Arduino

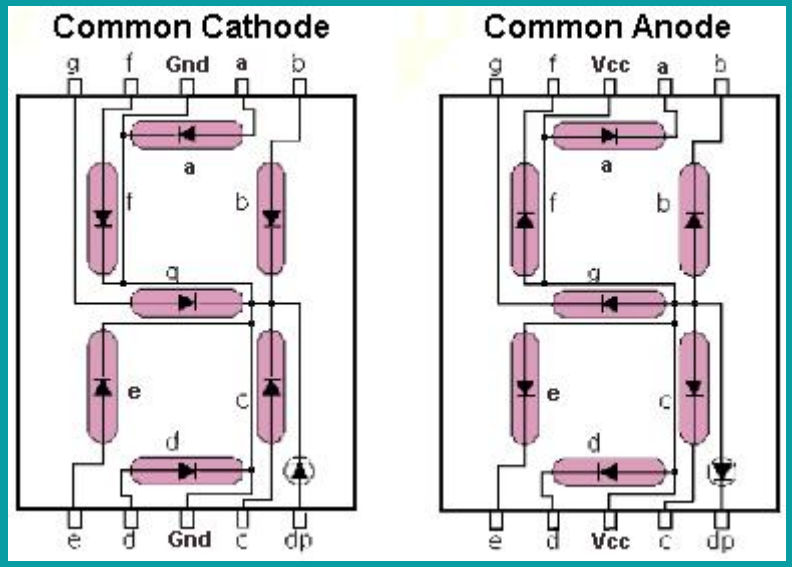
Led, pulsanti, PWM, tastierino,
display sette segmenti, display LCD

Accendere e spegnere i led in sequenza

```
void setup() {
 for(int i=2;i<7;i++)
 pinMode(i, OUTPUT);
}
void loop() {
  for(int i=2;i<7;i++)
  {
    digitalWrite(i,HIGH);
  delay(4000);
  digitalWrite(i,LOW);
  delay(1000);
  }
}
```

Display a sette segmenti

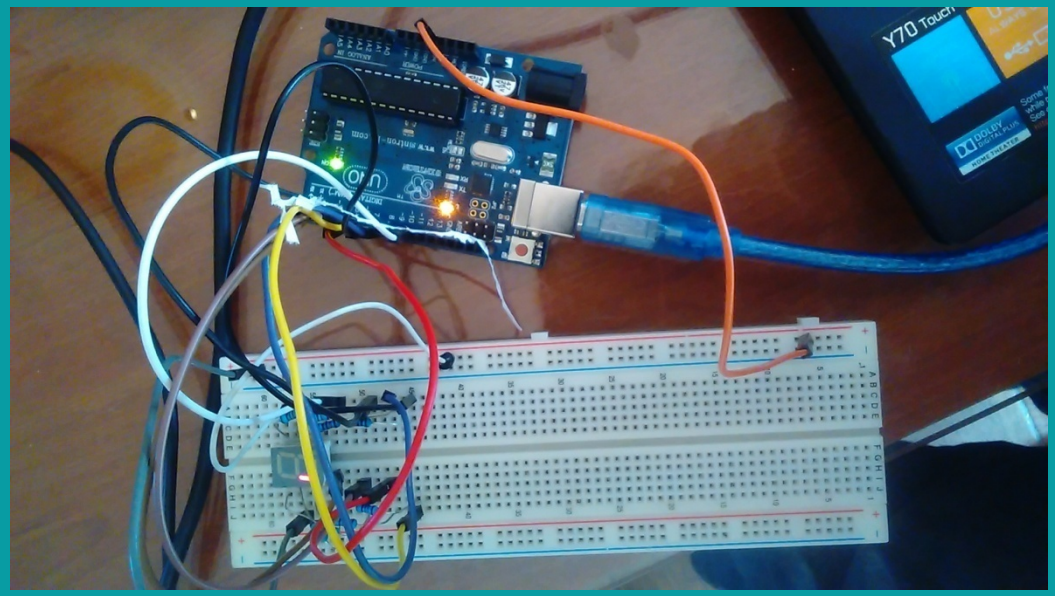
Il display a sette segmenti è formato da sette led che si possono illuminare indipendentemente l'uno dall'altro. Con questo display si può simulare un contatore o un convertitore.
I display a sette segmenti possono essere ad anodo comune o a catodo comune
La loro struttura è qui riportata
Come si vede, ad ogni pin corrisponde un led e, ciascuno, può essere collegato su una uscita di Arduino.
Se il display è a catodo comune, un segmento si illuminerà solo se il relativo pin si trova a livello logico basso; viceversa se un display è ad anodo comune, il segmento si illuminerà se il relativo pin è a livello logico alto.



Esempio

```
int i;
void setup() {
  for(i=3;i<10;i++){
    pinMode(i,OUTPUT);}
    for(i=3;i<10;i++){
    digitalWrite(i,HIGH);}
}
void loop() {for(i=3;i<10;i++){
digitalWrite(i,LOW);
delay(3000);
digitalWrite(i,HIGH);}
}
```

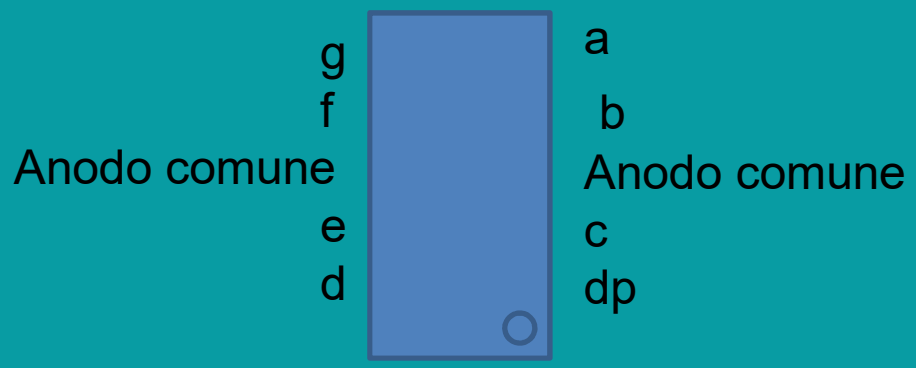
Questo è un semplice esempio per provare il display a 7 segmenti. L'esempio è per un display ad anodo comune



Contatore

Dall'esperimento fatto, si può dedurre la piedinatura del display. Un modo corretto anche per meglio coordinare la parte software con quella hardware, è di porre i pin nel seguente modo:

a->3
b->4
c->5
dp->6
d->7
e->8
f->9
g->10



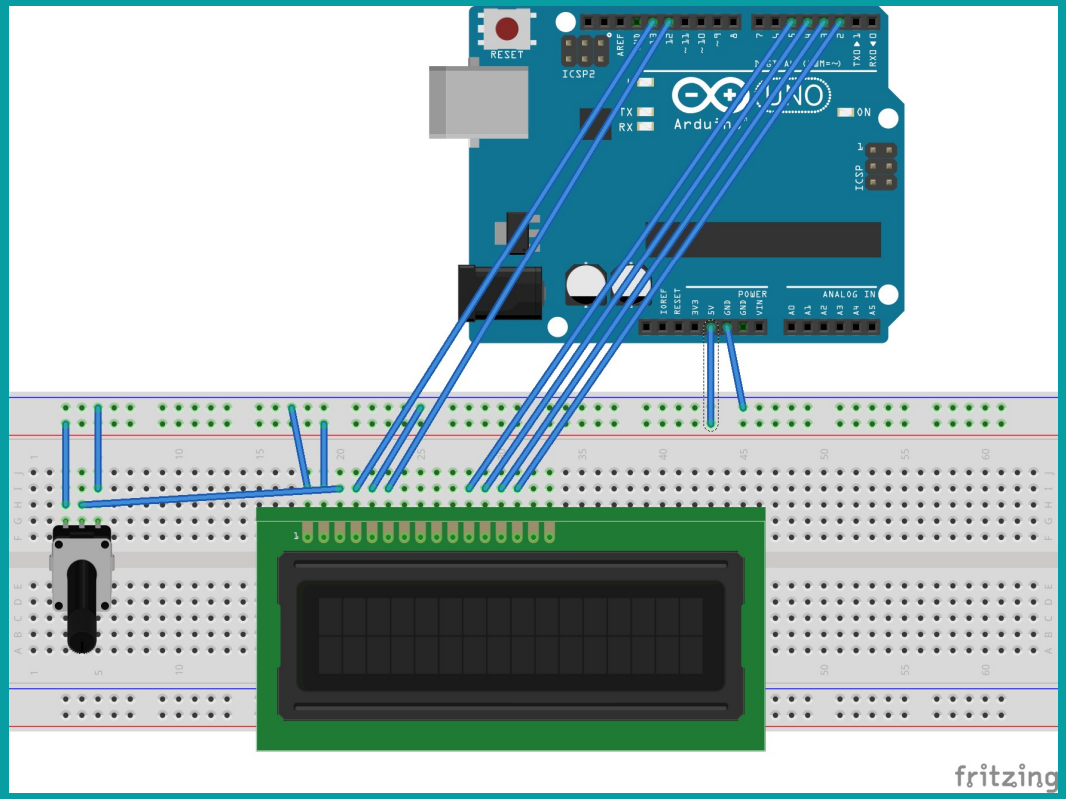
Il conteggio dei pin è stato fatto in senso antiorario prendendo come riferimento il puntino

