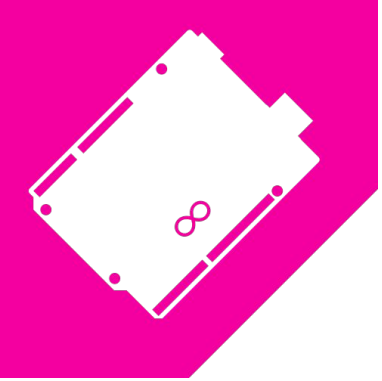


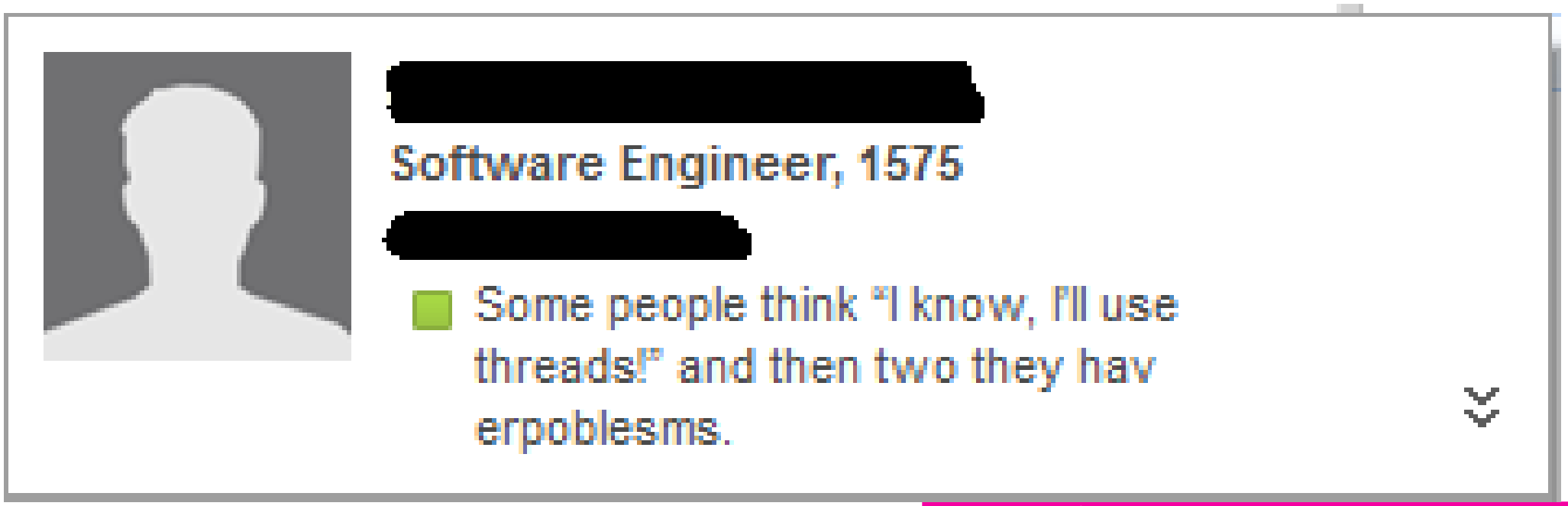
Lezione 4

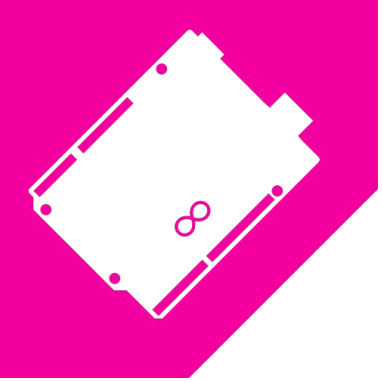
Un corso gentilmente offerto con il sudore e le lacrime di MugRomaTre e Roma Tre

e Magliana

MultiThreading

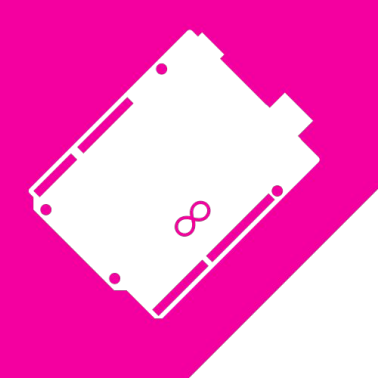
- Risposta breve: Non si può fare
- Risposta meno breve: Si può fare, circa meno quasi
  - Probabilmente sarebbe più giusto chiamarlo multitasking
    - però alla nasa fanno le cose così





Problema

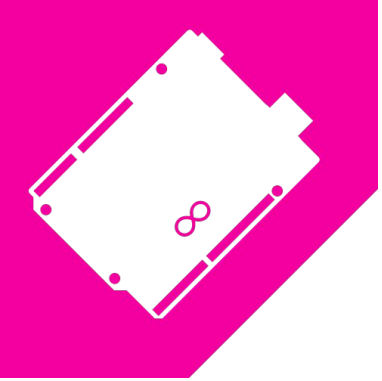
- Voglio eseguire contemporaneamente 2 azioni sulla mia arduino
- Queste due azioni si influenzano reciprocamente
- Sono ancora giovane, non voglio farmi venire un esaurimento nervoso scrivendo questo programma, e passare i migliori anni davanti a me a riprendermi mentre tutti i miei amici si danno alla pazza gioia a via di libetta



Problema: un esempio pratico

- Realizziamo un semplice gioco di simon says
  - l'RGB led mostra uno dei tre colori – Rosso, Verde, Blu. Il giocatore deve premere uno dei 3 pulsanti corrispondente al colore. Se la risposta è giusta, Bip e incrementa la velocità. se sbagliata, Bop e decrementa la velocità.
- Quale è il primo passo per trasformare questo gioco in un programma?

