



Progetto Arduino

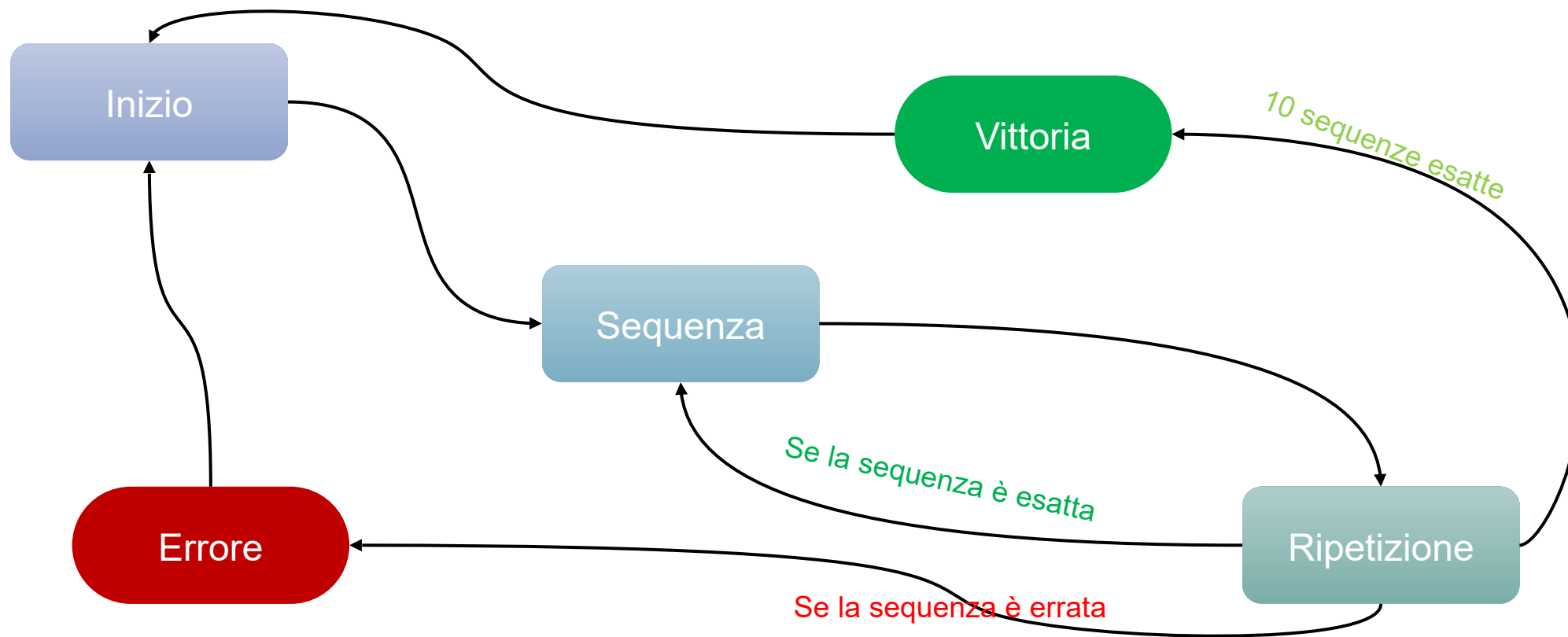
GIULIA CUTTONÉ

Simon game

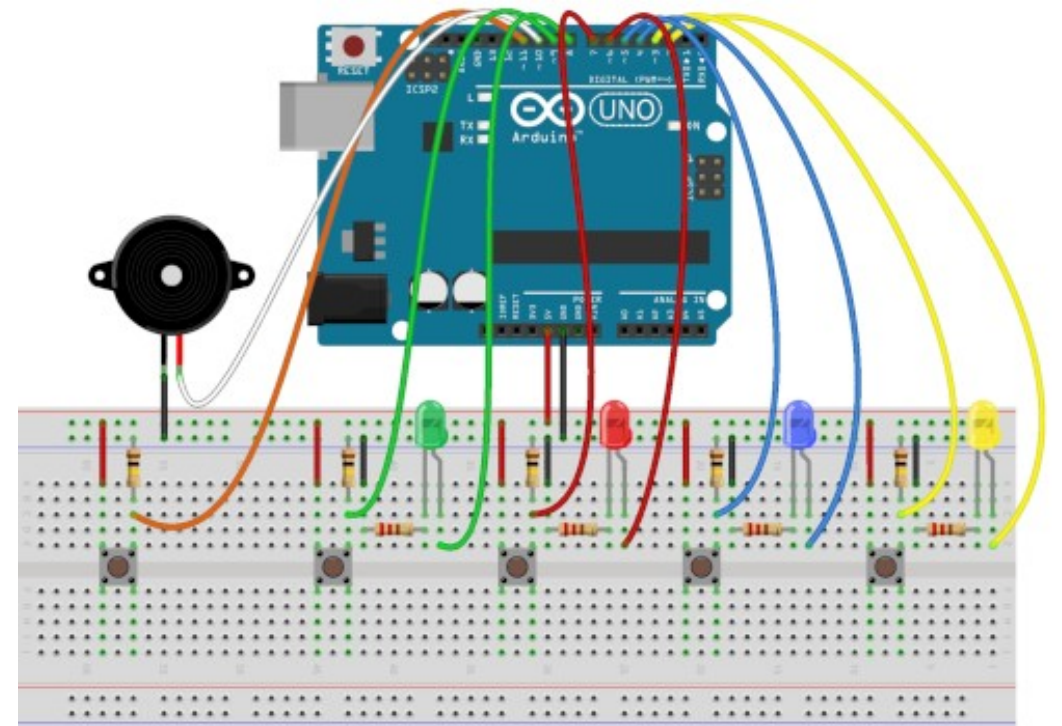
- Il gioco è una sorta di variante elettronica del gioco per bambini noto nel mondo anglosassone come ***Simon says***.
- "Simon" si presenta come un disco con quattro grandi pulsanti di colore rosso, blu, verde e giallo.
- Questi pulsanti si illuminano secondo una sequenza casuale; all'illuminazione di ciascun pulsante è associata anche l'emissione di una determinata nota musicale. Una volta terminata la sequenza, il giocatore deve ripeterla premendo i pulsanti nello stesso ordine.
- Se riesce in questo compito il giocatore si vede proporre una nuova sequenza, uguale alla precedente con l'aggiunta di nuovo pulsante/tono; la sequenza da ripetere diventa quindi sempre più lunga e il compito del giocatore più difficile.



Funzionamento del gioco



Schema del circuito Arduino



Codice del programma

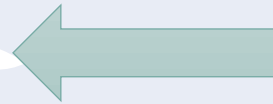
Definizione delle costanti

```
#define BUZZER_PIN 2  
#define START_BUTTON_PIN 13
```



porte pin (buzzer e pulsante start)

```
#define LEVELS 10
```



numero di livelli impostato a 10

```
#define STATE_STOPPED 0  
#define STATE_SHOW 1  
#define STATE_REPEAT 2
```



fasi del gioco

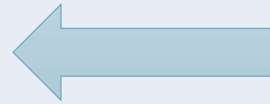
Dichiarazione e inizializzazione degli array

```
