PROGRAMMAZIONE A STATI FINITI CON ARDUINO

*Nella classica programmazione con l'IDE di Arduino, abbiamo un ciclo **setup** ed un ciclo **loop**, nel **setup** le istruzioni vengono eseguite solamente all'avvio mentre nel ciclo **loop** le istruzioni vengono eseguite ciclicamente.*

*La scritta void sta ad indicare che il ciclo **setup** ed il ciclo **loop** , non restituiscono nessun risultato.*

Questo aspetto verrà approfondito successivamente quando verranno trattate le funzioni.

```
void setup() {
  // put your setup code here, to run once.

  // scrivi qui il tuo codice di setup,
  //verrà eseguito una sola volta.
}

void loop() {
  // put your main code here, to run repeatedly.

  //scrivi il tuo codice principale qui,
  //verrà eseguito ripetutamente.
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

La scelta di inserire tutto il programma nel ciclo loop, non è però consigliata nel caso in cui il programma sia complesso e composto da svariate parti.

Proviamo ad esempio a strutturare un programma che esegua tre semplici funzioni su un led collegato su un'uscita della scheda. Supponiamo ad esempio che il led possa eseguire 3 operazioni definite e cioè:

- *Acceso a luce fissa.*
- *Lampeggiante lento*
- *Lampeggiante veloce*

Le 3 operazioni possono essere selezionate premendo un pulsante e più esattamente ad ogni pressione del pulsante, avremmo un cambio di programma, come se di fatto ci fosse un ciclo loop differente.

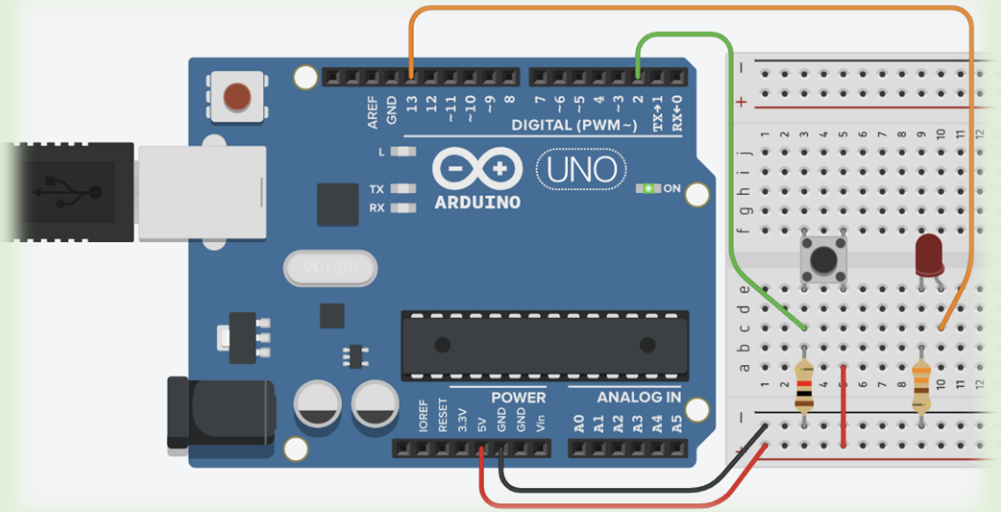
Proviamo a vedere una possibile soluzione con il led collegato al pin 13 (led a bordo della scheda) ed il pulsante sul pin 2.

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Possiamo subito definire lo schema di collegamento ed il ciclo setup.

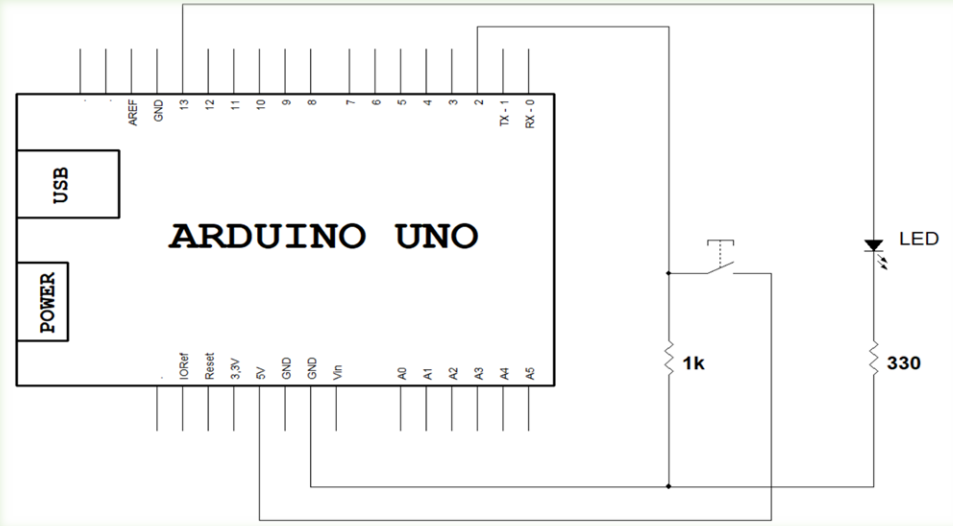
```
#define LED 13
#define PULSANTE 2