- Risposta: suddividerlo in Task autonomi

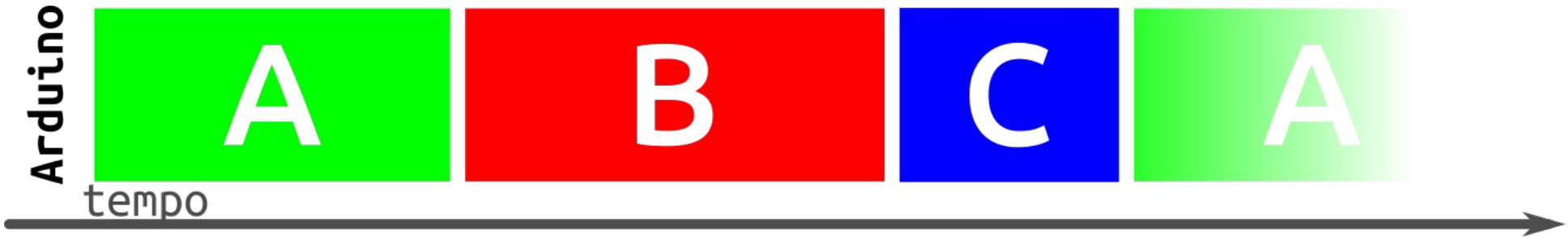


Cosa è un Task nel nostro contesto

- Un task è una azione che dipende un certo numero di input, persiste nel tempo, ed è in grado di interrompersi per riprendere l'esecuzione dopo.
- Per realizzare un sistema che riesce a dare l'impressione di essere in grado di eseguire più cose contemporaneamente, ogni task appena ne ha la possibilità si interrompe per lasciare tempo ad altri task

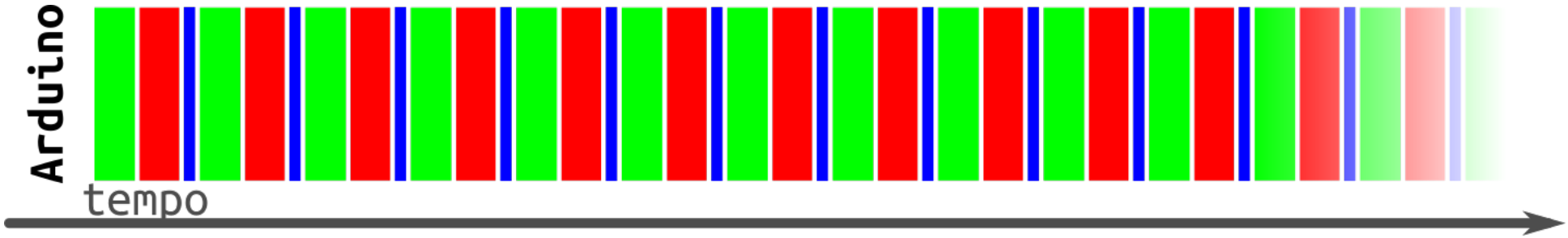
Concettualmente

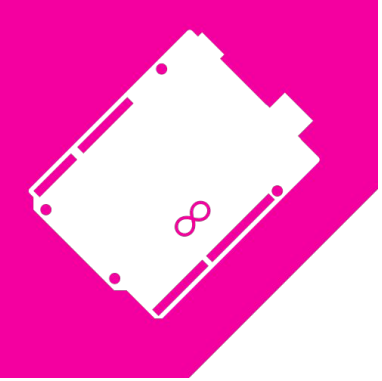
Mono - Facile



Eliminando delay
e le pause

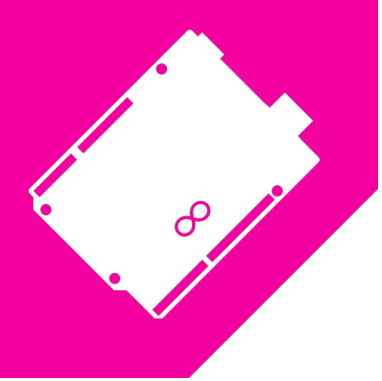
Multi - Difficile





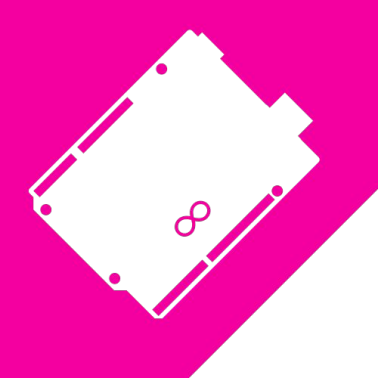
Il primo Task: Scegliere il Colore

- l'idea è trovare una unità funzionale del problema. Nel nostro caso:
  - scegliere un colore, quanto deve rimanere visibile a secondo del livello di difficoltà, e accendere il led corrispondente
  - Questa è solo una delle possibili decomposizioni, e probabilmente non una delle migliori
- Trovata l'unità funzionale, possiamo provare a scrivere l'algoritmo



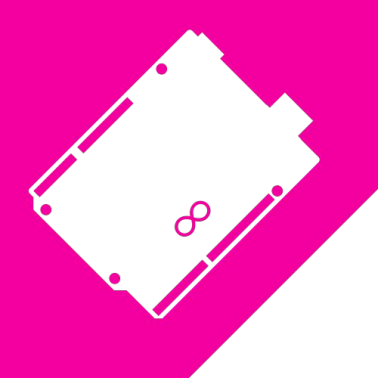
Scegliere il Colore – PseudoAlgoritmo

- Prerequisito: Esiste una variabile “difficoltà”
- 1 – scelgo un colore fra Rosso Verde Blu, e lo salvo in una variabile “colore”
- 2 - calcolo quanto deve durare il colore basandomi sulla “difficoltà”, e lo salvo in una variabile “istante_spegnimento” sommadolo con l'instante attuale
- 3 – aspetto che il tempo attuale superi “istante_spegnimento”, e quando succede spengo il led “colore” e ritorno al punto 1.
  - Il punto 3 corrisponde al loop in uno sketch arduino



