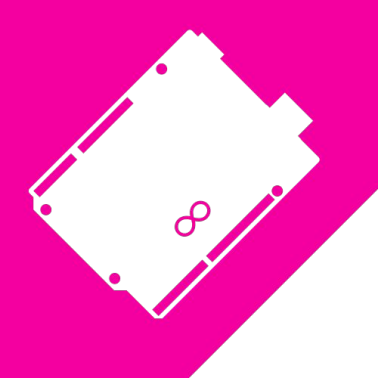
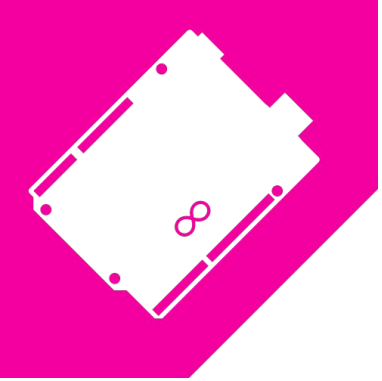


Lezione 3

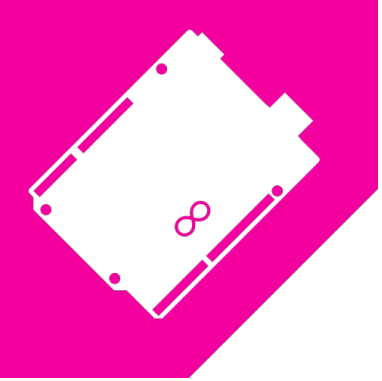
Un corso gentilemente offerto con il sudore e le lacrime di MugRomaTre e Roma Tre

e Magliana



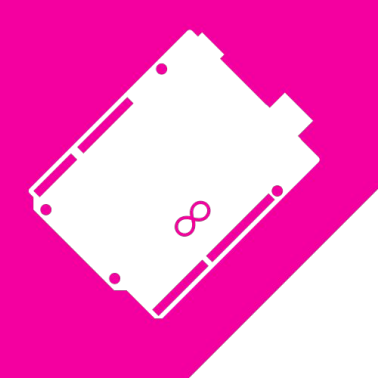
Grazie

- shootout a chi ha linkato sulla pagina del mug https://123d.circuits.io
  - per simulare circuiti, contiene anche una arduino uno programmabile



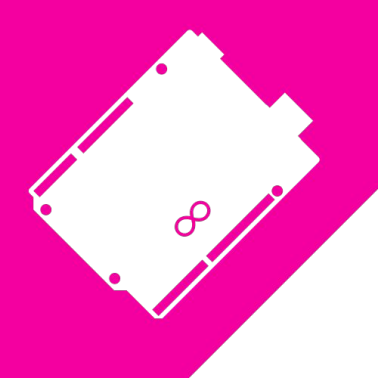
Oggi parliamo di

- Protocolli hardware di comunicazione
- Internet of things
- Il braccio di leonardo
  - ha fatto palestra ultimamente
- Esercizi vecchi e nuovi per arduino



Protocolli di comunicazione Hardware – un accenno

- accendere e spegnere pin è bello, ma a volte bisogna trasmettere byte
  - sensori più complessi, orologi, altre arduino...
- 3 famiglie di protocolli supportati da arduino
  - Serial
    - detta anche usart, uart
    - comunicazione alla pari
  - SPI
    - un master, uno slave
  - i2c
    - chiamata anche TWI
    - un master, più slave

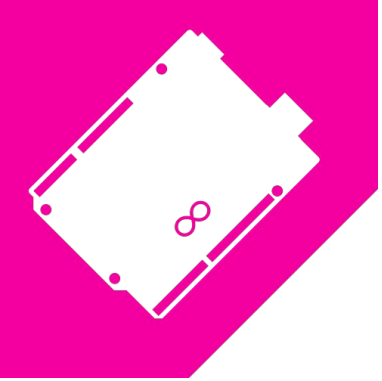


Serial

- già usato negli esempi
  - Serial.begin, Serial.println …
  - https://www.arduino.cc/en/Reference/Serial
  - 3 fili: RX → TX, TX → RX, GND → GND
- gli interlocutori decidono a priori la velocità di lettura e scrittura
  - 90% se non funziona è perché usano velocità differenti
- ciascuno scrive e legge quando vuole
  - attenzione che potrei saturare il mio buffer
- utilizzata per comunicare con gli umani, o come interfaccia semplice per moduli bluetooth, xbee...

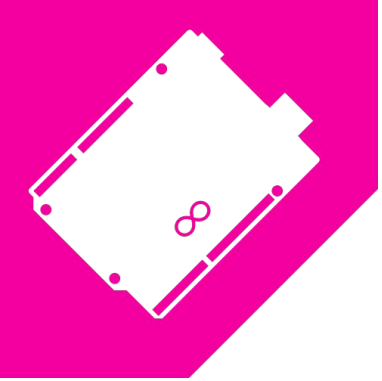
SPI

- https://www.arduino.cc/en/Reference/SPI
- 4 (+1 connessioni) MOSI → MOSI, MISO→MISO, CLK→CLK, GND→GND
  - opzionale: per comunicare con più slave, da arduino un pin SS (slave select) è connesso ad uno slave per selezionarlo. Ogni slave è connesso ad uno SS differente
- Il Master inizia la comunicazione, e trasmette dei byte
  - Se lo slave è un sensore, il master dovrà “chiedergli” di inviare i dati
- Molto veloce, usato dove è necessario tramsettere molti byte



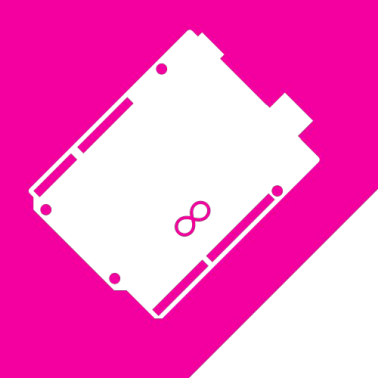
i2c - twi

- https://www.arduino.cc/en/Reference/Wire
- 3 connessioni: SDA→SDA*, SCL→SCL*, GND→GND*
- un Master, più slave. Ogni slave ha un indirizzo
  - il master deve richiedere agli slave di parlare
- Bassa Velocità, usato per sensori che devono trasmettere poco
  - Ma una arduino può essere configurata anche come slave



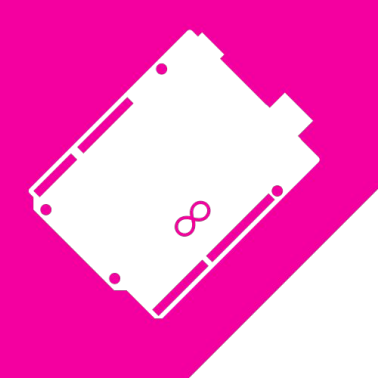
Quale dei 3?