void setup() {
  pinMode(LED,OUTPUT);       //definisco il pin 13 come uscita
  pinMode(PULSANTE,INPUT);   //definisco il pin 2 come ingresso
  digitalWrite(LED,LOW);     //spengo il pin 13
}
```



La prima considerazione da fare, è quella legata al fatto che ***l'evento che fa cambiare la funzione da realizzare è sempre lo stesso e cioè la pressione del medesimo pulsante collegato al pin2.***

Pertanto sarà necessario distinguere il quale stato si trova il sistema alla pressione del pulsante, in questo modo saremo in grado di passare allo stato successivo.

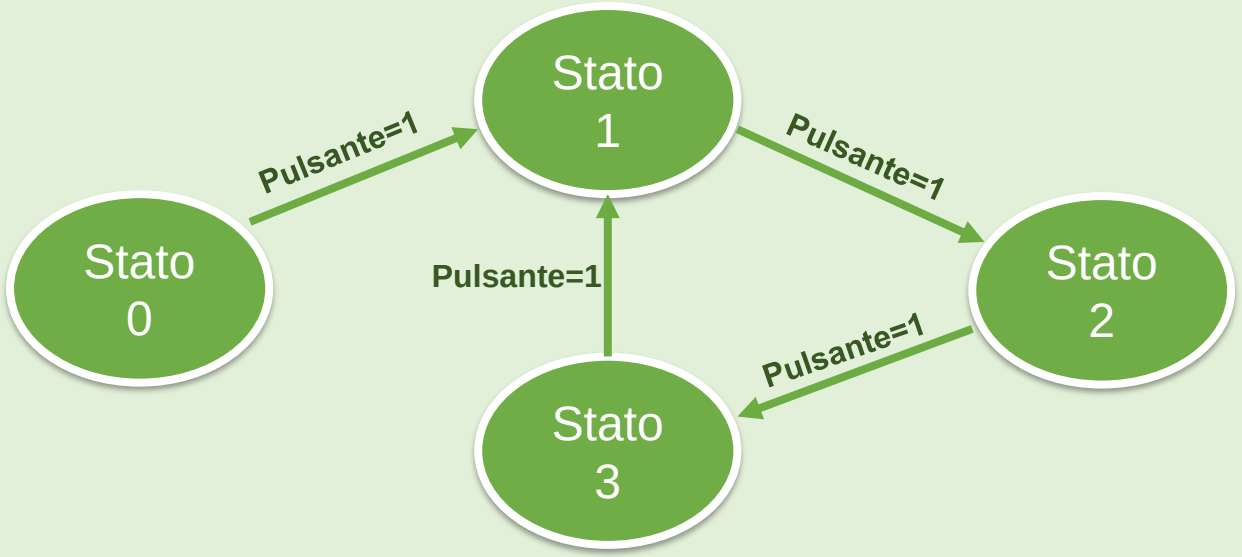


PROGRAMMAZIONE A STATI FINITI CON ARDUINO

E' fondamentale conoscere a priori la quantità di stati in cui il sistema si può trovare, e nello specifico abbiamo 4 possibilità:

- *Stato 0 = led spento (stato iniziale)*
- *Stato 1 = led acceso.*
- *Stato 2 = led lampeggiante lento.*
- *Stato 3 = led lampeggiante velocemente.*

Rappresentiamo gli stati con del cerchi, e sul passaggio da uno stato all'altro mettiamo l'evento che causa la transizione.



PROGRAMMAZIONE A STATI FINITI CON ARDUINO

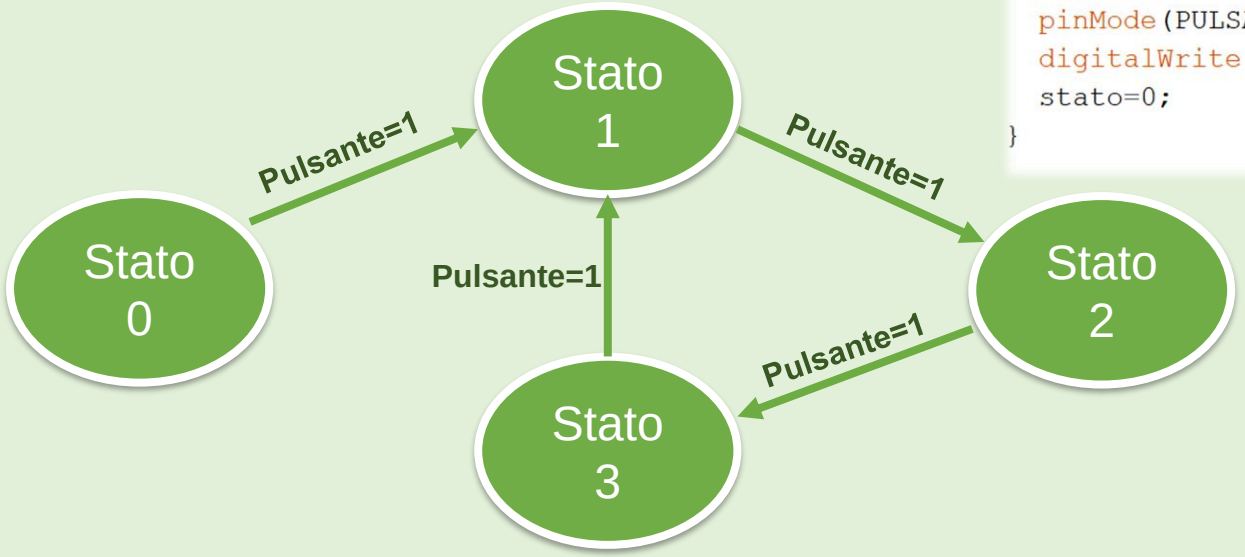
Avendo lo stesso evento che causa il passaggio da uno stato all'altro, servirà memorizzare su una variabile lo stato in cui il sistema si trova e di conseguenza effettuare il passaggio allo stato successivo all'evento che sarà sempre la pressione del pulsante.

La variabile stato verrà inizializzata al valore 0 nel setup.