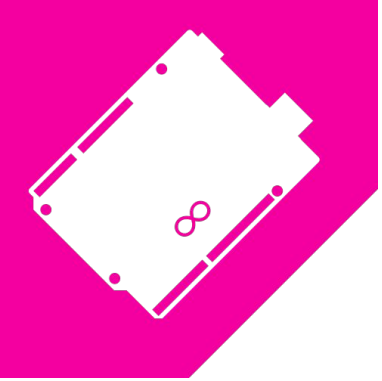
Notato Qualcosa?

- al punto 2 Aspetto che succeda qualcosa, e mentre aspetto non faccio nessun lavoro utile.
- Modificando leggermente l'algoritmo, potremmo fare in modo che l'algoritmo di prima ceda il controllo per qualcosa di più utile, per poi riprendere l'esecuzione periodicamente



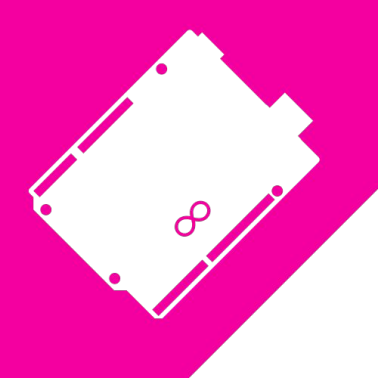
Scegliere il Colore – PseudoAlgoritmo Concorrente

- Prerequisito: Esiste una variabile ***Persistente*** “difficoltà”
  - Persistente significa che deve sopravvivere ad ogni esecuzione della funzione loop. Per esempio una variabile Globale fuori dalla funzione va bene, ma sono possibili soluzioni differenti
- 1 – Uguale a prima
- 2 – Uguale a prima
  - ma anche “istante_spegnimento” diventa persistente



...Continua

- 3 - Se Il tempo attuale è minore di “istante_spegnimento”
  - 3A - allora fermati. Prima o poi Qualcuno tornerà ad eseguire l'algoritmo dal punto 3
  - 3B – altrimenti spegni il led, ed esegui le azioni del punto 1 e 2. poi Fermati, qualcuno tornerà ad eseguire l'algoritmo dal punto 3
- Da notare che le azioni 1 e 2 sono state inglobate nel punto 3, che è la parte che va nel loop



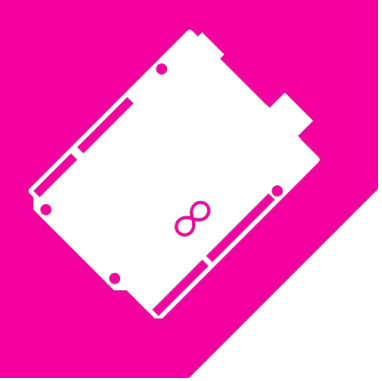
Basta, fammi vedere il codice

._. ok

```
int difficoltà;
long instante_spegnimento;
int led_color;
     //0 per rosso, 1 per verde, 2 per blu

void setup(){
     //...altro setup...

     setupRGBled();
          //...funzione scritta da qualche parte...
     difficoltà = 0;
     istante_spegimento = 0;
          //sporco trucco: così obbligo la prima esecuzione di loop a scegliere il colore del led
}

void loop(){
     if(millis() > instante_spegnimento){
          spegniRGBled();
               //...funzione scritta da qualche parte...
          led_color=random(3);
               //scegli il nuovo colore
          accendiRGBLed(led_color);
               //...funzione scritta da qualche parte...
          istante_spegnimento = millis() + 2000 - difficoltà*100;
               //spegni il led fra 2 secondi, meno se sei stato bravo
     }

     //...altre azioni ...
}
```

E il resto è lasciato come esercizio per il lettore :D

- Non c'è molto altro. una volta isolati e implementati i task, vanno inseriti uno dopo l'altro nel loop
- Ovviamente più diventa complesso il programma, più è facile avere problemi inaspettati. Come al solito, è saggio aggiungere funzionalità una alla volta