- Dipende da cosa parla chi sta dall'altra parte della comunicazione
  - Molti componenti parlano più protocolli. alcuni hanno una uscita analogica ed una connessione spi o i2c per comunicazioni avanzate
- i protocolli sono semi-standard, è importante leggere il datasheet
  - nel datasheet ci stanno i messaggi che si possono inviare
  - se il componente è stato fatto per arduino, ci sarà una libreria



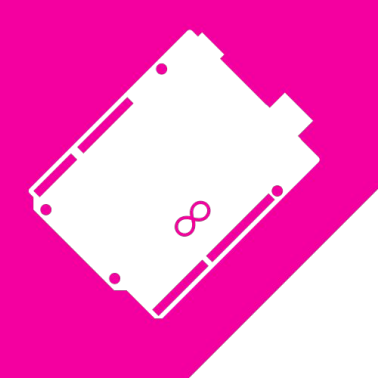
IOT

- Internet of thing – letteralmente internet delle cose
  - si, non significa assolutamente niente
    - e significa tutto allo stesso tempo
- L'idea è di avere oggetti “aumentati” con la capacità di comunicare in rete
  - per esempio una rete di sensori remota, o strumenti di telepresenza



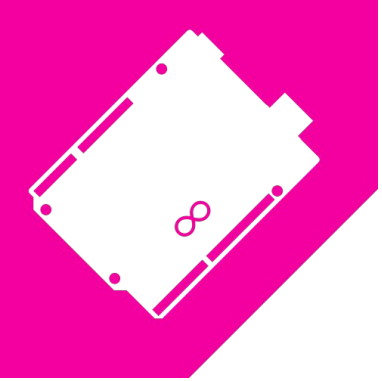
IOT – chi può farlo?

- non essendo uno standard, ma solo una filosofia, chiunque può fare la sua cosa
  - Arduino Yun, Temboo, Proton, MQTT…
- non è necessario neanche avere una cosa troppo complicata – dopo faremo una demo
- Di solito se nel tuo progetto internet è usato in qualche maniera, potresti essere considerato IOT



IOT – cose serie
MQTT

- https://en.wikipedia.org/wiki/MQTT
- http://pubsubclient.knolleary.net/
- é un protocollo per scambio di messaggi da usare via internet
  - molto leggero, molto elastico
  - Serve una macchina che fa da server
  - permette di scoprire altre risorse nella stessa rete
  - usato spesso per sensori remoti



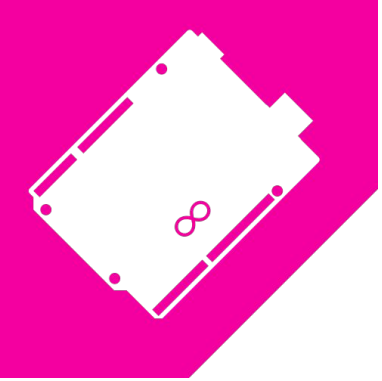
IOT – Arduino

Arduino ha molti modi per connettersi ad internet, ma di solito il processo è sempre lo stesso: lo sketch che scriviamo comunica con un modulo intermediario che si preoccupa di comunicare con la rete. alcune opzioni:

- Arduino Yun: una arduino leonardo unita ad un router wifi, la arduino è L'utente del Router
- ESP8266: un modulo poco costoso che funziona da scheda wifi per arduino. Può sostituire completamente l'arduino
- moduli bluetooth vari: costano poco e hanno più utilizzi, servono per connettersi ad un device (lo smartphone per esempio)

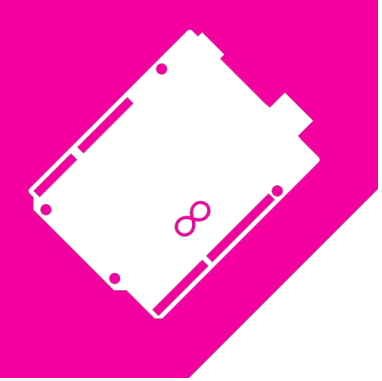
Arduino: come ci connettiamo?

- Cavo Usb:
  - Pro
    - L'alimentazione passa per il pc
    - Funziona tramite il seriale quindi la trasmissione è veloce
    - Non c'è bisogno di moduli aggiuntivi per l'arduino.
  - Contro:
    - Non portabile.
    - Necessita di un pc acceso con un programma in esecuzione che legge e spara i dati.
- demo alla fine



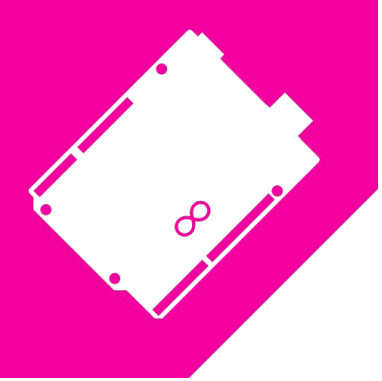
Arduino: come ci connettiamo?

- Modulo Bluetooth:
  - Pro:
    - Facilissimo da montare
    - Funziona tramite seriale quindi la trasmissione è veloce
    - Portabile
    - Possibilità di collegare il dispositivo ad uno smartphone.
    - Il modulo bluetooth costa pochissimo.
  - Contro:
    - Ha bisogno di uno smartphone vicino per inviare dati su internet.
    - Anche se poco costoso bisogna comunque aggiungere il modulo bluetooth
    - Piace un sacco a Rosati



Arduino: come ci connettiamo?