Programma contatore

```
int i;
void setup() {
  for(i=3;i<=10;i++){
    pinMode(i,OUTPUT);}
    for(i=3;i<=10;i++){
    digitalWrite(i,HIGH);}
}
void loop() {
zero();
delay(3000);
spento();
uno();
delay(3000);
spento();
due();
delay(3000);
spento();
}
```

```
 spento(){
  for(i=3;i<=10;i++)
  digitalWrite(i,HIGH);
}
void zero(){
  for(i=3;i<10;i++)
  digitalWrite(i, LOW);
}
void uno(){
  digitalWrite(4,LOW);
  digitalWrite(5,LOW);
}
void due(){
  digitalWrite(3,LOW);
  digitalWrite(4,LOW);
  digitalWrite(10,LOW);
  digitalWrite(6,LOW);
  digitalWrite(7,LOW);
}---------
```

LCD

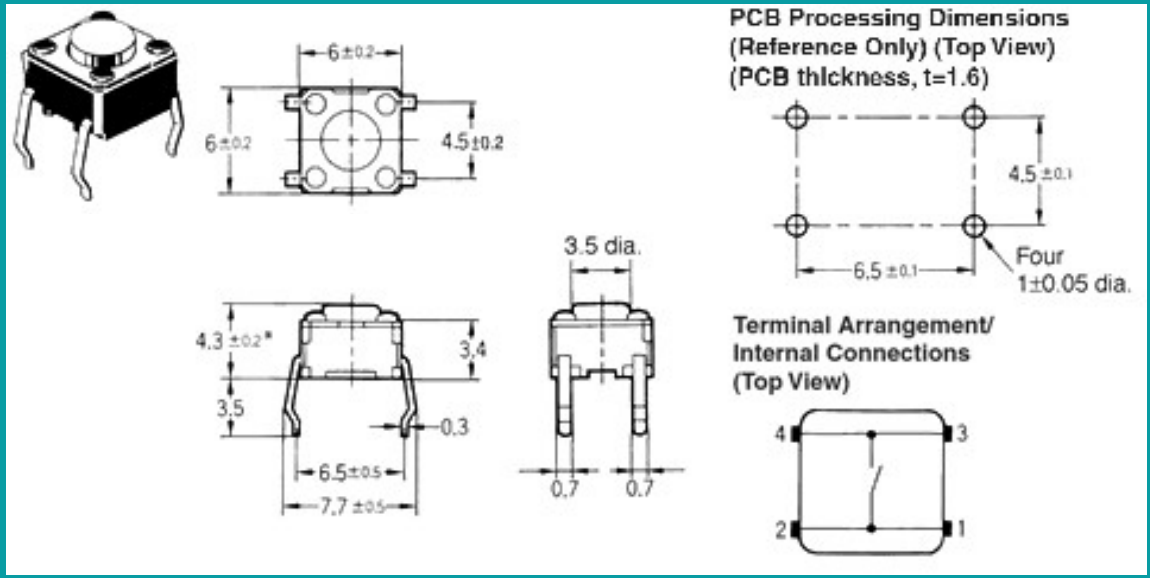



```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
  lcd.begin(16, 2);
  lcd.print("Hello BABY");
}
void loop() {
  lcd.setCursor(0, 0);
 for(int i=0;i<100;i++){
  lcd.print(i);
 delay(3000);
 lcd.clear();}
}
//lcd(RS,E,D4,D5,D6,D7)
```

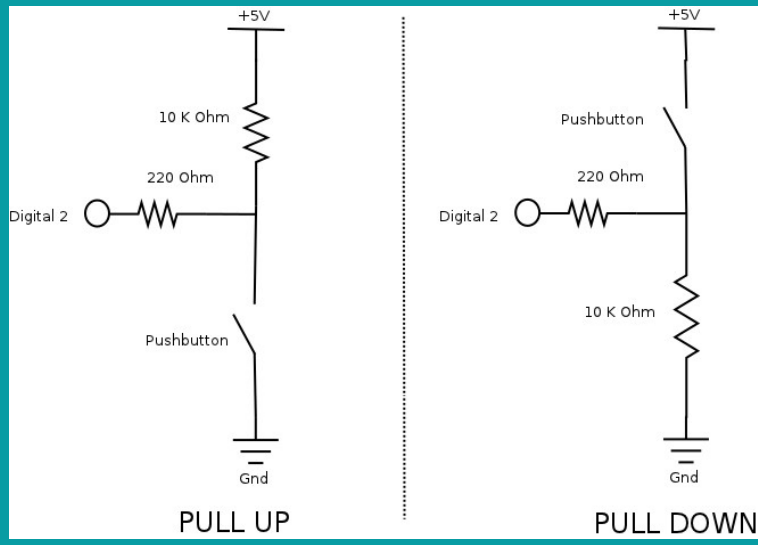
LCD inseguitore di numeri

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup() {
  lcd.begin(16, 2);
  lcd.print("Hello BABY");
}
void loop() {
  lcd.clear();
int  i=0;
 do{if(i<=14){
 lcd.setCursor(i,0);
 lcd.print(i);
 delay(3000);
 lcd.clear();}
 else if(i>14 && i<28){
 lcd.setCursor(i-14,1);
 lcd.print(i);
 delay(3000);
 lcd.clear();}
 i++;}while(i<28);
}
//lcd(RS,E,D4,D5,D6,D7)
```

Schema interruttore



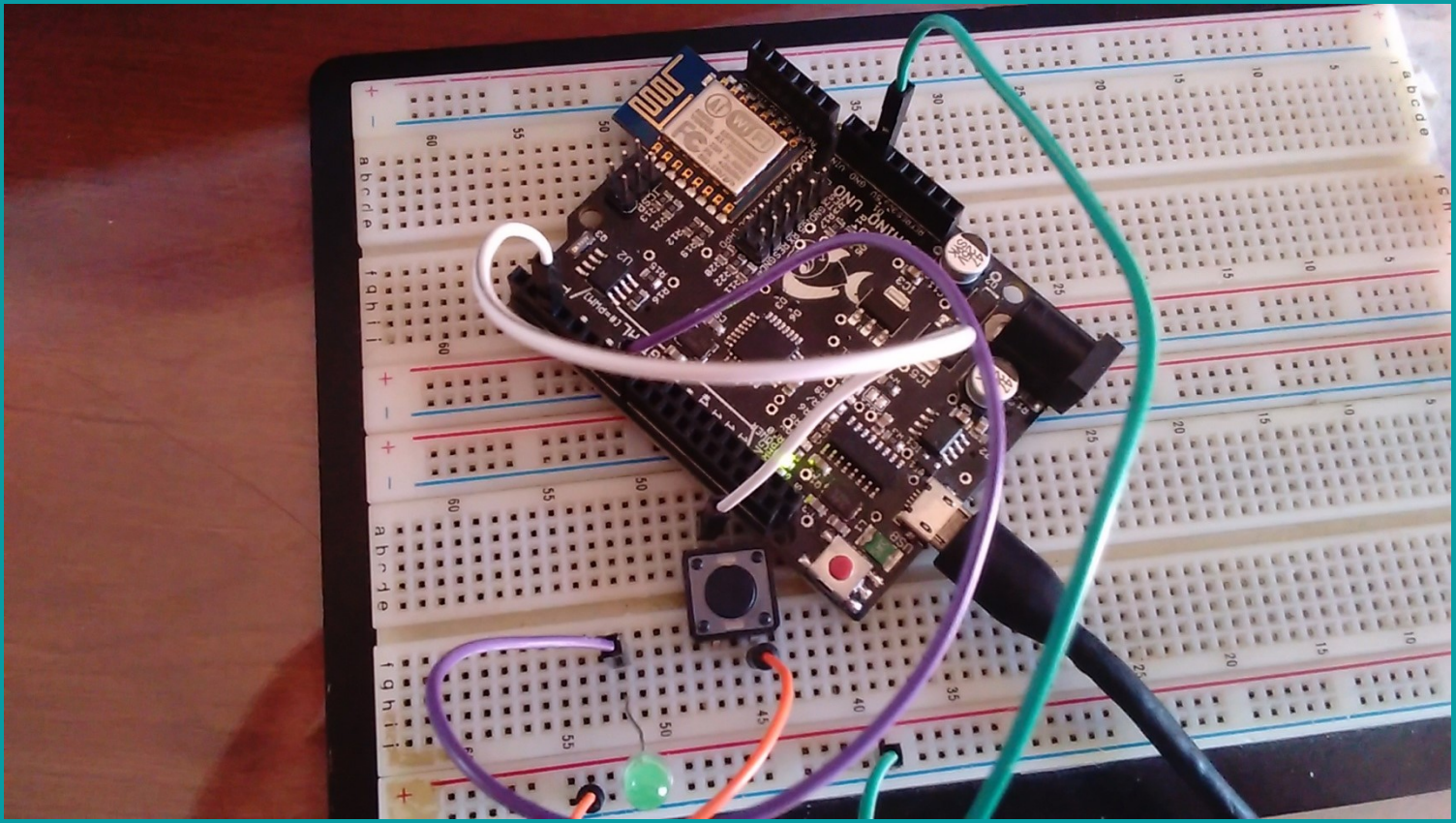
Resistenza di pull up e pull down



•Quando si attiva la resistenza di pull-up interna relativa ad un pin scelto come input,
la corrente che fluisce è debole
Le resistenze di Pull-up e Pull-down si utilizzano in elettronica per forzare uno stato logico ed eliminare eventuali fluttuazioni di corrente
Con arduino, per abilitare una resistenza di pull up basta forzare alto il livello logico iniziale del pin di input
I pin di ingresso dei circuiti digitali, compresi quelli dei microcontrollori, presentano una impedenza abbastanza alta, assieme alla possibilità di operare su segnali ad elevata frequenza. Questo significa che, se sono lasciati flottanti, ovvero senza alcun riferimento di tensione, sono soggetti a captare qualsiasi disturbo elettromagnetico o elettrostatico e quindi a commutare le loro uscite in modo casuale. E si definisce ***floating*** - flottante un pin di ingresso a cui non è collegato nulla.

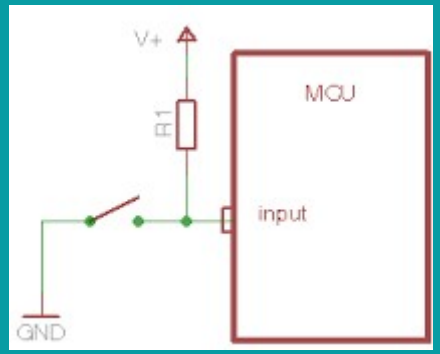
Input/ Output con resistenza di pull up

- Abilitare le resistenze di pull up



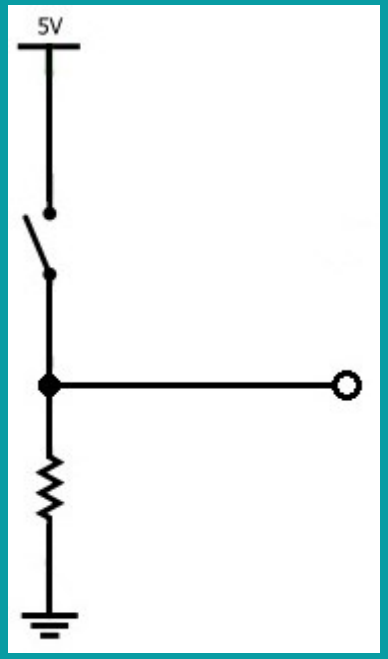
Interruttore e led con resistenza di pull up