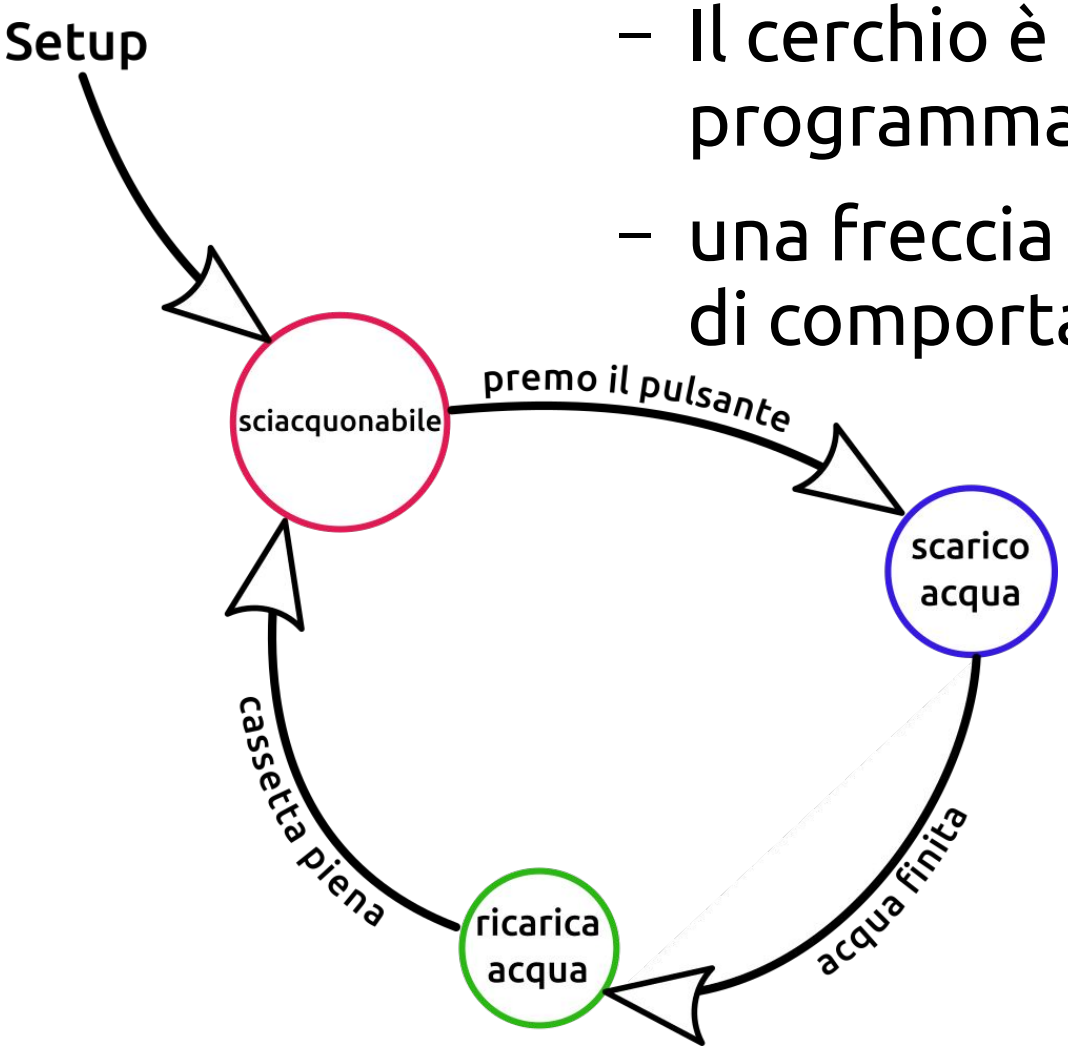


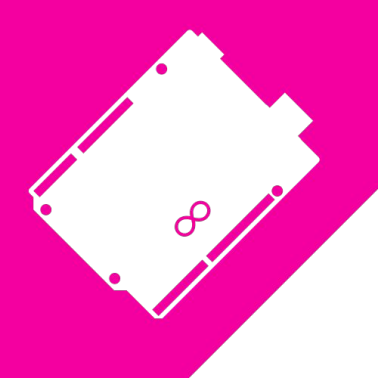
Macchine a Stati

- una macchina a stati è un modo di modellare un programma, dove il comportamento cambia nel tempo in risposta a certi stimoli
  - Un esempio è lo sciacquone: nello stato normale aspetta che qualcuno prema il bottone, poi scarica l'acqua e durante il processo di scarico premere il bottone è inutile. Ad un certo punto inizia a ricaricarsi d'acqua, e al fine del processo di ricarica diventa di nuovo utilizzabile

Lo sciacquone a Stati:

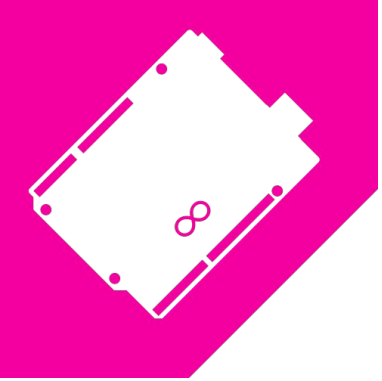
- un modo per visualizzare la cosa:
  - Il cerchio è un comportamento del programma
  - una freccia è la causa per il cambio di comportamento





Come si implementa?

- Switchone
  - vedi la bomba della seconda lezione
- e enum
  - vedi fra qualche slide



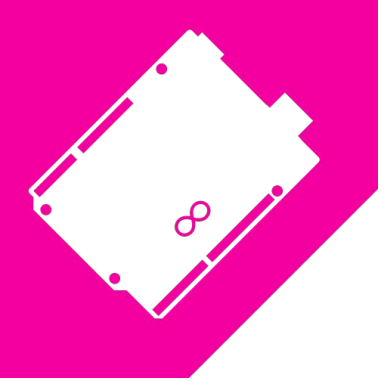
enum

- Un enum è un tipo che può assumere solo uno di una serie di valori
  - come un boolean può essere solo true o false, un enum può essere solo un valore di quelli definiti dal programmatore
- Va definito prima di usarlo
  - enum NomeEnum { Lista, Di, Possibili, Valori };
- E si usa come una variabile di tipo NomeEnum;
  - NomeEnum nomeVariabile = Possibili;

switch

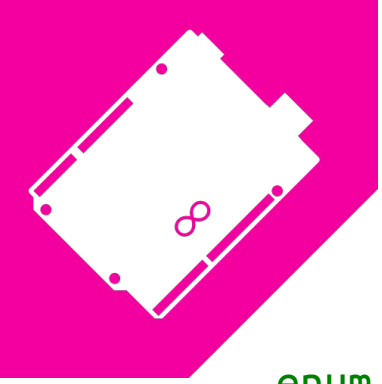
- serve per scegliere un pezzo di codice da eseguire, in base al valore di una variabile
  - e gli enum sono delle buone scelte per le variabili

```
enum Selettore {AZIONE_1, AZIONE_2, AZIONE_3};

Selettore select = AZIONE_1;
switch(select){
case AZIONE_1:
    //codice eseguito se select ha valore AZIONE_1
    break;  //importante non scordarselo
case AZIONE_2:
    //codice in caso di AZIONE_2;
    break;
case AZIONE_3:
    //codice in caso di AZIONE_3;
    break;
default:
    //si può anche mettere una sezione da eseguire
    //se select ha un valore non trattato nei casi precedenti
    break;
}
```