- Soluzione wi-fi:
  - Pro:
    - Il dispositivo può connettersi ad internet in maniera totalmente indipendente.
  - Contro:
    - Collegarsi con una rete wifi è molto lento e non si può eseguire in parallelo.
    - Gestire il codice lato arduino della comunicazione wifi non è semplicissimo
    - Solitamente per connettersi ad un wi-fi bisogna inserire una password, salvo non la si cabli nel codice non è un'operazione banale.
    - Il wi-fi è un modulo in più da aggiungere.
    - Necessita comunque di un access point per connettersi.
    - Non portabile quanto il bluetooth

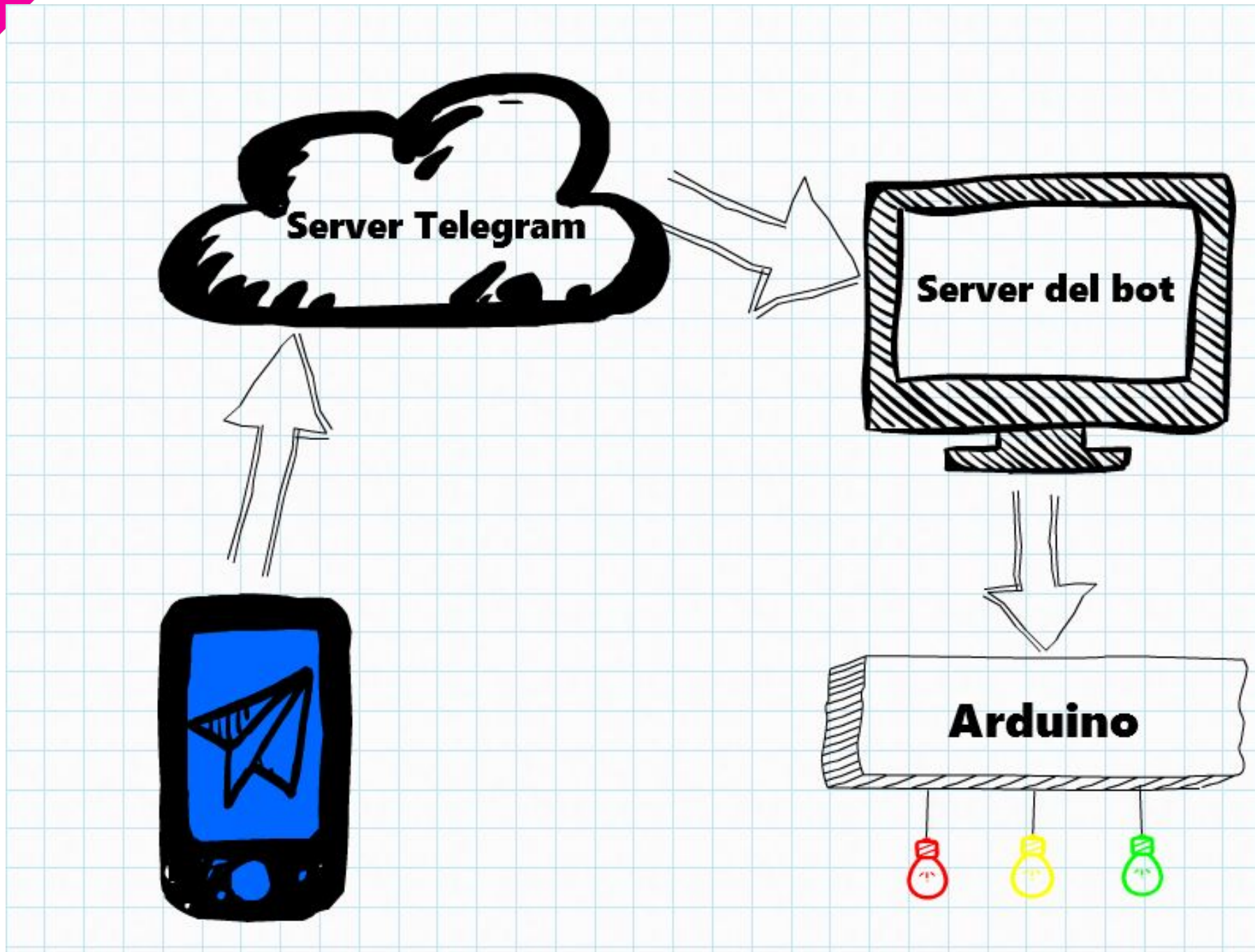


Demo

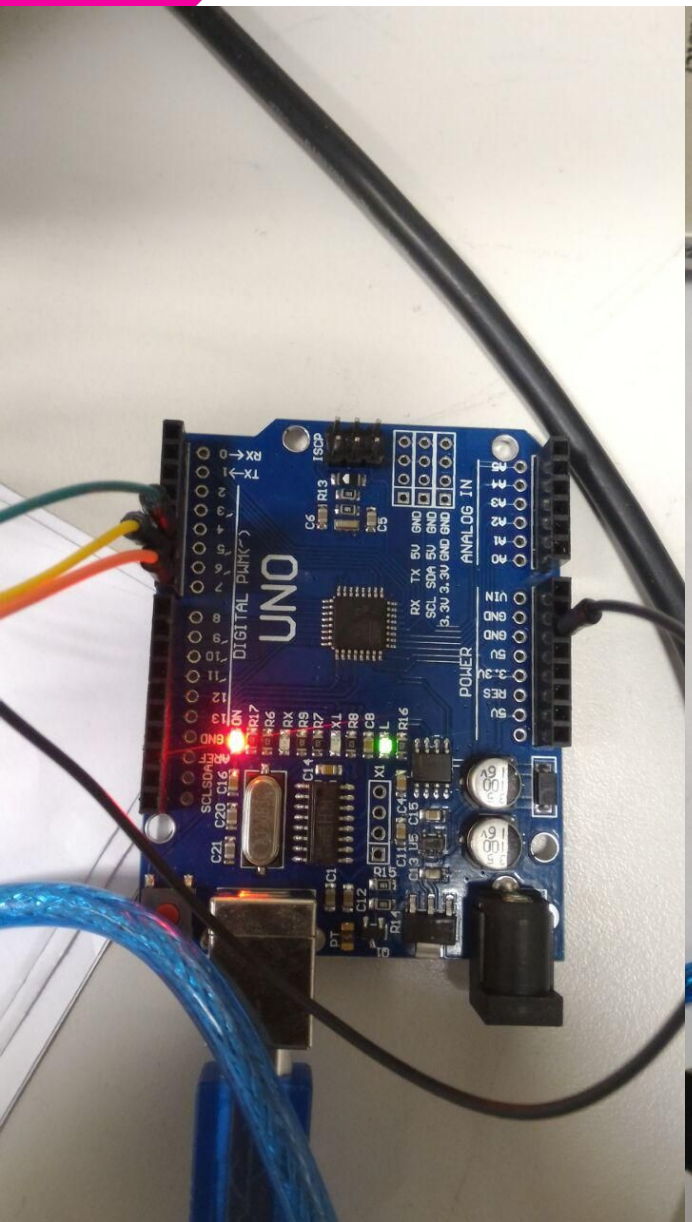
- Usiamo un bot Telegram per controllare dei led sulla arduino
  - Telegram è una applicazione di chat che permette di essere usata con dei bot
  - piace molto ad Andrea Rosati
  - https://telegram.me/mugroma3
    - è il cananle telegram del mug
- il codice è disponibile online
  - https://github.com/mugroma3/EsempioArduinoTelegram