int buttons[4] = {11, 9, 7, 5};  
int led[4] = {10, 8, 6, 4};  
int notes[4] = {523, 587, 659, 698};
```



array di pin
(pulsanti, led e note)

```
int sequence[LEVELS];
```



array che mostra la sequenza

Dichiarazione di e inizializzazione di variabili

```
int level = 0;  
int index = 0;
```



livello del gioco e indice

```
int state =  
STATE_STOPPED;
```



stato attuale
inizializzato a
STATE_STOPPED

Funzione setup

Descrizione

La funzione **setup()** viene chiamata all'avvio di uno sketch.

Usata per inizializzare variabili, modalità pin, iniziare a usare le librerie, ecc.

La funzione **setup()** verrà eseguita solo una volta, dopo ogni accensione o reset della scheda Arduino.

```
void setup() {
```

```
  for(int i=0; i<4; i++){  
    pinMode(buttons[i], INPUT);  
    pinMode(led[i], OUTPUT);  
  }
```



impostiamo i pin dei pulsanti in 'input' e i pin dei led in 'output'

```
  pinMode(START_BUTTON_PIN,  
INPUT);  
  pinMode(BUZZER_PIN, OUTPUT);
```



pin del pulsante start in 'input' e pin del buzzer in 'output'

```
  randomSeed(analogRead(A0));  
}
```



sequenza casuale di numeri,
A0 è un pin analogico non collegato

Funzione loop

Descrizione

Dopo aver creato una funzione **setup()**, che inizializza e imposta i valori iniziali, la funzione **loop()** si ripete consecutivamente, consentendo al programma di cambiare e rispondere.