Mettendo insieme:

- uso una variabile persistente per salvare lo stato in cui mi trovo
- nello switch, un case per ogni stato. in ogni case eseguo le azioni associate allo stato, e controllo se è necessario cambiare stato

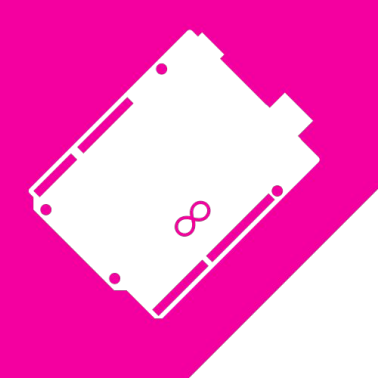


in pratica

```
enum StatoSciacquone {SCIACQUONABILE, SCARICO_ACQUA, RICARICO_ACQUA};

StatoSciacquone wc;
void setup(){
    wc=SCIACQUONABILE;
        //stato iniziale
}

void loop(){
    switch(wc){
    case RICARICO_ACQUA:
        ricaricaAcqua();
        wc=SCIACQUONABILE;
        break;
    case SCARICO_ACQUA:
        scaricaAcqua();
        wc=RICARICO_ACQUA;
        break;
    case SCIACQUONABILE:
        aspettaCatenella();
        wc=SCARICO_ACQUA;
        break;

    }
}
```

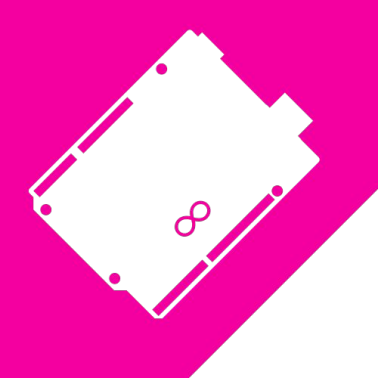
Protip:

- posso eseguire più macchine a stati insieme, e niente vieta di mischiare questa tecnica con la precedente
  - per chi vuole chiarimenti, MUG e passa la paura



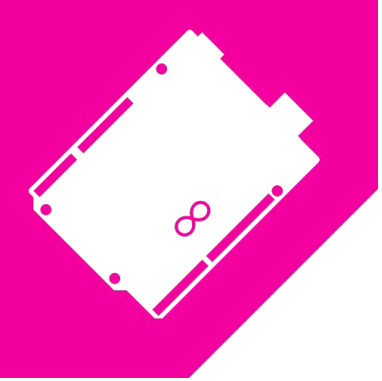
Ospiti di Oggi

- Vincenzo D'Orso
- John D'Orazio
- la loro Hard-doo-eeno



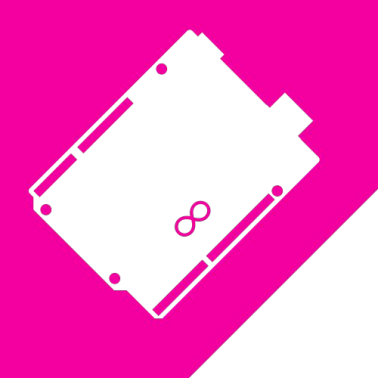
Cosa proviamo oggi? - 1

- 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

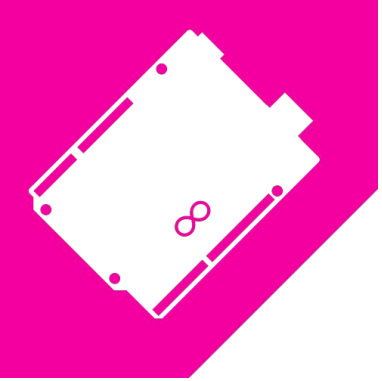


Blink Soluzione

```
/*
 * 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
}
```

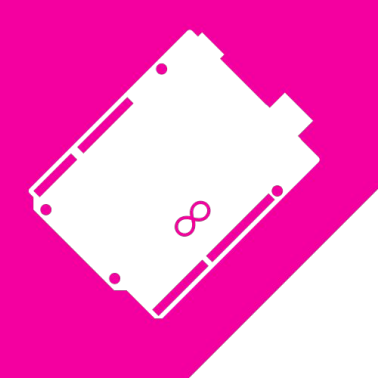
SerialAnalogRead

- 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)



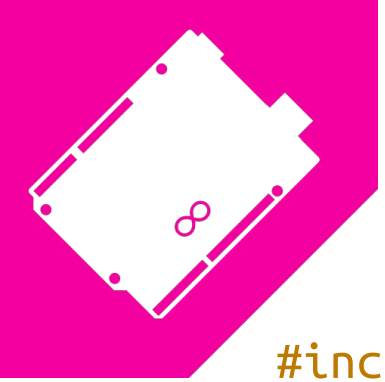
SerialAnalogRead Soluzione

```
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);
}
```