```
int led = 7;
int in = 2;
int val = 0;
void setup() {
    pinMode(led, OUTPUT);
    pinMode(in, INPUT);
   digitalWrite(in, HIGH); //si attiva la resistenza di pull up
              }
void loop(){
    val = digitalRead(in);
        if (val == HIGH) {
              digitalWrite(led, LOW);
        } else {
              digitalWrite(led, HIGH);
               }
           }
```



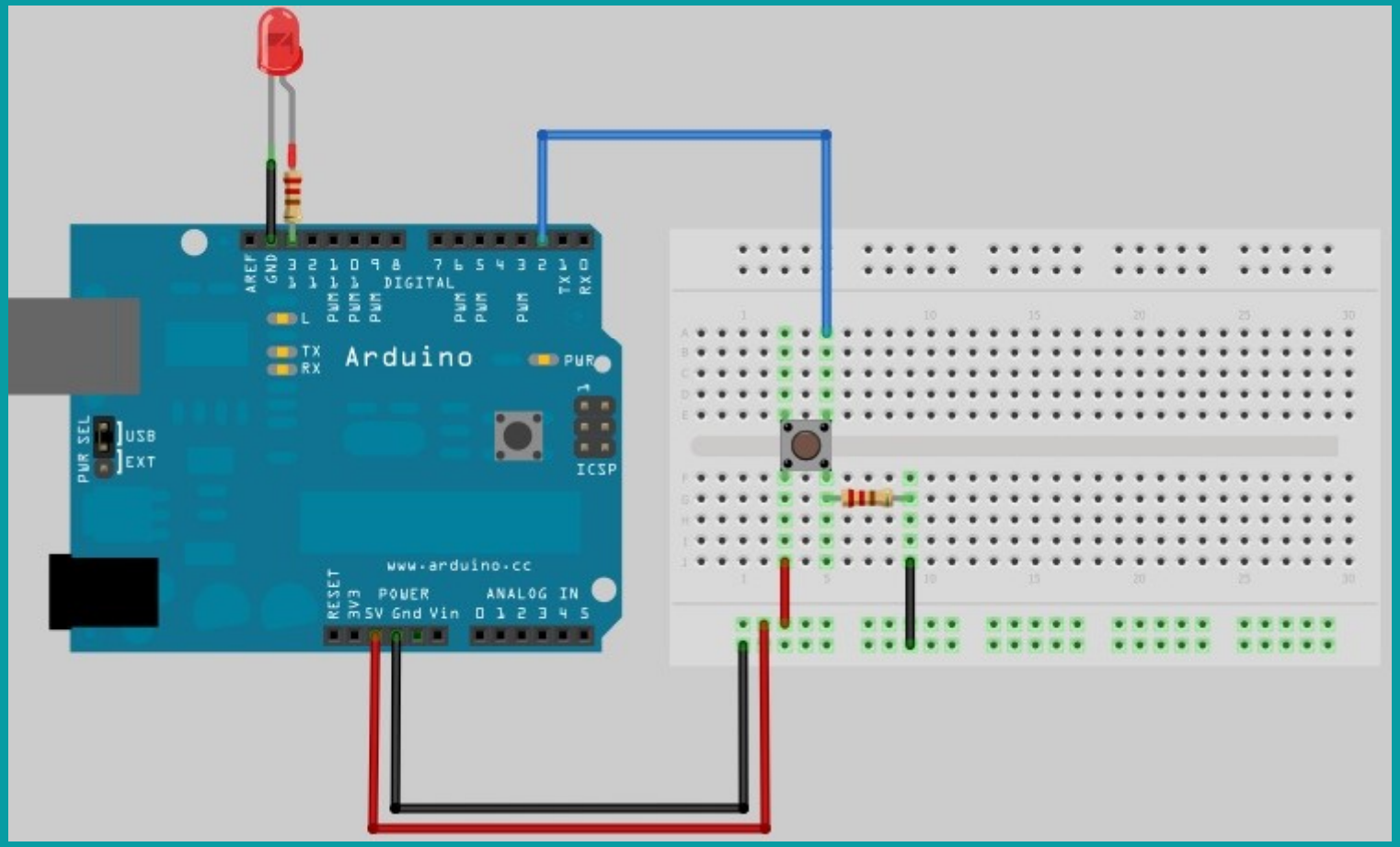
Interruttore e led senza resistenza di pull up

```
const int led=13;
const int bottone=3;