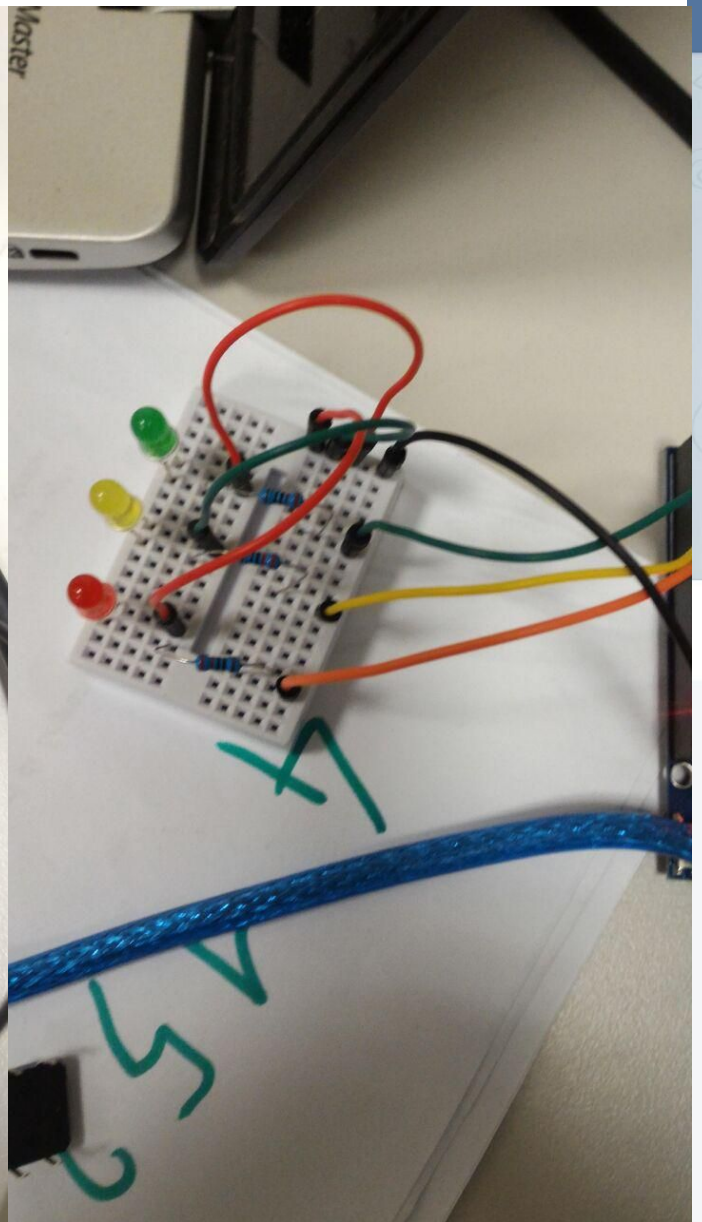
    - http://dummyscodes.blogspot.it/2014/08/using-java-rxtx-library-for-serial.html
    - http://www.massimobiagioli.it/blog/arduino-java-serial-rxtx-it-works/
    - http://jaegerbox.net/botticelli-index/

Architettura



Demo


UNO
DIGITAL PWM(~)
ANALOG IN
POWER



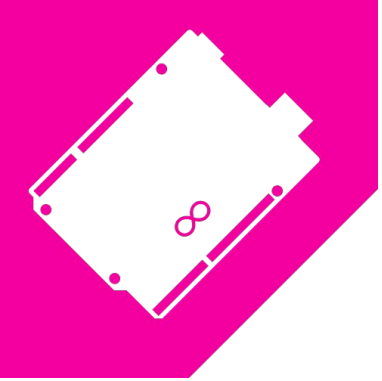

91%
07:23
MugLabBot
bot
Yellow 18:02
Yellow 18:02
Red 18:29
Red 18:29
Yellow 18:31
Yellow 18:31
Messaggi
Red
Yellow
Green

Ospiti di Oggi

- Leonardo Franco
- Il suo braccio bionico

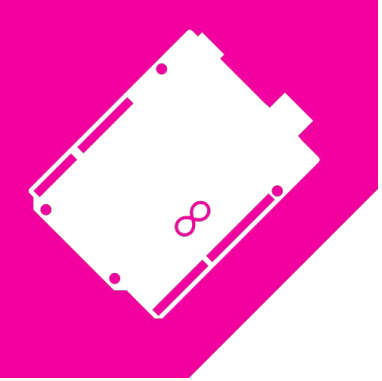
Idea per chi è riuscito a finire gli esercizi dell'ultima volta:

https://www.youtube.com/watch?v=8AKZk2ldPWs



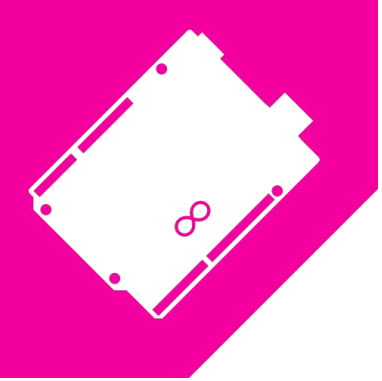
Cosa proviamo oggi? - 1

- Blink
- SerialAnalogRead
- Knob
- Wave
- PhotoServo
- PhotoServoClock



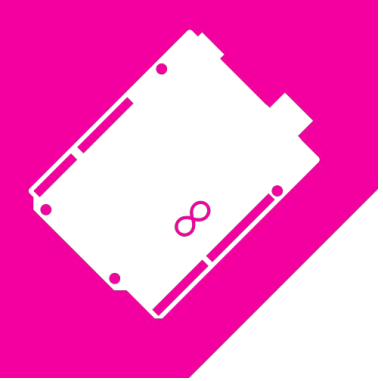
Blink

- Voglio accendere e spegnere il led sul pin 13, acceso per 700ms e spento per 350ms
- Hint: delay(ms) aspetta ms millisecondi
- Hint: non devo aggiungere nessun led, perché ce n'è uno già sulla scheda arduino



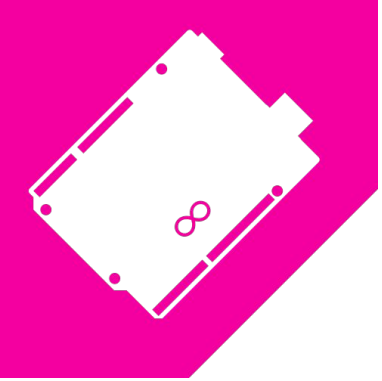
Blink Soluzione

```
/*
 * the setup function runs once when you press
 *  reset or power the board
 */

void setup() {
  pinMode(13, OUTPUT);
          // initialize digital pin 13 as an output.
}

          // the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);
          // turn the LED on (HIGH is the voltage level)
  delay(700);
          // wait for 700 ms
  digitalWrite(13, LOW);
          // turn the LED off by making the voltage LOW
  delay(350);
          // wait for 350 ms
}
```



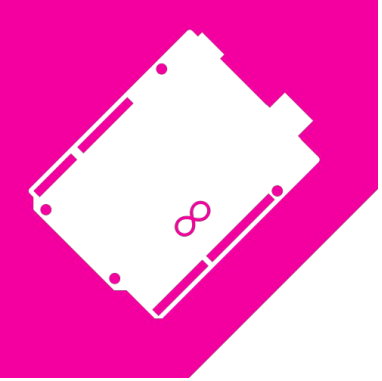
SerialAnalogRead

- Voglio stampare sul monitor seriale il voltaggio letto sul pin A0
- Ricordati di aprire il monitor seriale!
  - puoi anche aprire il plotter seriale per vedere
- Sul pin A0 potrei mettere il trimmer o il la fotoresistenza
  - come spiegato nelle slide della lezione precendente
- Hint: analogRead(pin) tutta la vita
- Hint: Serial.println(val)



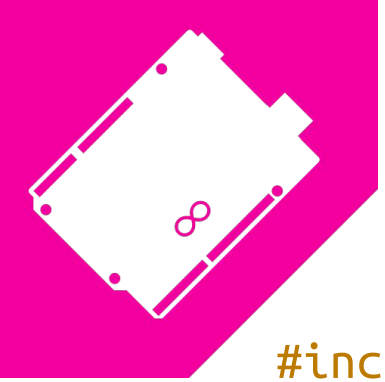
SerialAnalogRead Soluzione

```
void setup() {
  Serial.begin(9600);
            // initialize the serial communication:
            // remember to select this same velocity on the serial monitor
}

void loop() {
            // send the value of analog input 0:
  Serial.println(analogRead(A0));
            // wait a bit for the analog-to-digital converter
            // to stabilize after the last reading:
  delay(2);
}
```



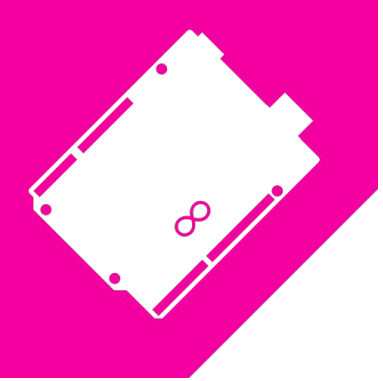
Knob

- Voglio controllare la posizione del servo con un potenziometro
- Servo: marrone → gnd, rosso→ 5V, arancione → pin 9
- Hint: int res= map(value, fromLow, fromHigh, toLow, toHigh) scala value da un range ad un altro.
  - y = map(x, 0, 1023, 0, 180); scala x da 1023 a 180



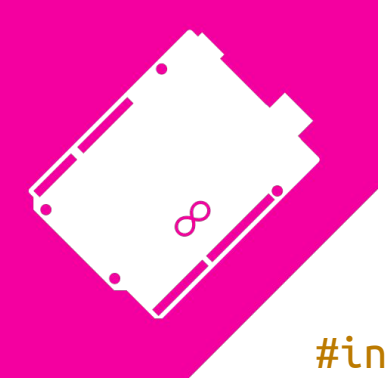
Knob Soluzione

```
#include <Servo.h>

Servo myservo;
    // create servo object to control a servo
    // hint: variables outside a function are accessible everywhere

void setup() {
    myservo.attach(9);
            // attaches the servo on pin 9 to the servo object
}

void loop() {
    int val = analogRead(A0);
            //reads the value of the potentiometer
            //(value between 0 and 1023)
    val = map(val, 0, 1023, 0, 180);
            //scale it to use with a servo (value between 0 and 180)
    myservo.write(val);
            //sets the servo position according to the scaled value
    delay(15);
            // waits for the servo to get there
}
```



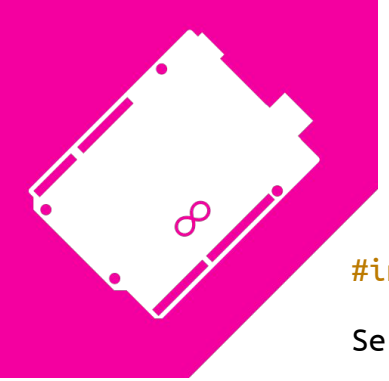
Wave

- Voglio essere salutato dal servo
- Il servo dovrebbe muoversi a destra e sinistra, e poi aspettare 10 secondi prima di salutare di nuovo
  - Hint: per chi non sa cosa è un for loop: http://www.arduino.cc/en/Reference/For
- Pro: posso utilizzare la fotoresistenza per farmi salutare solo quando sono davanti alla arduino?
  - Hint: si