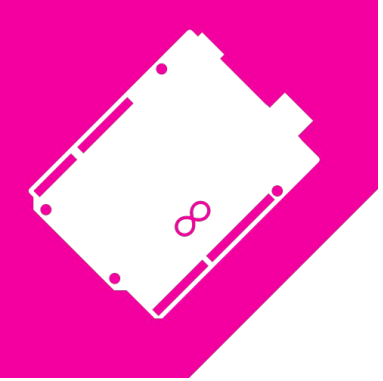
Knob

- 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



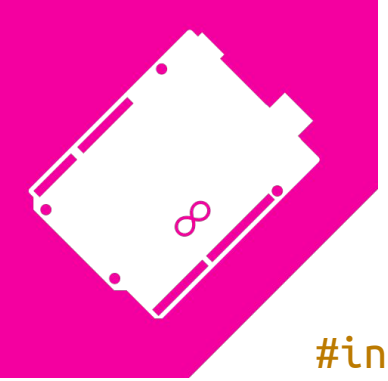
Knob Soluzione

```
#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
}
```



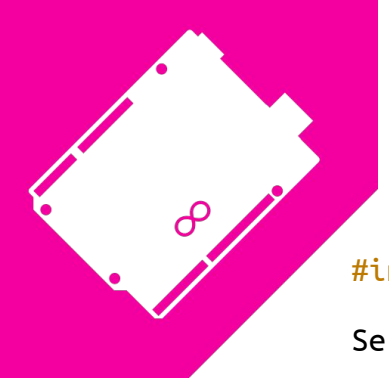
Wave

- 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



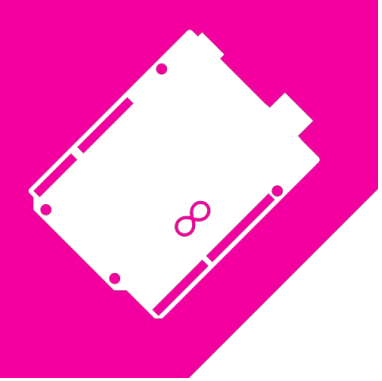
Wave Soluzione

```
#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);
}
```



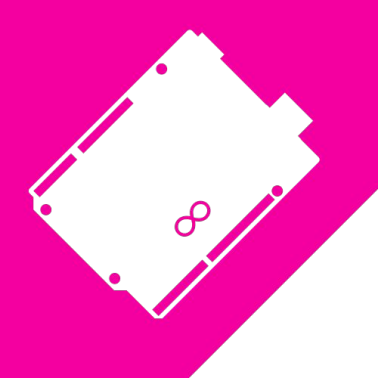
Wave Pro Soluzione

```
#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);
}
```



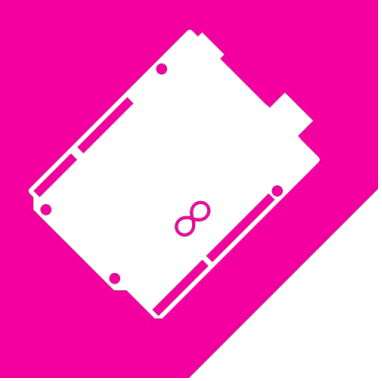
PhotoServo

- 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



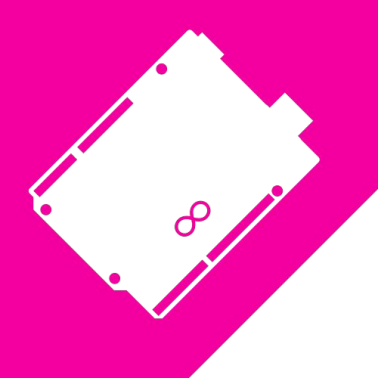
PhotoServoClock

- 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



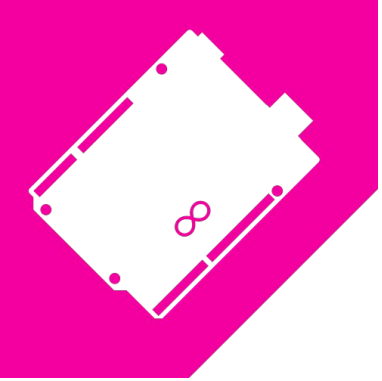
PhotoServoClock Soluzione

```
#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



Tone

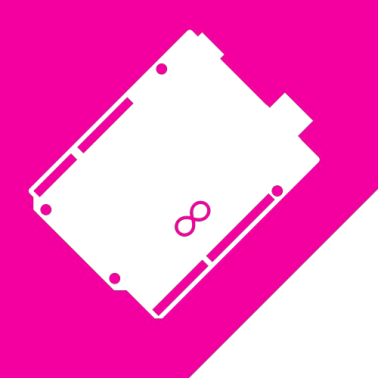
- Voglio usare la funzione tone() per far suonare un buzzer
  - http://arduino.cc/en/Reference/Tone
  - tone(pin, frequency, duration)
    - noTone(pin) per interrompere la nota
    - delay(duration*1.3) dopo tone, per separare più note
- Hint: Il La ha una frequenza di 440Hz
- il Buzzer lo spiego nella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

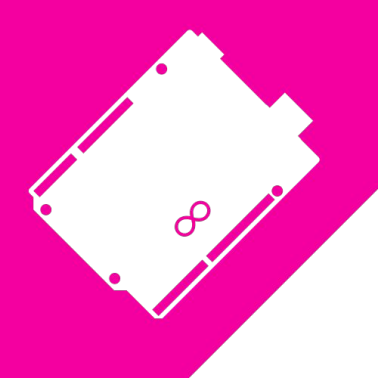
Tone Soluzione

```
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
}
```



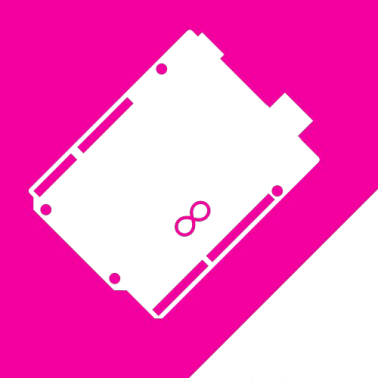
LightTeremin

- “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



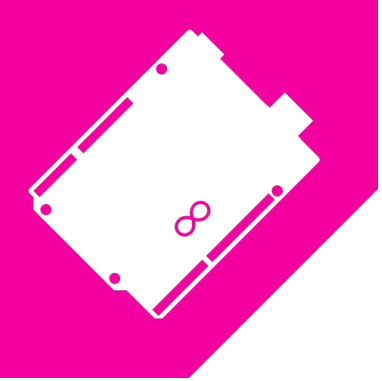
LightTeremin Soluzione