void setup() {
  pinMode(led, OUTPUT);
  pinMode(bottone, INPUT);
}

void loop() {
  int valore=digitalRead(bottone);
  if(valore==HIGH){
    digitalWrite(led,HIGH);
  }
  else digitalWrite(led,LOW);
}
```

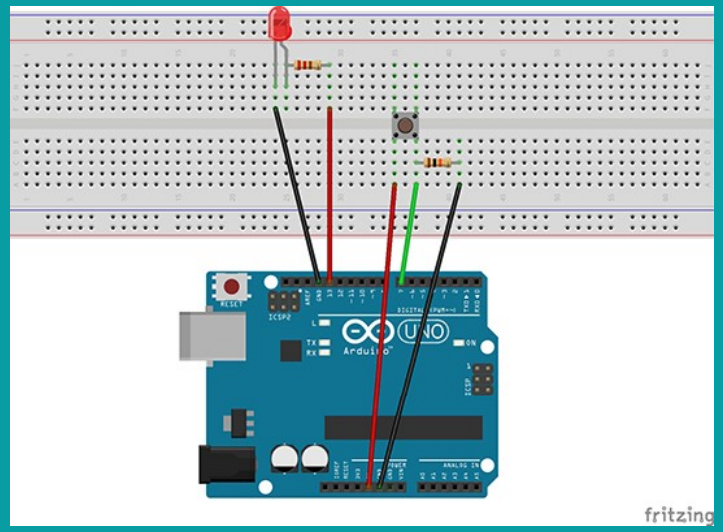


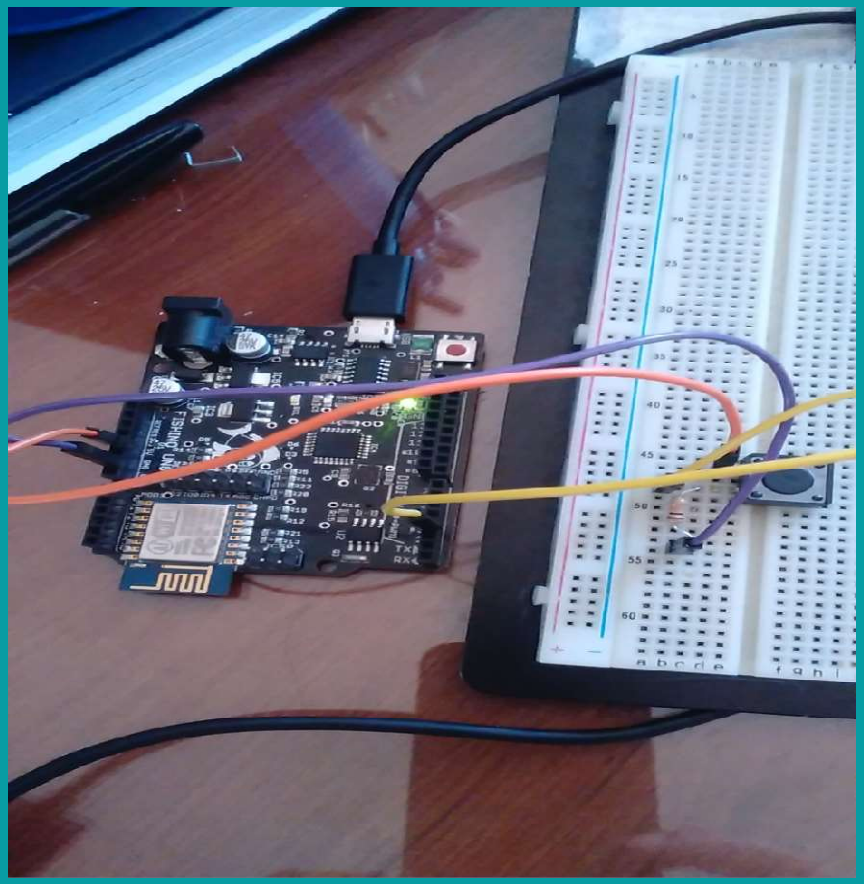
In uscita si avrà lo stato logico LOW senza premere il pulsante, premendo il pulsante si avrà lo stato logico HIGH

Input/output senza resistenza di pull up interna



Oppure





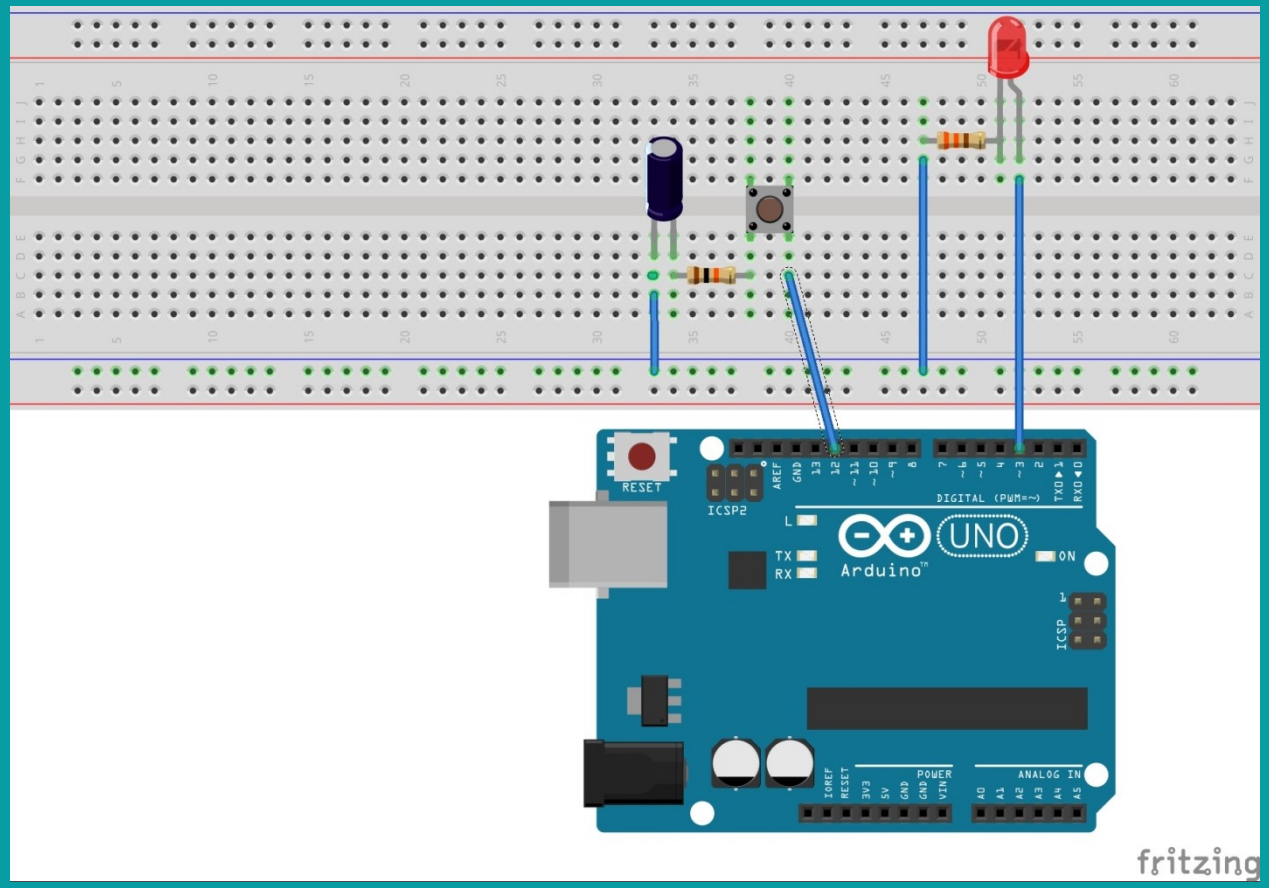
Interruttore con condensatore

Il condensatore è un accumulatore di cariche elettriche. Se si collega il condensatore in serie ad una resistenza e ad una batteria, esso si carica in un tempo paria a 5*R*C. R è il valore della resistenza e C è la capacità del condensatore. Nel circuito della slide 14, si può porre una capacità in serie alla resistenza sull'interruttore. In questo modo, sembrerà che l'interruttore sia posto a livello logico alto per un certo tempo e il led non si spenge subito. Naturalmente, si deve dare il tempo al condensatore di caricarsi, altrimenti, il led non resta acceso nel tempo dell'intera scarica del condensatore. Il programma è simile a quello della slide 13

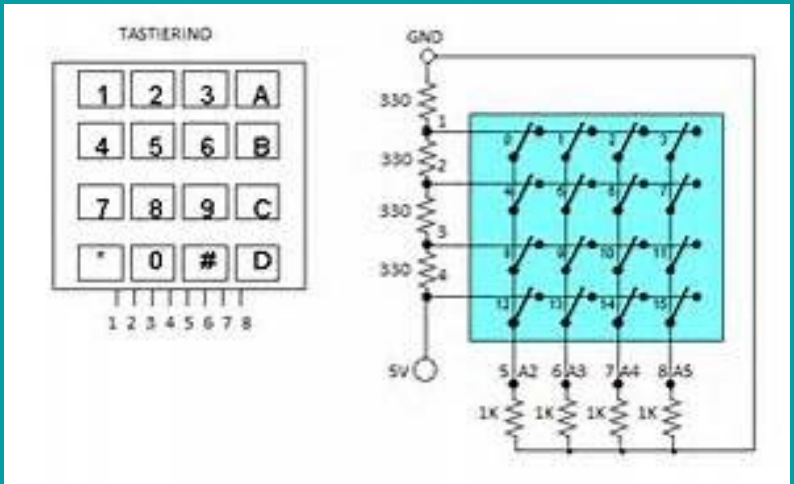
Programma interruttore e condensatore

```
const int led= 3;
const int pulsante=12;
int stato=0;
void setup() {
  pinMode(led,OUTPUT);
  pinMode(pulsante,INPUT);
}

void loop() {
  stato=digitalRead(pulsante);
  if(stato==HIGH)
  digitalWrite(led,HIGH);
  else  if(stato==LOW) digitalWrite(led,LOW);
}
```

Circuito con condensatore



Tastierino


TASTIERINO
1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D
1 2 3 4 5 6 7 8
GND
330
330
330
330
5V
5 A2 6 A3 7 A4 8 A5
1K 1K 1K 1K

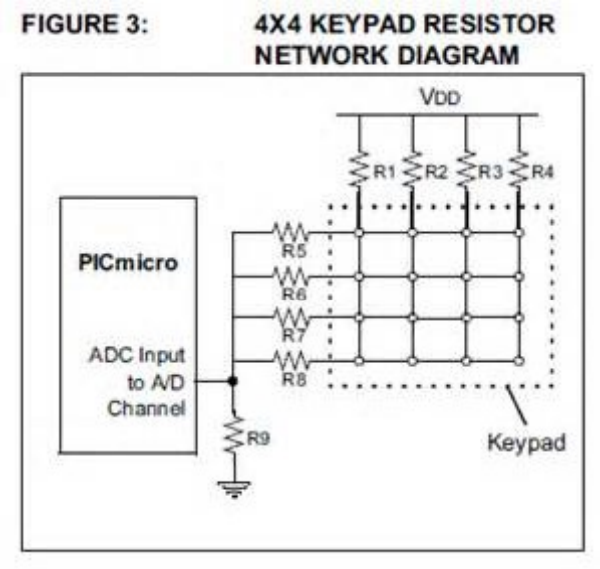


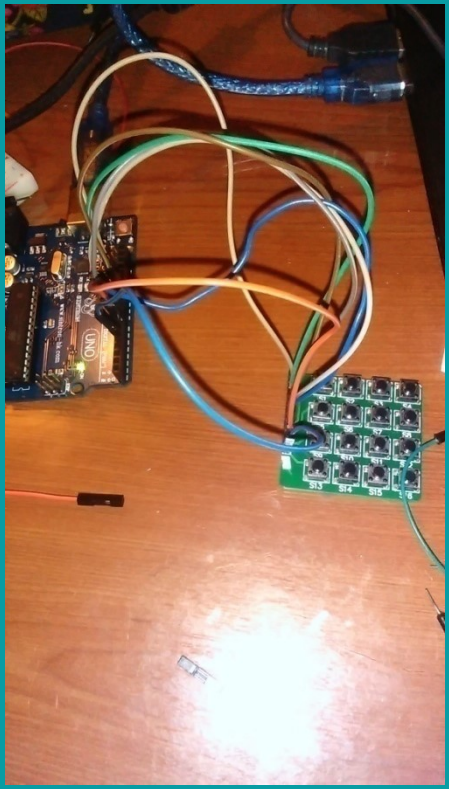
FIGURE 3:
4X4 KEYPAD RESISTOR NETWORK DIAGRAM
VDD
R1 R2 R3 R4
PICmicro
R5
R6
R7
R8
ADC Input to A/D Channel
R9
Keypad

Programma tastierino

```
 const char keymap[4][4]= {
{'1', '2', '3',':'} ,
{ '4', '5','6','-'},
{ '7','8', '9','?'},
{ '0','*','+','!'}};
int numCar=0;
int offset=0;
int tempTime=0;
int time;
const int rowPins[4]={5,6,7,8};
const int colPins[4]= {9,10,11,12};
char getKey() {
char key=0;
for(int column=0;column<4;column++)
{
digitalWrite(colPins[column],LOW);
for(int row=0;row<4; row++)
{
if(digitalRead(rowPins[row])==LOW)
{
delay(20);
while(digitalRead(rowPins[row])== LOW);
key= keymap[row][column];
}
}
digitalWrite(colPins[column],HIGH);
}
return key;}
```