Usata per controllare attivamente la scheda Arduino.

```
void loop() {
```

```
  if(digitalRead(START_BUTTON_PIN) == HIGH)
    welcome();
```



se il pulsante start viene premuto inizia il gioco
(viene chiamata la funzione 'welcome()')

```
  if(state == STATE_SHOW)
    showNextLevel();
```



se ci troviamo nello stato 'STATE_SHOW' viene
mostrata la sequenza successiva
(viene chiamata la funzione 'showNextLevel()')

```
  if(state == STATE_REPEAT)
    repeatSequence();
```



se ci troviamo nello stato 'STATE_REPEAT' deve
essere ripetuta la sequenza
(viene chiamata la funzione 'repeatSequence()')

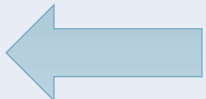
```
  delay(10);
}
```


Funzione welcome


Descrizione

La funzione **welcome()** indica l'inizio di una nuova partita generando 7 note casuali.

```
void welcome(){  
    int randomNumber;  
  
    for(int i=0; i<7; i++){  
        randomNumber = random(4);  
        tone(BUZZER_PIN, notes[randomNumber], 150);  
        digitalWrite(led[randomNumber], HIGH);  
        delay(75);  
        digitalWrite(led[randomNumber], LOW);  
        delay(75);  
    }  
  
    level = 0;  
    state = STATE_SHOW;  
    delay(1200);  
}
```



ciclo che genera una nota casuale dall'array 'notes' di posizione 'randomNumber' e illumina il led corrispondente



Impostiamo il livello a 0 e passiamo allo stato successivo

Funzione showNextLevel

Descrizione

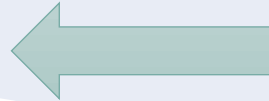
La funzione **showNextLevel()** genera una nuova nota e mostra tutta la sequenza creata fin adesso.

```
void showNextLevel(){  
    sequence[level] = random(4);
```



genera una nota casuale e la aggiunge alla sequenza in posizione 'level'

```
    for(int i=0; i<=level; i++){  
        playNote(sequence[i]);  
        delay(200);  
    }
```



ciclo che suona le note della sequenza attuale

```
    index = 0;  
    level++;  
    state = STATE_REPEAT;  
}
```



incrementiamo il livello e passiamo allo stato successivo

Funzione repeatSequence

Descrizione

La funzione **repeatSequence()** richiede la ripetizione della sequenza attuale.

```
void repeatSequence(){  
    int selectedNote = readButtons();
```



legge il pulsante premuto

```
    if(selectedNote >= 0){  
        if(selectedNote == sequence[index]){  
            playNote(selectedNote);  
            while(readButtons() != -1);  
            index++;
```



se un pulsante è stato premuto, e corrisponde alla nota aspettata possiamo suonarla

```
        if(index >= level){  
            if(level < LEVELS){  
                state = STATE_SHOW;  
                delay(1000); }  
            else win(); }
```



se non abbiamo raggiunto il livello massimo ritorniamo alla visualizzazione della sequenza

altrimenti si conclude il gioco con una vittoria

```
    }  
    else error(selectedNote);  
}
```



se il pulsante premuto non corrisponde alla sequenza si genera un errore e termina la partita

Funzione win

Descrizione

La funzione **win()** indica la vittoria (conclusione di tutti e 10 i livelli) facendo lampeggiare tutti e 4 i led.

```
void win(){  
  int randomNumber;  
  delay(200);  
  
  for(int i=0; i<12; i++){  
    randomNumber = random(4);  
    tone(BUZZER_PIN, notes[randomNumber], 150);  
  
    for(int j=0; j<4; j++){  
      digitalWrite(led[j], HIGH);  
      delay(75);  
  
      for(int j=0; j<4; j++){  
        digitalWrite(led[j], LOW);  
        delay(75); }  
  
    state = STATE_STOPPED;  
  }
```



accende tutti i led



spegne tutti i led



ferma il gioco

Funzione error

Descrizione

La funzione **error()** indica che è stato commesso un errore (tasto premuto fuori sequenza) e lo fa notare attraverso una nota bassa.

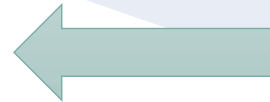
La funzione contiene un parametro intero 'note' che indica la nota errata

```
void error(int note){  
    tone(BUZZER_PIN, 200, 1000);
```



nota bassa

```
    digitalWrite(led[note], HIGH);  
    delay(1000);  
    digitalWrite(led[note], LOW);
```



fa lampeggiare il led
corrispondente alla nota errata

```
    state = STATE_STOPPED;  
}
```



ferma il gioco

Funzione playNote

Descrizione: La funzione **playNote()** suona una delle 4 note musicali (indicata come parametro)

```
void playNote(int note){  
    tone(BUZZER_PIN, notes[note], 150);  
    digitalWrite(led[note], HIGH);  
    delay(150);  
    digitalWrite(led[note], LOW);  
}
```

Funzione readButtons

Descrizione: La funzione **readButtons()** legge tutti e 4 i pulsanti e sa quale è stato premuto

```
int readButtons(){  
    for(int i=0; i<4; i++)  
        if(digitalRead(buttons[i]) == HIGH)  
            return i;  
  
    return -1;  
}
```