```
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

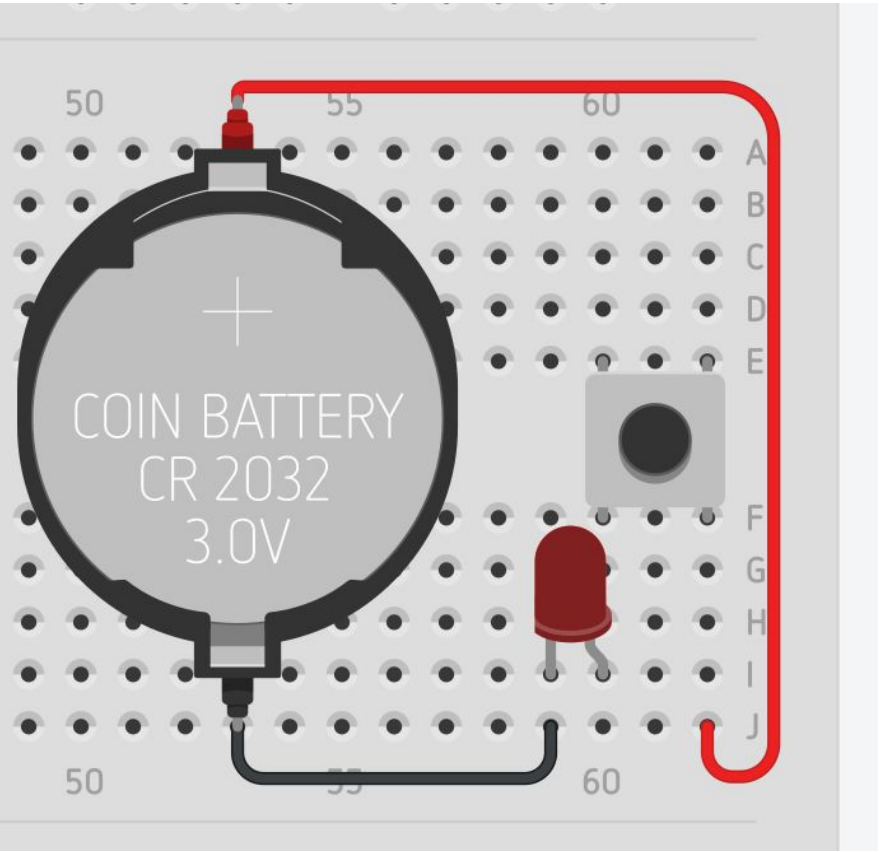


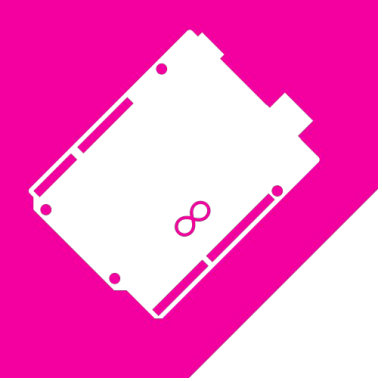
RGBled Soluzione

```
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

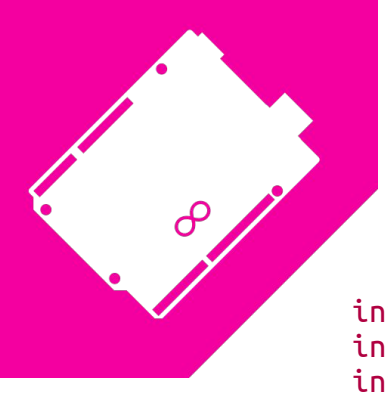
- Un bottone, quando è premuto, conduce elettricità fra i due piedini posti sullo stesso lato
  - attenzione quando lo mettete nella breadboard perché può essere un po' duro





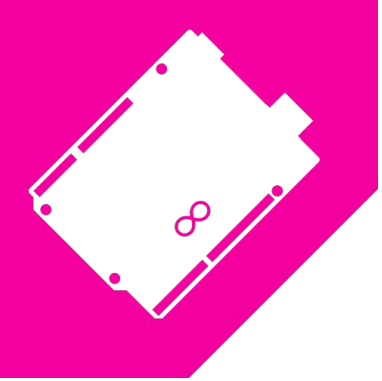
Buttons - 2

- 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



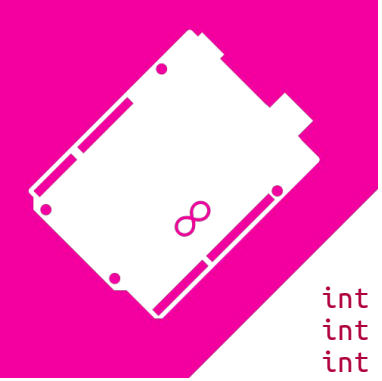
Buttons Soluzione