```
void setup()
{ Serial.begin(9600);
for (int row=0;row<4;row++)
{
pinMode(rowPins[row],INPUT);
digitalWrite(rowPins[row],HIGH); //attiva pull-up
}
for (int column=0;column<4;column++)
{
pinMode(colPins[column],OUTPUT);
digitalWrite(colPins[column],HIGH); //colonne tutte
inattive
}
}
void loop() {
char key=getKey();
if(key != 0) {
Serial.println(key);
delay(20);
}}
```

tastierino

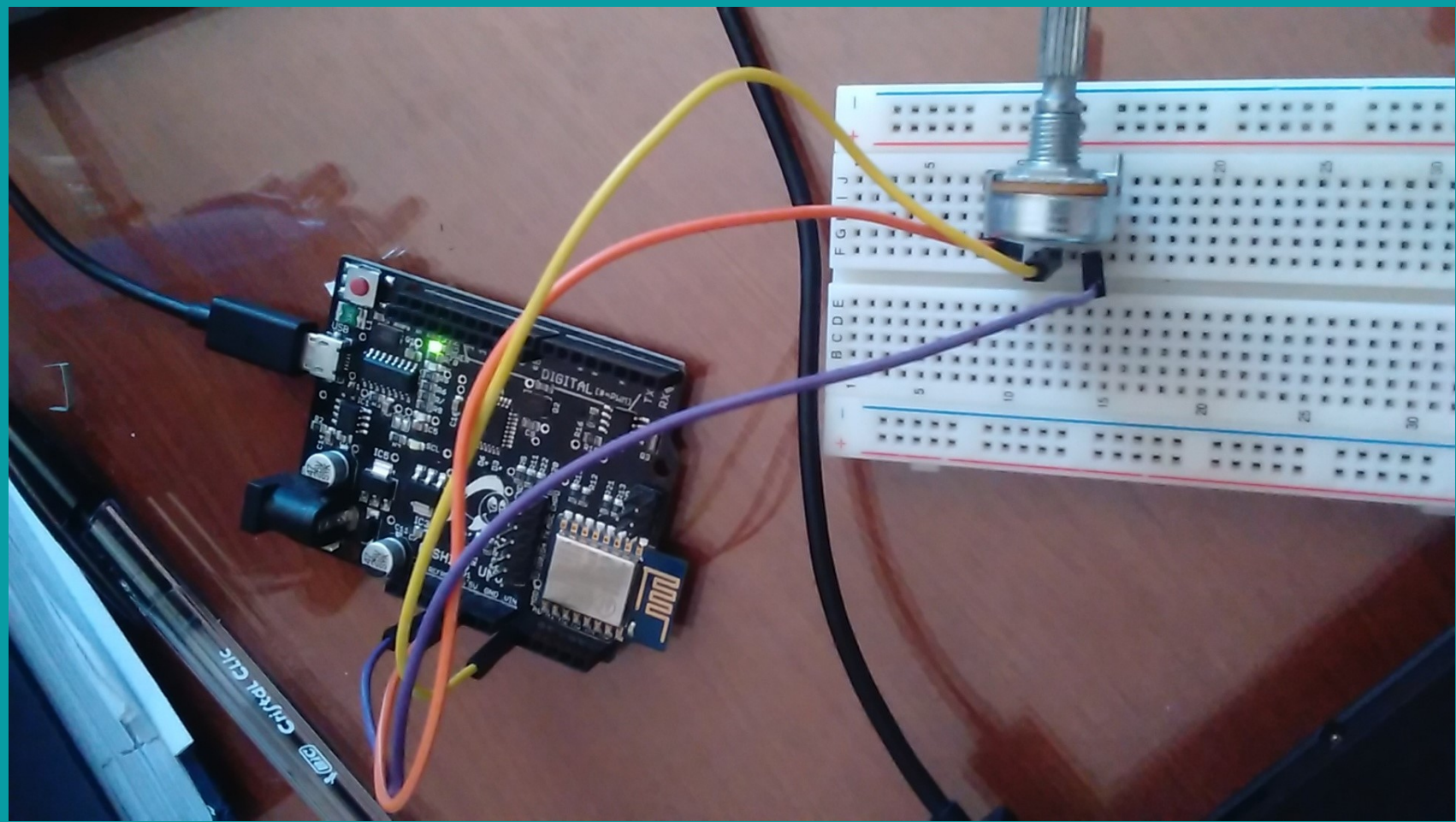


Leggere un dato analogico

•Un dato analogico potrebbe essere il valore di un potenziometro; il programma per leggere un valore analogico e riportarlo su un monitor seriale è il seguente:

```
int val=0;
void setup() {
  Serial.begin(9600);
}
void loop() {
   val=analogRead(0);
  Serial.println(val);
  delay(3000);
}
// Il potenziometro non è polare. Bisogna
//ricordarsi solo che il pin centrale è il sensore
//gli altri due laterali sono per GND e VCC
```

Circuito con potenziometro



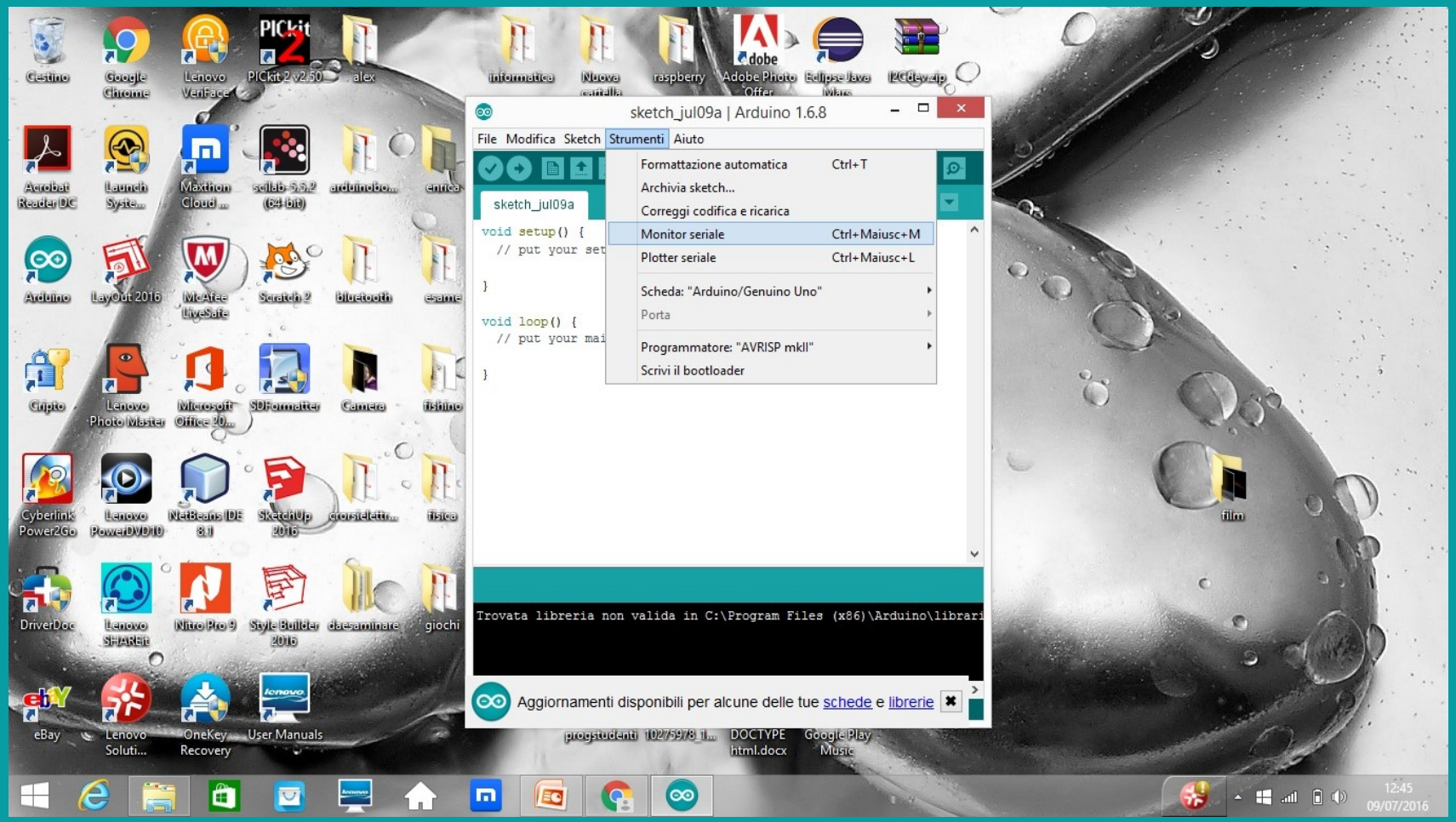
Comando serial

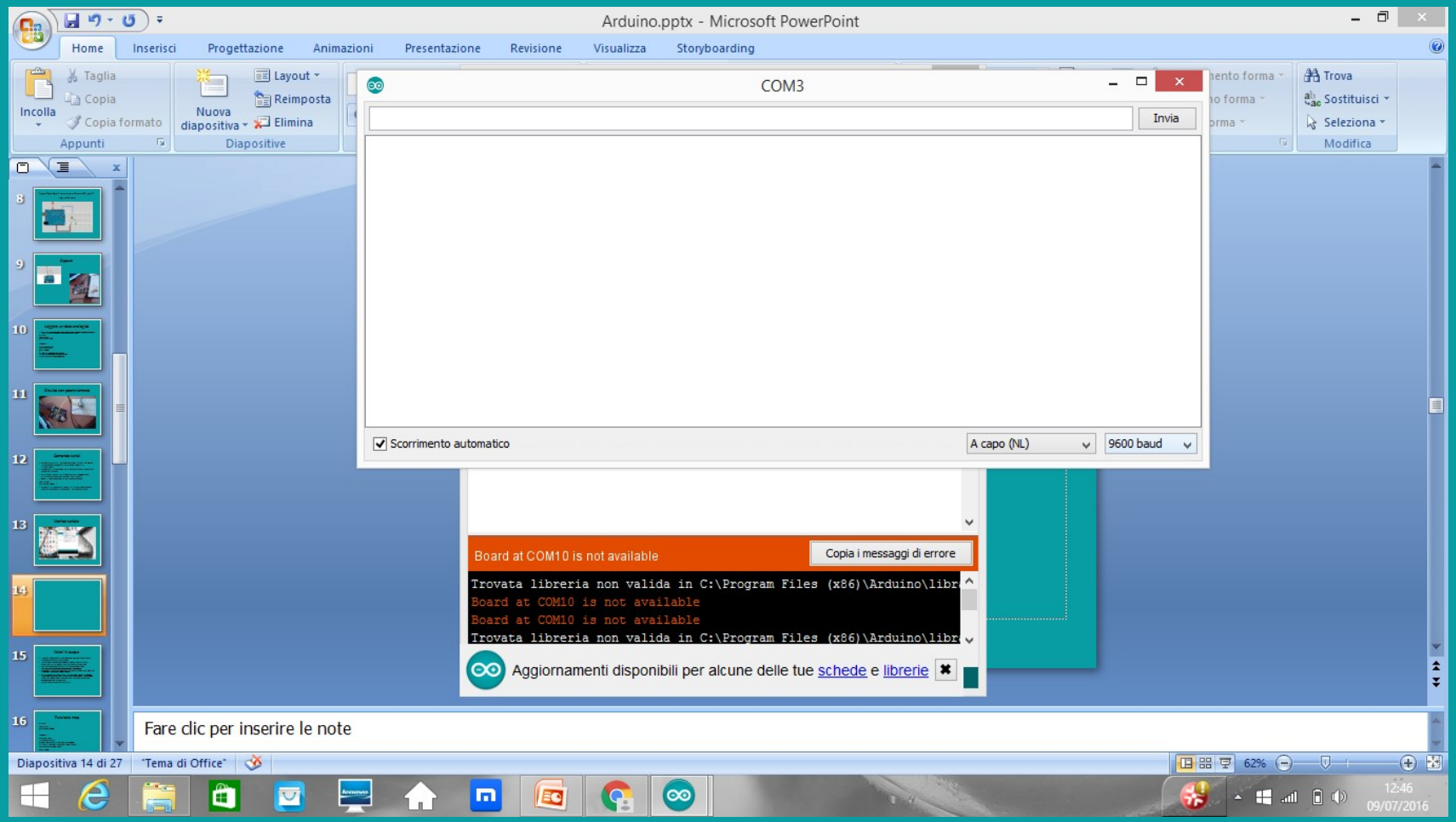
- Per poter inizializzare una comunicazione seriale con arduino, bisogna aggiungere il comando Serial.begin(velocità di trasmissione) nel blocco setup
- La trasmissione seriale avviene sia se si vuole trasmettere un dato tra arduino e il monitor seriale, che se si vuole comunicare con il modulo bluetooth
- La velocità di trasmissione dati si misura in baud, baud rate, simboli per secondo
- Per arduino il baud è settato per default a 9600 ma può essere anche cambiato a seconda delle esigenze
- Quindi il comando va posto nel seguente modo:

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

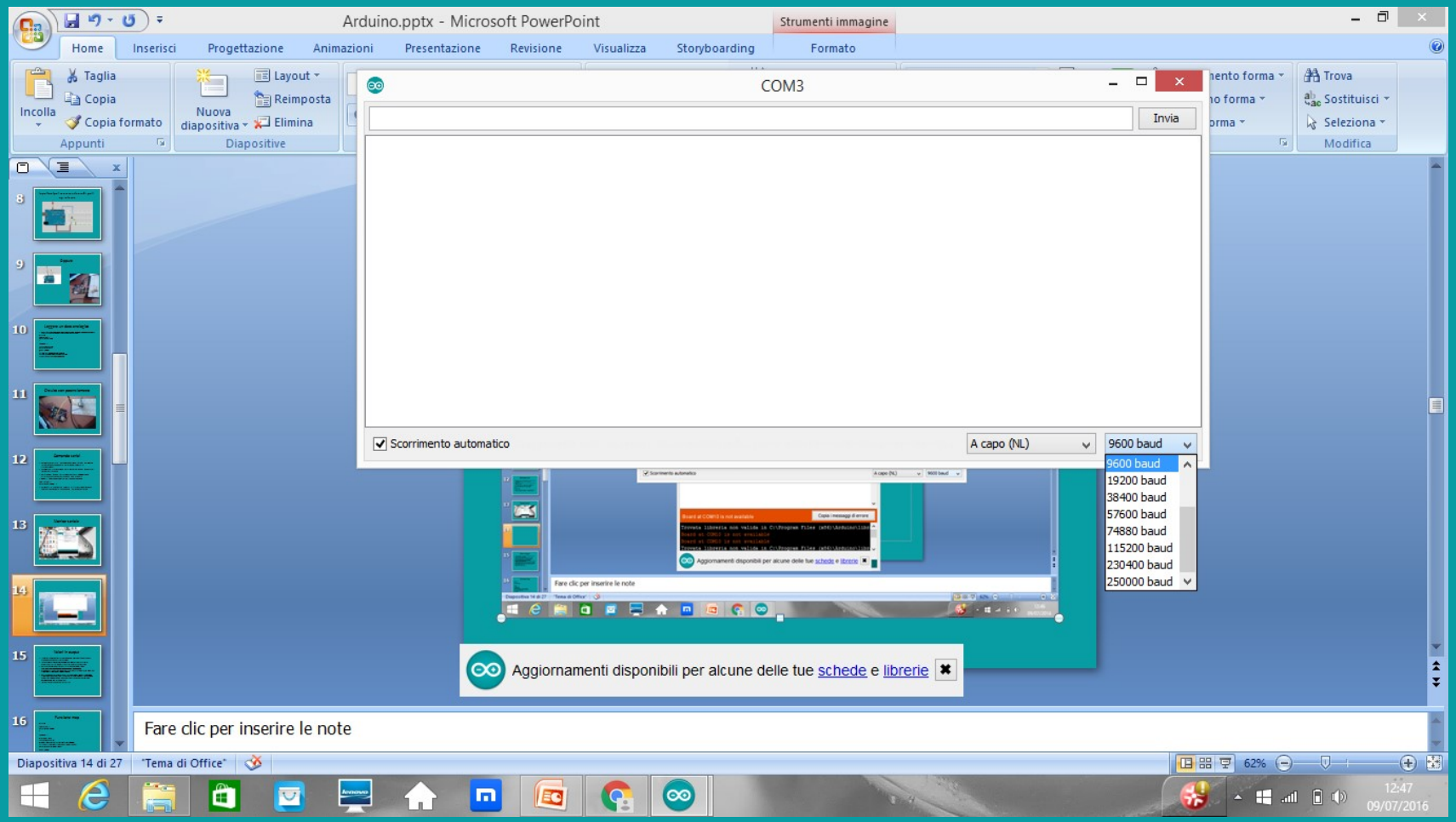
- Se si pensa di cambiare la velocità di trasmissione, bisogna ricordarsi di cambiare anche quella del monitor seriale

Monitor seriale




Arduino.pptx - Microsoft PowerPoint
Home
Inserisci
Progettazione
Animazioni
Presentazione
Revisione
Visualizza
Storyboarding
Incolla
Taglia
Copia
Copia formato
Appunti
Nuova diapositiva
Layout
Reimposta
Elimina
Diapositive
Trova
Sostituisci
Seleziona
Modifica
COM3
Invia
Scorrimento automatico
A capo (NL)
9600 baud
Board at COM10 is not available
Copia i messaggi di errore
Trovata libreria non valida in C:\Program Files (x86)\Arduino\libr
Board at COM10 is not available
Board at COM10 is not available
Trovata libreria non valida in C:\Program Files (x86)\Arduino\libr
Aggiornamenti disponibili per alcune delle tue schede e librerie
Fare clic per inserire le note
Diapositiva 14 di 27
"Tema di Office"
62%
12:46
09/07/2016

Monitor seriale



Comandi serial

- Serial.print(valore variabile)
  stampa il valore di una variabile senza andare daccapo
- Serial.println(valore variabile)
  stampa il valore di una variabile e va daccapo
- Serial.print("frase")
  stampa una frase senza andare daccapo
- Serial.println("frase")
  stampa una frase e va daccapo
- Serial.read();

Legge un parametro passato sul monitor seriale

Serial.read()

```
char valore_letto;
 void setup(){pinMode(13, OUTPUT);
 Serial.begin(9600);}
 void setup(){
 valore_letto=Serial.read();
 if(valore_letto==‘a’)
 digitalWrite(13,HIGH);
 if(valore_letto==‘b’)
 digitalWrite(13,LOW);}
```

Valori in output

- I valori letti sul monitor seriale sono da 0 1023 perché la scheda arduino ha un convertitore a 10 bit
- In realtà il massimo valore del potenziometro è scritto sul dispositivo stesso. Nel nostro caso, il valore va da 0 k a 10 K
- Per avere il valore giusto, basta fare una proporzione:
valore_min_letto:valore_max_letto=valore_min_reale:valore_max_reale
- Arduino utilizza una funzione che fa automaticamente ciò: **map(val, from low, from high, to low, to high)**
- Val sta ad indicare il valore su cui fare la proporzione, from low e from high, indicano rispettivamente, il valore più basso e quello più alto letto, to low e to high indicano rispettivamente il valore più piccolo e quello più grande reali.
- Il programma è nella slide successiva

Funzione map