```
#define LED 13
#define PULSANTE 2

int stato;

void setup() {
  pinMode(LED,OUTPUT);       //definisco il pin 13 come uscita
  pinMode(PULSANTE,INPUT);   //definisco il pin 2 come ingresso
  digitalWrite(LED,LOW);     //spengo il pin 13
  stato=0;
}
```



PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Ad ogni pressione del pulsante dovremo pertanto verificare in quale stato ci troviamo per poter poi decidere quale sarà lo stato da raggiungere.

Con la funzione digitalRead(PULSANTE) si legge lo stato del pulsante, se è premuto vale 1.

In questo modo possiamo gestire il passaggio tra i vari stati, non abbiamo effettuato alcuna funzione, ma possiamo già decidere di ottimizzare il programma utilizzando lo ***switch-case****.*

```
void loop() {
   if((digitalRead(PULSANTE)==1)&&(stato==0)) {
    stato=1;
   }
   if((digitalRead(PULSANTE)==1)&&(stato==1)) {
    stato=2;
   }
   if((digitalRead(PULSANTE)==1)&&(stato==2)) {
    stato=3;
   }
   if((digitalRead(PULSANTE)==1)&&(stato==3)) {
    stato=1;
   }
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Con lo switch-case il programma diventa il seguente, ma ancora non è stata realizzata alcuna funzione ma solo il passaggio tra i vari stati.

