire.write("ciao"); //i 4 byte del master 'c' 'i' 'a' 'o'
}
```

EEPROM

La ***EEPROM*** è una memoria non volatile, utilizzata per memorizzare piccole quantità di dati che devono essere mantenuti quando viene tolta l'alimentazione elettrica.

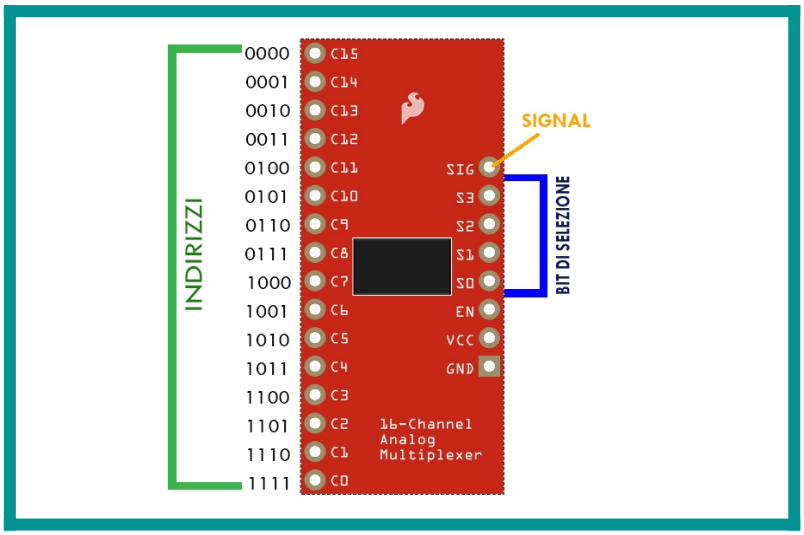
```
#include <EEPROM.h>
void setup() {

}

void loop() {
  EEPROM.read(0); //andrà a leggere ciò che sarà scritto nel byte 0, arriva fino a 1023
  EEPROM.write(0, 'o'); //in posizione zero scrivo la lettera 'o'

  //per percorrere le celle della eeprom
  for(int=0; i<EEPROM.length(); i++)
    EEPROM.write(i, 0); //scrivo 0 su ogni cella, resetto la eeprom
}
```

MULTIPLEXER

Il ***multiplexer*** è un circuito che funge da selettore elettronico. Si compone di

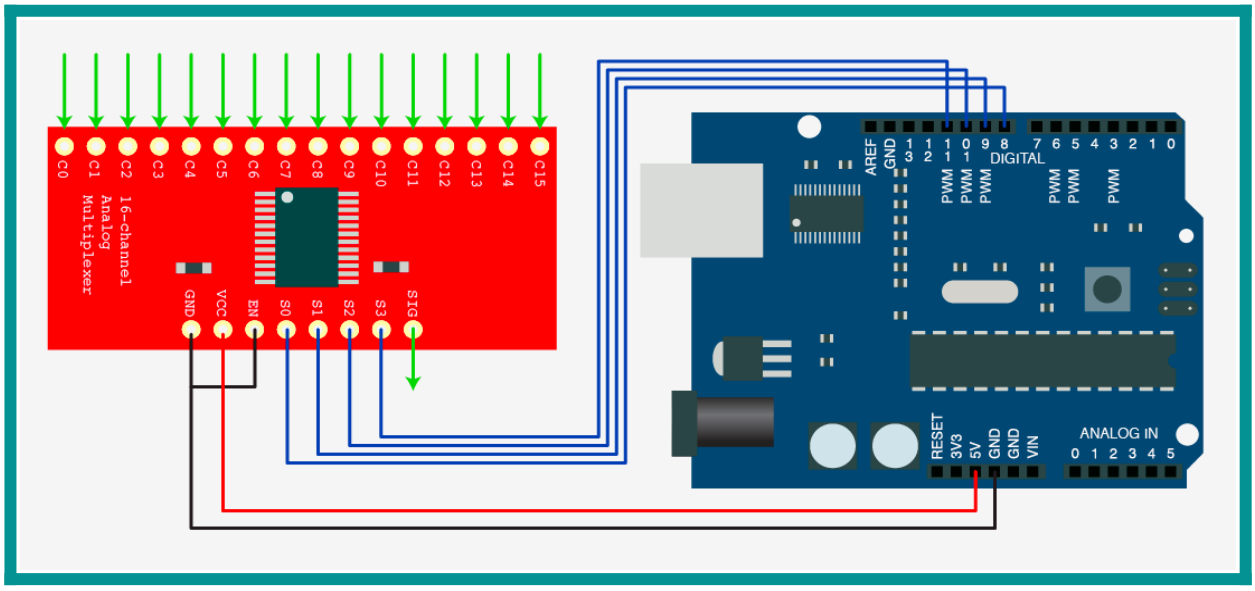
- n linee d'ingresso (in foto C0-C15)
- log2n linee di selezione (S0-S3)
- 1 linea di uscita (SIG)

Il mux è un componente digitale, pertanto lavora su coppie di ***bit 0-1***. In particolare ciascuna delle 16 linee di ingresso ha un indirizzo in base binaria a 4 bit che viene individuato dai 4 bit di selezione. Ecco il funzionamento: Arduino compone un indirizzo settando ad 1 (HIGH) e 0 (LOW) i bit di selezione e tale indirizzo viene utilizzato per individuare il pin a cui sarà collegato il SIGNAL.



HARDWARE

Al contrario di quanto possa sembrare, il collegamento è molto semplice:



PROGRAMMA:

```
int s0 = 8;      //Sono i pin di Arduino che rappresentano i bit di selezione e vanno collegati al multiplexer.
int s1 = 9;
int s2 = 10;
int s3 = 11;
int sig = 0;     //E' il SIGNAL del mux, cioè il pin che di volta in volta sarà collegato ad uno dei 16 ingressi.

/* Dichiarazione della matrice "multiplexer" che contiene gli indirizzi degli ingressi: in questo caso, siccome abbiamo un mux 16:1 la matrice ha dimensioni [16][4]. Se ad esempio avessimo un mux 8:1, allora la matrice sarebbe [8][3] perchè 3 bit sono sufficienti  per codificare 2^3 = 8 indirizzi */

int multiplexer[16][4] = { {0,0,0,0}, //canale 0
                           {0,0,0,1}, //canale 1
                           {0,0,1,0}, //canale 2
```

```
                    {0,0,1,1}, //canale 3
                    {0,1,0,0}, //canale 4
                    {0,1,0,1}, //canale 5
                    {0,1,1,0}, //canale 6
                    {0,1,1,1}, //canale 7
                    {1,0,0,0}, //canale 8
                    {1,0,0,1}, //canale 9
                    {1,0,1,0}, //canale 10
                    {1,0,1,1}, //canale 11
                    {1,1,0,0}, //canale 12
                    {1,1,0,1}, //canale 13
                    {1,1,1,0}, //canale 14
                    {1,1,1,1}  //canale 15
                  };

void setup() {

  pinMode(s0,OUTPUT);
  pinMode(s1,OUTPUT);
  pinMode(s2,OUTPUT);
  pinMode(s3,OUTPUT);
  Serial.begin(9600);
}

void loop() {

  interfaccia(2);
  Serial.read(sig);      /* Potete leggere o scrivere sul pin "sig" a seconda di ciò che è connesso
                            all'ingresso selezionato mediante la funzione "interfaccia". */
  interfaccia(6);
  Serial.read(sig);
}

/* La funzione "interfaccia" prende in input il numero (intero) del canale che si vuole collegare al
SIGNAL per leggere o scrivere: in questo caso la funzione accetta valori nel range 0-15 perchè le
righe della matrice multiplexer sono 16.  */

void interfaccia(int canale) {
  int controllore[] = {s3,s2,s1,s0}; //Tale matrice rappresenta l'indirizzo di selezione

   /* Il ciclo for scrive nella linea di selezione, bit per bit, l'indirizzo dell'ingresso che si vuole
    collegare al SIGNAL (la riga infatti è data dal canale, che è un parametro di input) */
  for(int i=0;i<4;i++) {   //4 perchè s3,s2,s1,s0 = 4
    digitalWrite(controllore[i],multiplexer[canale][i]);
  }
}
```

DISPLAY I^2C

```
#include <LiquidCrystal_I2C.h>

//inizializzo lo schermo
LiquidCrystal_I2C schermo(0x27,16,2) //ci sono due righe e per ogni riga max 16 caratteri
//per trovare 0x27 cerco su google arduino scanner, copio il codice nell'ide e lo faccio partire e mi trova così il numero 0x..

void setup(){
  schermo.begin();
  schermo.prin("ciao mondo");

  schermo.setCursor(3,1); //il cursore va avanti di 3 spazi e mi scrive nella seconda riga
  schermo.print("ciao");
}

void loop(){

}
```