```
int val=0;
void setup() {
  Serial.begin(9600);
}

void loop() {
 int proporzione;
   val=analogRead(0);
   proporzione=map(val, 0,1023,0, 10000);
   //(value, from low, from high, to low, to high)
  Serial.println(proporzione);
  delay(3000);
}
```

Debounce

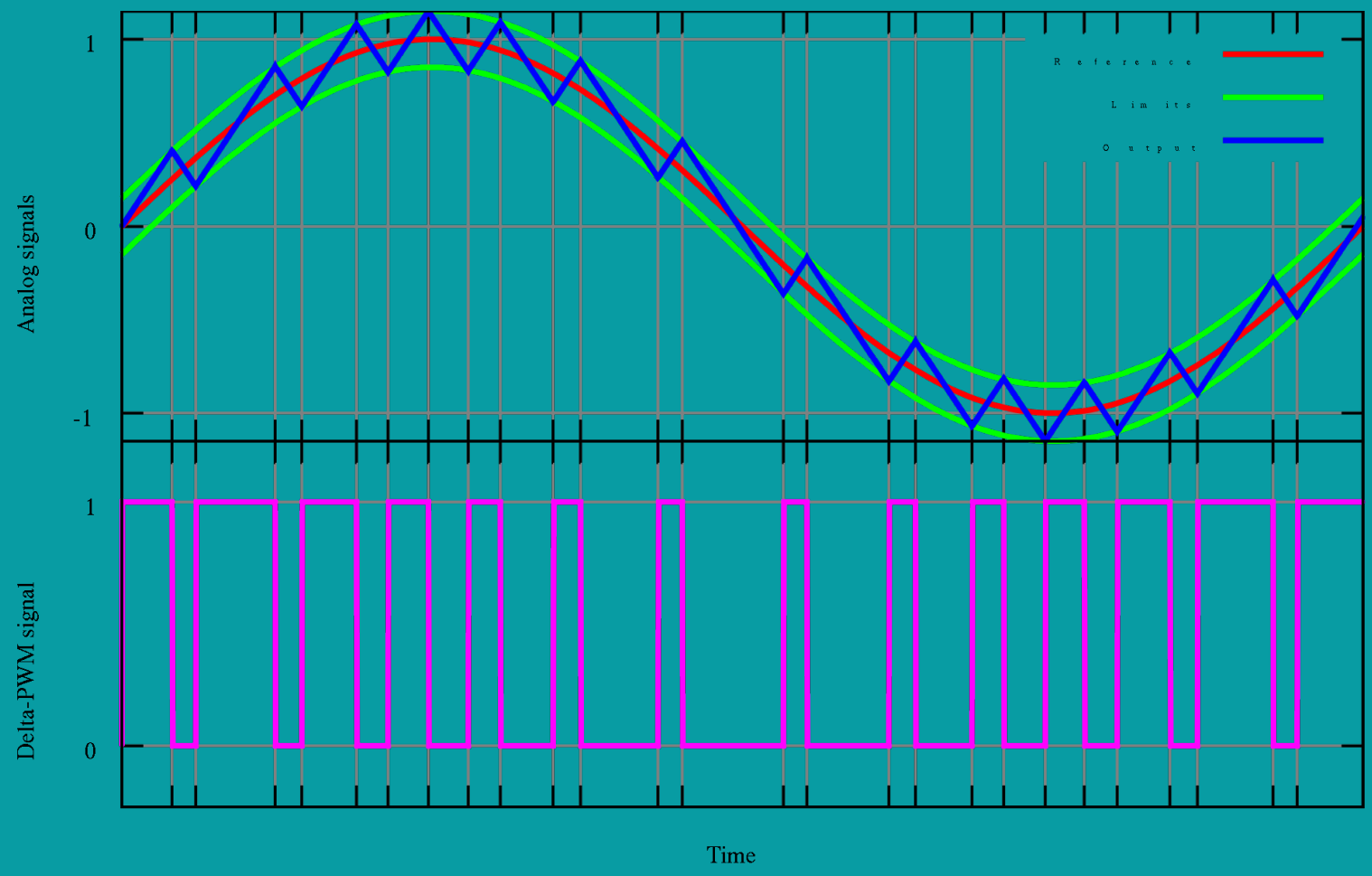
- Il problema del rimbalzo è presente in tutti i circuiti elettronici quando bisogna cambiare stato. Infatti, quando si preme un interruttore per cambiare uno stato, succede che il valore non viene letto immediatamente o viene letto in maniera sbagliata a causa dei rimbalzi meccanici.
- I contatti infatti, si aprono e si chiudono nel giro di pochi millisecondi fino ad assumere una situazione stabile. Ciò però può portare ad un errore nella lettura dei comandi se si pigia frettolosamente il tasto di comando e poi lo si rilascia.
- Anche per la scheda Arduino può sussistere questo problema
- Il seguente programma risolve il problema del rimbalzo

Debounce

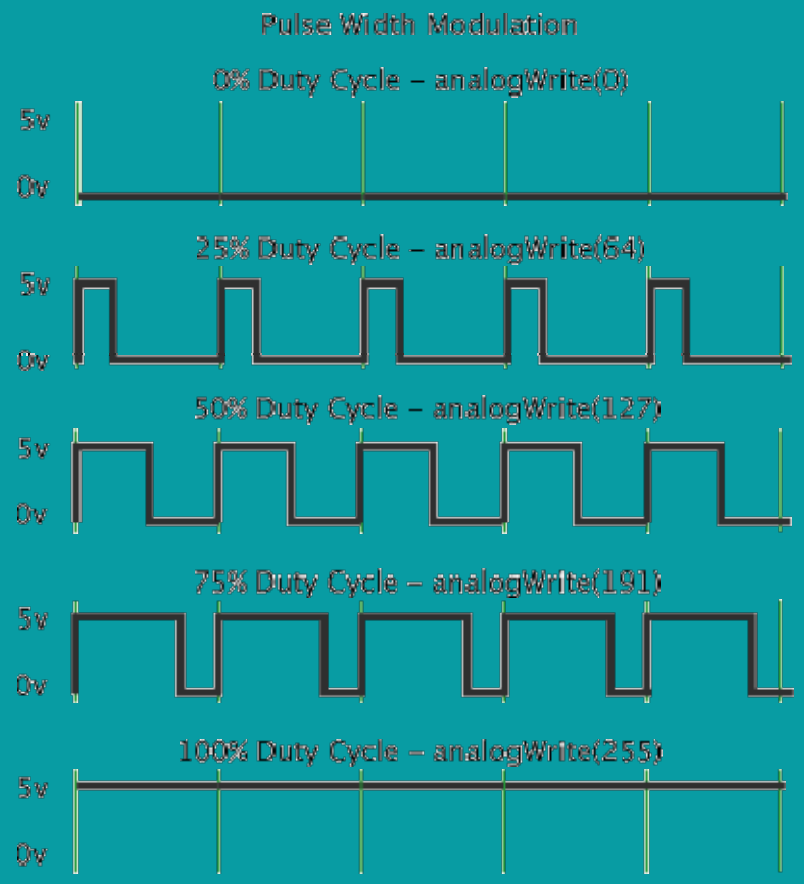
```
#define debounce 200  //Tempo di rimbalzo
const int pulsante= 2;
const int led=13;
int state = HIGH;
int lettura;
int previous = LOW;  //Lettura precedente del pin di input state
long time = 0;
void setup()
{
    pinMode(pulsante, INPUT);
    pinMode(led, OUTPUT);
}
void loop()
{
    lettura= digitalRead(pulsante);
    if (lettura == HIGH && previous == LOW && millis() - time > debounce) {
       //Inverte l'otput
       if (state == HIGH)
          state = LOW;
       else
          state = HIGH;
       time = millis();
    }
 digitalWrite(led, state);

 previous = lettura;
}
```

- Il circuito è quello senza resistenza di pull up
- La funzione mills() legge quanti millisecondi resta il tasto premuto
- Se il tempo misurato è maggiore di quello di rimbalzo, si inverte lo stato

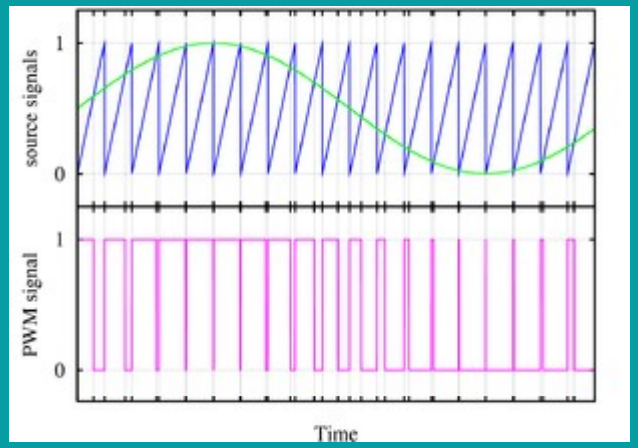
PWM


Reference
Limits
Output
1
0
-1
Analog signals
1
0
Delta-PWM signal
Time

PWM


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

PWM



La **modulazione di larghezza di impulso** (o **PWM**, acronimo del corrispettivo inglese *pulse-width modulation*), è un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell'impulso positivo e di quello negativo (duty-cycle), allo stesso modo è utilizzato per protocolli di comunicazione in cui l'informazione è codificata sotto forma di durata nel tempo di ciascun impulso.

Con la tecnica PWM si può variare la tensione in uscita attraverso dei semplici comandi

Arduino e la PWM

•Arduino uno ha dei pin digitali dedicati alla PWM. I pin sono: 3,5,6,9,10, 11. Alcuni sono caratterizzati da una frequenza differente, basta guardare la seguente tabella:

Timer	Frequenza Base	Output Compare		Pin Arduino	Frequenza Default
TCCR0	62500Hz	OC0	OC0A	6	976Hz
	62500Hz		OC0B	5	976Hz
TCCR1	31250Hz	OC1	OC1A	9	488Hz
	31250Hz		OC0B	10	488Hz
TCCR2	31250Hz	OC2	OC1A	11	488Hz
	31250Hz		OC2B	3	488Hz

Primo esempio di PWM

- Il valore della PWM con arduino può variare tra 0 e 255.
- Un semplice programma è il seguente:

```
 void setup(){
 pinMode(3, OUTPUT);}
 void loop(){
analogWrite(3,255);
delay(3000);
 analogWrite(3, 125);
 delay(3000);
 analogWrite(3,0);
 delay(3000);}
```

Questo programma fa accendere un led sul pin 3 partendo da una luminosità massima ad una intermedia fino a spegnersi. Il comando è analogWrite anche se si lavora sul pin digitale. Il primo parametro è il pin di riferimento e il secondo il valore della PWM. Al posto di un led si può porre anche motorino e far cambiare così la velocità.

PWM variabile

- Si può far variare la PWM dall'esterno tramite il monitor seriale o tramite un tastierino numerico o tramite potenziometro
- Per cambiare la PWM automaticamente in sequenza di valori interi, si può utilizzare un ciclo for

PWM con ciclo for

```
int i ;
void setup(){
pinMode(3,OUTPUT);}
void loop(){ for(i=0;i<255;i++)
analogWrite(3,i);}
```

//un programma così fa variare lentamente la
//PWM. Gli incrementi di PWM possono
//essere cambiati a seconda delle esigenza

PWM con ciclo for

```
int i ;
 void setup(){
pinMode(3,OUTPUT);
Serial.begin(9600);}
 void loop(){ for(i=0;i<255;i+=10){
 analogWrite(3,i);
 Serial.println(i);
 delay(3000);}}
```

PWM pilotata da un potenziometro

```
void setup() {
  pinMode(3, OUTPUT);
}

void loop() {
  int res=analogRead(5);
  int val=map(res,0,1024,0,255);
  digitalWrite(3,val);
}
```

Interrupt

- L’interrupt è una interruzione che viene generata e controllata tramite software
- Un qualsiasi microcontrollore quando esegue un loop, si blocca solo se c’è un comando particolare che lo fa andare in un’altra routine
- Sulla scheda Arduino l’interrupt viene generato da un cambio di livello sul piedino 2 o sul piedino 3
- La scheda arduino può generare fino a 26 interrupt
- Es: un semaforo segnala rosso; si può interrompere tale segnalazione tramite un pulsante
- Il segnale rosso può essere un led acceso su un piedino di arduino; l’interruzione può essere causata da un pulsante posto sul piedino 2 o sul piedino 3 che fa cambiare di livello e genera l’interruzione
- Dall’esempio nella slide successiva si possono dedurre le regole base

Esempio

```
int pin = 8;
volatile int state = LOW;  //per l'interrupt è utile questa dichiarazione
void setup()
pinMode(pin, OUTPUT);
digitalWrite(pin, LOW);
attachInterrupt(1, g_intr, LOW); //la funzione attachInterrupt è quella che genera //l'interrupt; il parametro 1 indica che il piedino
sul quale è generato è il 3. Il secondo parametro è la funzione  //nella  quale si entra quando si genera un interrupt. Il terzo
parametro indica l'vento che scatena //l'interrupt
}
void loop()
{
//fa qualcosa…
delay(100);
}
void g_intr()
{
state = !state;
digitalWrite(pin, state);
}
//è stato scritto in giallo tutto ciò che riguarda la funzione interrupt
```

Funzioni Interrupt

- Le funzioni interrupt sono le seguenti:
  - la funzione denominata "interrupts();" serve per abilitare l'interrupt globale.
  - la funzione denominata "noInterrupts();" serve per disabilitare l'interrupt globale.
  - la funzione denominata "attachInterrupt(interrupt, function, mode);" serve per collegare l'interrupt alla tabella dei vettori dell'interrupt.
  - la funzione denominata "detachInterrupt(interrupt);" serve per disabilitare l'interrupt specificato.

Parametri funzione attachInterrupt

- attachInterrupt(interrupt, function,mode)
  - Interrupt →Specifica il numero dell'interrupt. Valori possibili sono 0 se l'evento si genera sul pin 2 oppure 1 se l'evento si genera sul pin 3.
  - Function→ specifica il nome della routine di gestione dell'interrupt.
  - Mode → Specifica quale tipo di attività deve essere valutata quando si verifica l'evento che genera l'interrupt.

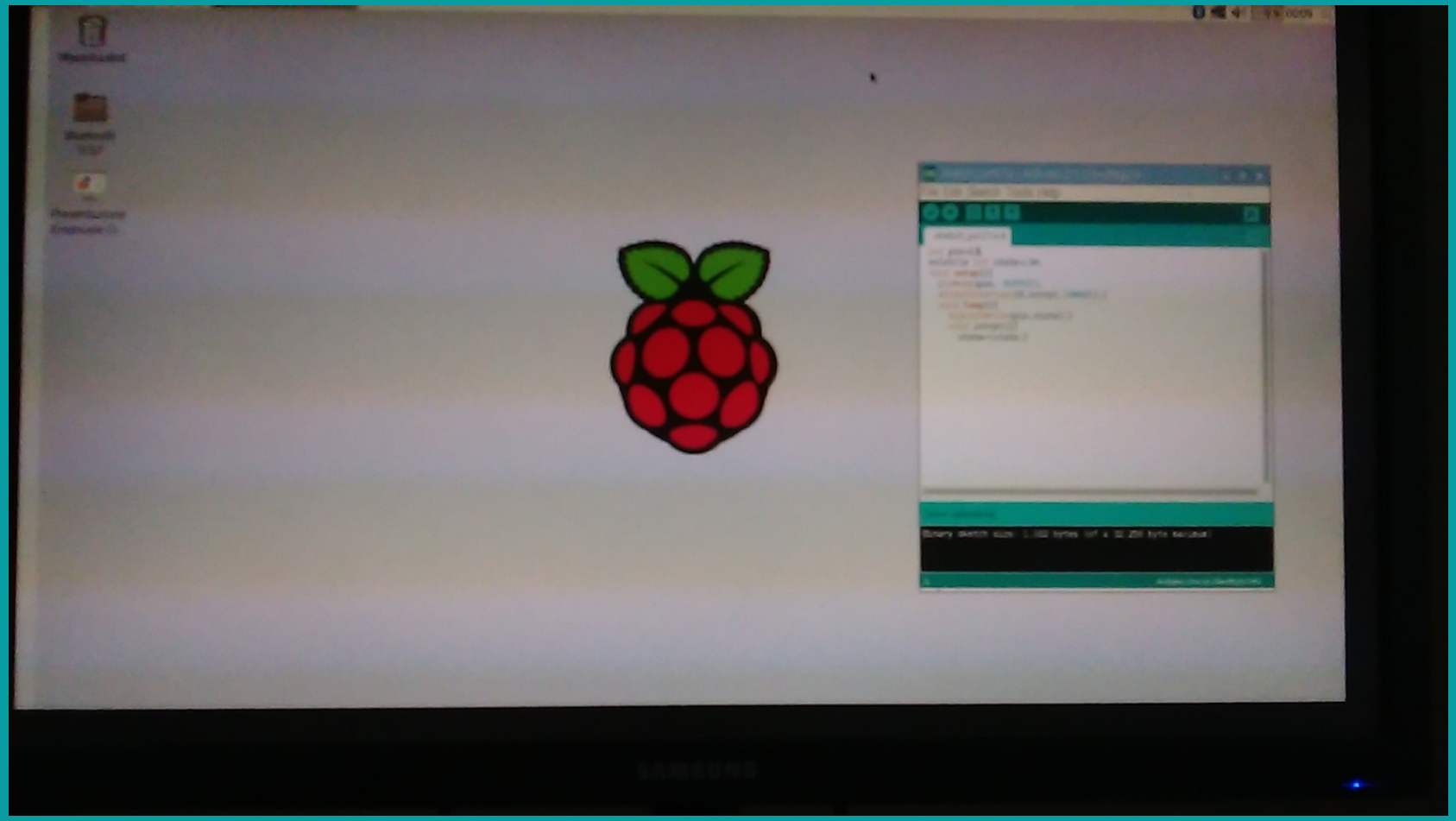
Modalità di interrupt

- Nella modalità denominata "LOW" si genera un interrupt quando il pin è a livello basso.
- Nella modalità denominata "CHANGE" si genera un interrupt quando il pin passa da un livello all'altro, cioè quando si passa da HIGH a LOW e viceversa.
- Nella modalità denominata "RISING" si genera un interrupt solo quando il pin passa dal livello LOW al livello HIGH.
- Nella modalità denominata "FALLING" si genera un interrupt solo quando il pin passa dal livello HIGH al livello LOW.

Esempio

```
int pin=13;
volatile int state=LOW;
void setup(){
  pinMode(pin, OUTPUT);
  attachInterrupt(0,intrpt,CHANGE);}
  void loop(){
    digitalWrite(pin,state);}
    void intrpt(){
      state=!state;}
```

Arduino e Raspberry…..amici per la USB



Arduino e Raspberry…amici per la USB

Arduino può essere programmato
tramite Raspberry sulla porta USB.