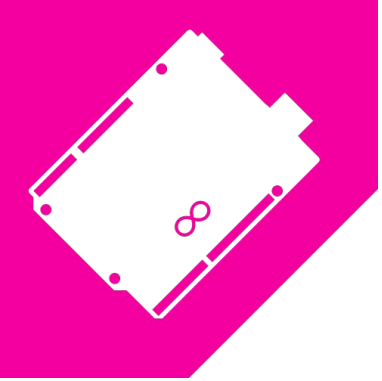
Wave Soluzione

```
#include <Servo.h>

Servo myservo;
    // create servo object to control a servo

void setup() {
    myservo.attach(9);
            // attaches the servo on pin 9 to the servo object
    myservo.write(90);
            //go to middle position
    delay(1000);
}

void loop() {
            //a for loop that repeats 3 times
    for(int i=0; i<3; i++){
            myservo.write(0);
            delay(1000);
            myservo.write(180);
            delay(1000);
    }
    myservo.write(90);
    delay(10000);
}
```



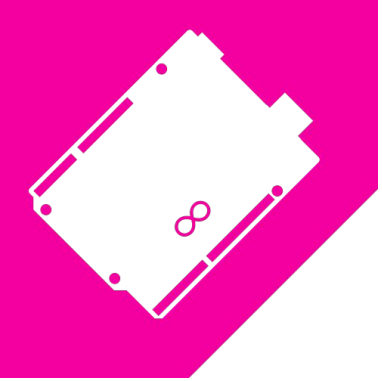
Wave Pro Soluzione

```
#include <Servo.h>

Servo myservo;
      // create servo object to control a servo

void setup() {
      myservo.attach(9);
                  // attaches the servo on pin 9 to the servo object
      myservo.write(90);
                  //go to middle position
      delay(1000);
}

void wave(){
            //using a separate function to keep everything clean
      for(int i=0; i<3; i++){
                  myservo.write(0);
                  delay(1000);
                  myservo.write(180);
                  delay(1000);
      }
      myservo.write(90);
      delay(1000);
}

int lastReading=0;
      //a good place to remember last value, across function calls
void loop() {
      int reading=analogRead(A0);
      if(abs(reading - lastReading) > 20){
            //we have a significant change in luminosity! somebody is here
            wave();
      }
      lastReading=analogRead(A0);
            //update value for next cycle
      delay(100);
}
```



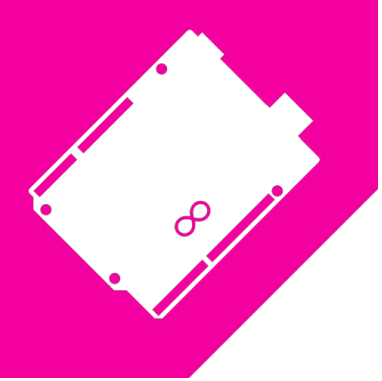
PhotoServo

- Voglio controllare la posizione del servo con la fotoresistenza
- Praticamente il codice uguale all'esempio Knob
  - La fotoresistenza non ha una risposta lineare, ci si può sbizzarrire con map
- Hint: nelle slide della scorsa volta ho mostrato come collegare una fotoresistenza



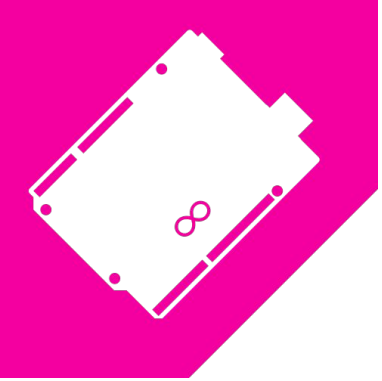
PhotoServoClock

- Come l'esempio di prima, ma voglio che ogni 3 secondi il braccetto vada a 90 gradi
- Hint: long val = millis() ritorna il numero di millisecondi passati dall'inizio dello sketch
  - Se chiamo millis() più volte, posso sottrarre i risultati per sapere quanti millisecondi sono passati fra due chiamate
  - Posso controllare che siamo passati 3000 millisecondi per fare una azione, e aggiornare un contatore



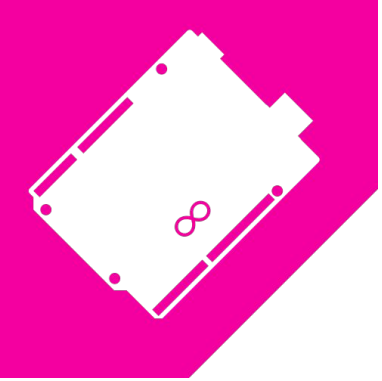
PhotoServoClock Soluzione

```
#include <Servo.h>

Servo myservo;
      //create servo object to control a servo

int potpin = 0;
      // analog pin used to connect the potentiometer
int val;
      // variable to read the value from the analog pin

long timePoint;
      //variable to save a time
void setup() {
      myservo.attach(9);
            // attaches the servo on pin 9 to the servo object
      timePoint = millis();
            //record current time
}

void loop() {
      if(millis() - timePoint > 3000){
            //have 3000 ms passed? if yes execute this action
            myservo.write(90);
            delay(500);
            timePoint = millis();
                  //update timePoint, ready for next tick
      }
      val = analogRead(potpin);          // reads the value of the potentiometer (value between 0 and 1023)
      val = map(val, 0, 1023, 0, 180);   // scale it to use it with the servo (value between 0 and 180)
      myservo.write(val);                // sets the servo position according to the scaled value
      delay(200);                         // waits for the servo to get there
}
```



Cosa proviamo oggi? - 2

- Tone
- LightTeremin
- Rgbled
- Buttons
- TeaTimer
- ALG ← chi arriva a questo mi contatti