```
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);
}
```



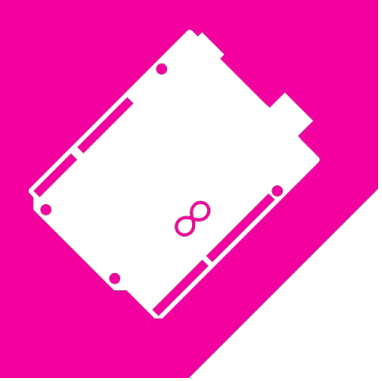
Tea Timer

- 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



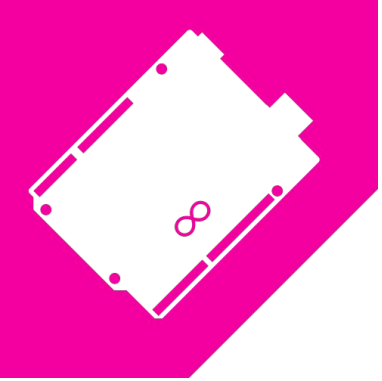
Tea Timer Soluzione

```
int but = 4;
int but_status = 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);
      }
}
```



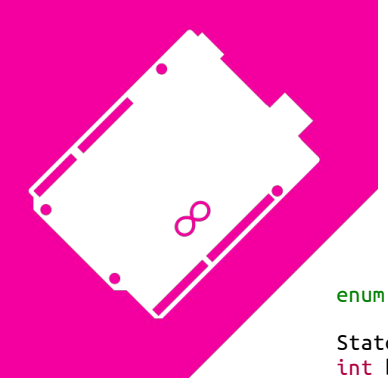
Cosa proviamo oggi? - 3

- sciacquone a stati
- simone dice



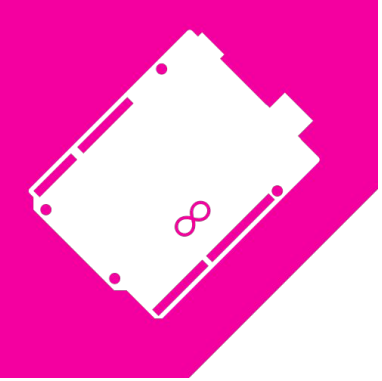
sciacquone a stati

- usiamo una macchina a stati per implementare uno sciacquone
- praticamente completiamo il codice di prima
- usiamo un buzzer per fare il rumore dell'acqua che scarica, e dell'acqua che ricarica
  - vedi esercizi buzzer
  - hint: tone(8, random(50, 10000)); fa un bel rumore
- e un bottone per simulare il pulsante
  - vedi esercizi button



soluzione

```
enum StatoSciacquone {SCIACQUONABILE, SCARICO_ACQUA, RICARICO_ACQUA};

StatoSciacquone wc;
int buzz=8;
int button = 9;

void setup(){
        pinMode(button, INPUT_PULLUP);
        wc=SCIACQUONABILE;
}

void ricaricaAcqua(){
        for(int i=0; i<5; i++){
                tone(buzz, 1000 + i*100, 100);
                delay(1000);
        }
        noTone(buzz);
}

void scaricaAcqua(){
        long future= millis() + 4000;
        while(millis()<future){
                tone(buzz, random(50, 10000));
                delay(10);
        }
        noTone(buzz);
}

void aspettaCatenella(){
        while(digitalRead(button)==LOW);
                //wait for button to be in a rest state
                //here button is HIGH
        while(digitalRead(button)==HIGH);
                //wait for a press
}

void loop(){
        switch(wc){
        case RICARICO_ACQUA:
                ricaricaAcqua();
                wc=SCIACQUONABILE;
                break;
        case SCARICO_ACQUA:
                scaricaAcqua();
                wc=RICARICO_ACQUA;
                break;
        case SCIACQUONABILE:
                aspettaCatenella();
                wc=SCARICO_ACQUA;
                break;

        }
}
```



simone dice

- simon says, il gioco come è spiegato nelle slide
- un led rgb, 3 bottoni. niente più
- magari anche un buzzer, no?

simone dice soluzione

```
int difficol;
long istante_spegnimento;
int led_color;
  //0 per rosso, 1 per verde, 2 per blu

int red_pin=3;
int green_pin=4;
int blue_pin=5;

int red_button=6;
int green_button=7;
int blue_button=8;

int rb_status=HIGH;
int gb_status=HIGH;
int bb_status=HIGH;

int buzz = 9;

void setup(){
  pinMode(red_pin, OUTPUT);
  pinMode(green_pin, OUTPUT);
  pinMode(blue_pin, OUTPUT);

  pinMode(red_button, INPUT_PULLUP);
  pinMode(green_button, INPUT_PULLUP);
  pinMode(blue_button, INPUT_PULLUP);

  difficol = 0;
  istante_spegimento = 0;
    //sporco trucco: cosi obbligo la prima esecuzione di loop a scegliere il colore del led
}

void loop(){
  if(millis() > instante_spegnimento){
    digitalWrite(red_pin, LOW);
    digitalWrite(green_pin, LOW);
    digitalWrite(blue_pin, LOW);

    led_color=random(3);
      //scegli il nuovo colore
    digitalWrite(led_color + red_pin, HIGH);
    istante_spegnimento = millis() + 2000 - difficol*100;
      //spegni il led fra 2 secondi, meno se sei stato bravo
  }

  boolean red_pressed = rb_status==HIGH && digitalRead(red_button)==LOW;
    rb_status=digitalRead(red_button);
  boolean green_pressed = gb_status==HIGH && digitalRead(green_button)==LOW;
    gb_status=digitalRead(green_button);
  boolean blue_pressed = bb_status==HIGH && digitalRead(blue_button)==LOW;
    bb_status=digitalRead(blue_button);

  if(red_pressed){
    if(led_color==0){
      difficol++;  //win!
    }
    istante_spegnimento = 0;
  }


  if(green_pressed){
    if(led_color==1){
      difficol++;  //win!
    }
    istante_spegnimento = 0;
  }

  if(blue_pressed){
    if(led_color==2){
      difficol++;  //win!
    }
    istante_spegnimento = 0;
  }
```