Tone

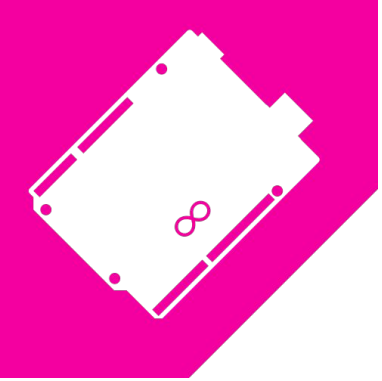
- Voglio usare la funzione tone() per far suonare un buzzer
  - http://arduino.cc/en/Reference/Tone
  - tone(pin, frequency, duration)
    - noTone(pin) per interrompere la nota
    - delay(duration*1.3) dopo tone, per separare più note
- Hint: Il La ha una frequenza di 440Hz
- il Buzzer lo spiego nella prox slide

Buzzer

- è un cosetto che fa rumore
- Quello nel nostro kit è un cristallo piezoelettrico che con una corrente alternata vibra
  - ha un senso: il – va collegato a GND, il + al pin arduino

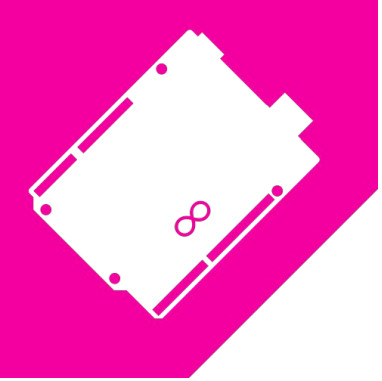
Tone Soluzione

```
void setup() {
    tone(8,220,500);
    delay(500*1.3);
    tone(8,220,500);
    delay(500*1.3);
    tone(8,220,500);
    delay(500*1.3);
    tone(8,174,375);
    delay(375*1.3);
    tone(8,262,125);
    delay(125*1.3);
    tone(8,220,500);
    delay(500*1.3);
    tone(8,174,250+125);
    delay(375*1.3);
    tone(8,262,125);
    delay(125*1.3);
    tone(8,220,1000);
    delay(1000);
    noTone(8);
}

void loop() {
  // empty
}
```



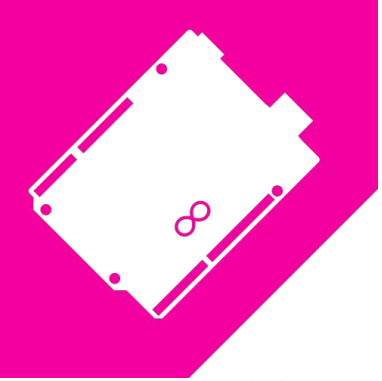
LightTeremin

- “The theremin (/ˈθɛrəmɪn/[1] THERR-ə-min; originally known as the ætherphone/etherphone, thereminophone[2] or termenvox/thereminvox) is an early electronic musical instrument controlled without physical contact by the thereminist (performer)” - Wikipedia
- Voglio controllare il suono emesso dal buzzer con una fotoresistenza
  - la fotoresistenza l'abbiamo vista alla lezione 2
  - bisogna convertire il range dell'analogRead nel range di tone
    - value=map(value, 0, 1023, 50, 10000); //per esempio



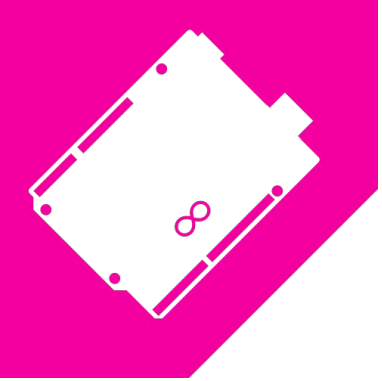
LightTeremin Soluzione

```
void setup() {
    //non serve niente
}

void loop() {
    int note = map(analogRead(A0), 0, 1023, 50, 10000);
    tone(8, note);
}
```



RGBled

- Nel kit di oggi è incluso un led rgb, già fornito di resistenze
  - possiamo attaccarlo direttamente ad arduino
  - un led rgb contiene all'interno 3 led separati
    - infatti si controlla con 4 piedini: - → GND, R → Rosso, G → Verde, B → Blu
- Voglio accendere e spegnere i led
- Hint: per accendere un led lo si attacca ad un pin, si imposta il pin come output, e si mette il pin nello stato high
  - per accenderne 3 basta collegare il led a tre pin differenti e ripetere la procedura
- Hint: è possibile anche usare analogWrite al posto di digitalWrite per ottenere meno luminosità e combinare i colori
  - random(max) o random(min, max) ritornano un valore a caso nell'intervallo

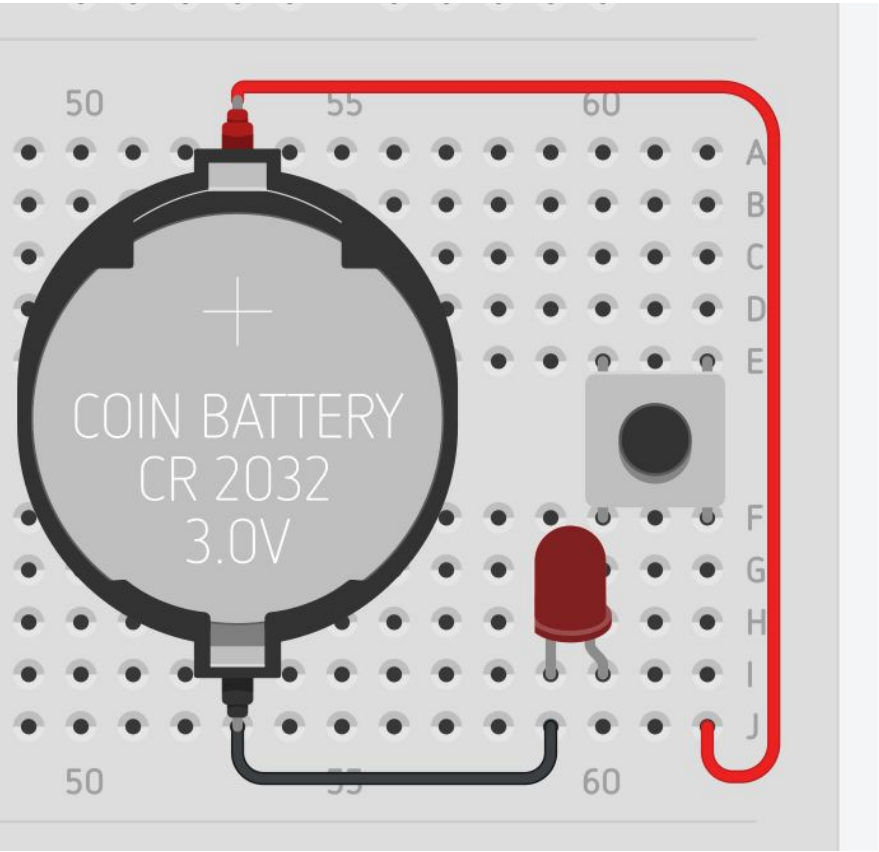


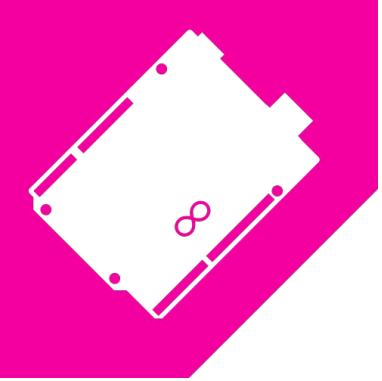
RGBled Soluzione

```
void setup() {
  pinMode(3, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop() {
  analogWrite(3, random(255));
  analogWrite(5, random(255));
  analogWrite(6, random(255));
}
```

Buttons

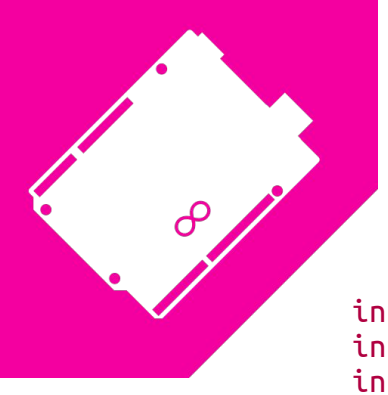
- Un bottone, quando è premuto, conduce elettricità fra i due piedini posti sullo stesso lato
  - attenzione quando lo mettete nella breadboard perché può essere un po' duro





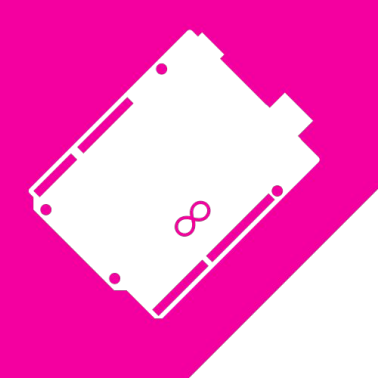
Buttons - 2

- Voglio leggere via seriale quando un bottone su un pin INPUT_PULL viene premuto o rilasciato
  - il bottone collega il pin a GND
- Voglio leggere solo una volta per pressione il messaggio in seriale
  - se stampo in seriale semplicemente il risultato di digitalRead, ad ogni esecuzione di loop il messaggio verrà rimandato
  - ho bisogno di ricordarmi lo stato precendente del bottone, e mandare un messaggio solo quando lo stato attuale cambia rispetto al precendete



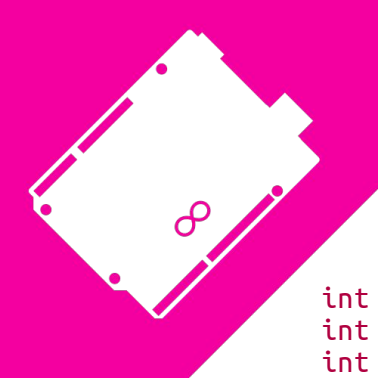
Buttons Soluzione

```
int button1 = 4; //pin for button 1
int button2 = 5; //pin for button 2
int but1_status = HIGH;
int but2_status = HIGH;
      //variables to save previous buttons state
void setup(){
      pinMode(button1, INPUT_PULLUP);
            //default state of button1 is HIGH
      pinMode(button2, INPUT_PULLUP);
            //default state of button2 is HIGH
      Serial.begin(9600);
}

void loop(){
      int but1_now=digitalRead(button1);
      if(but1_now != but1_status){
            if(but1_now==HIGH){
                  Serial.println(“bottone 1 rilasciato!”);
            }else{
                  Serial.println(“bottone 1 premuto!”);
            }
            but1_status=but1_now;
      }

      int but2_now=digitalRead(button2);
      if(but2_now != but2_status){
            if(but2_now==HIGH){
                  Serial.println(“bottone 2 rilasciato!”);
            }else{
                  Serial.println(“bottone 2 premuto!”);
            }
            but2_status=but2_now;
      }

      delay(300);
}
```



Tea Timer

- Qualcosa di più complicato
- Voglio programmare un timer per il tè, che mi avvisi con il buzzer quando è il momento di levare la bustina
- il programma ha un conto alla rovescia. ogni volta che premo il pulsante il conto alla rovescia dovrebbe essere aumentato di 10 secondi
- quando il conto alla rovescia è esaurito, il programma dovrebbe emettere 3 toni con il buzzer
- Hint: usare delay() renderebbe più complicato l'algoritmo, è meglio usare millis() per confrontarlo con l'istante in cui dovrebbe scadere il conteggio



Tea Timer Soluzione

```
int but = 4;
int but_s = HIGH;
int bzz = 8;

long future;
      //here we will save when the timer will expire
void setup(){
      pinMode(but, INPUT_PULLUP);
      int first_press=digitalRead(but);
      while(first_press==HIGH){
            delay(50);
            first_press=digitalRead(but);
                  //wait a little and read again
      }
            //button pressed! we can start the timer
      future=millis() + 10000;
}

void loop(){
      int but_now=digitalRead(but);
      if(but_now == LOW && but_s == HIGH){
            //we have a press if the previous state was different from this state
            future = future + 10000;
                  //increment timer deadline
      }
      but_status=but_now;
            //remember button state

      if(millis() > future){
            //timer finished! it's time to buzz
            tone(8, 440, 500);
            delay(500 * 1.3);
            tone(8, 660, 500);
            delay(500 * 1.3);
            tone(8, 990, 1000);
            delay(1000 * 1.3);
            noTone(8);
      }
}
```