In questo modo il ciclo switch-case viene eseguito sempre, un modo più elegante sarà quello di fare eseguire lo switch-else solo dopo la pressione del pulsante.

Inoltre dovremmo anche fare in modo di attendere il rilascio del pulsante dopo la sua pressione, evitando in questo modo ulteriori cambio di stato ad ogni singola pressione del pulsante.

```
void loop() {
  switch (stato) {
    case 0:
       if(digitalRead(PULSANTE)==1)   stato=1;
       break;
    case 1:
       if(digitalRead(PULSANTE)==1)   stato=2;
       break;
    case 2:
       if(digitalRead(PULSANTE)==1)   stato=3;
       break;
    case 3:
       if(digitalRead(PULSANTE)==1)   stato=1;
       break;
  }
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

In questo caso pertanto avremo che solo quando il pulsante verrà premuto si andrà a decidere in quale stato andare, e successivamente si attende il rilascio del pulsante prima di proseguire.

Ora non ci rimane che eseguire la funzione richiesta in ogni stato e questo lo si può fare testando il valore della variabile stato con un ulteriore switch-case.

```
void loop() {
  if(digitalRead(PULSANTE)==1) {
    switch (stato) {
    case 0:
       stato=1;
       break;
    case 1:
       stato=2;
       break;
    case 2:
       stato=3;
       break;
    case 3:
       stato=1;
       break;
   }
   while(digitalRead(PULSANTE)==1){} //attesa rilascio
  }
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Il controllo delle 3 funzioni da realizzare può essere inserito all'interno del ciclo loop in coda al codice precedentemente scritto.

*Le 3 funzioni **led_acceso**, **lampeggio_lento** e **lampeggio_veloce**, andranno poi scritte e vengono richiamate senza necessità di restituire alcun valore, pertanto saranno di tipo void.*

Inoltre bisognerà introdurre un ritardo di 1msec alla fine del ciclo, per poter contare lo scorrimento del tempo al fine di eseguire un lampeggio lento ed uno veloce.

Vediamo di seguito le 3 funzioni.

```
switch (stato) {
  case 1:
     led_acceso();
     break;
  case 2:
     lampeggio_lento();
     break;
  case 3:
     lampeggio_veloce();
     break;
}
delay(1);
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

*Innanzitutto dichiariamo 3 nuove variabili, due intere **cont_lento** e **cont_veloce**, per gestire i due tipi di lampeggi, ed una variabile booleana per memorizzare lo stato in cui si trova il led. Le variabili andranno ovviamente inizializzate nel setup.*

```
int stato,cont_lento,cont_veloce;
bool stato_led;

void setup() {
  pinMode(LED,OUTPUT);        //definisco il pin 13 come uscita
  pinMode(PULSANTE,INPUT);    //definisco il pin 2 come ingresso
  digitalWrite(LED,LOW);      //spengo il pin 13
  stato_led=0;
  stato=0;
  cont_lento=500;
  cont_veloce=100;
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

La prima funzione, riguarda l'accensione continua del led, in questo caso le operazioni da fare sono davvero poche, e cioè accendere il led e memorizzare il suo stato nella variabile ***stato_led****.*

```
void led_acceso(){
    digitalWrite(LED,HIGH);
    stato_led=1;
}
```

Le altre due funzioni sono praticamente identiche in quanto entrambe dovranno far lampeggiare il led, ma utilizzeranno due variabili differenti inizializzate nel setup a due valori differenti

```
cont_lento=500;
cont_veloce=100;
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

La funzione che fa lampeggiare il led lentamente invece dovrà decrementare la variabile di conteggio, e nel caso la variabile sia arrivata a 0 (trascorso un tempo di 500ms grazie al ritardo inserito nel loop) accendere o spegnere il led in base al suo stato precedente.

```
void lampeggio_lento(){
     cont_lento--;          //decremento di un'unità
     if(cont_lento==0){     //se sono passati 500msec
        cont_lento=500;     //inizializzo nuovamente la variabile

        if(stato_led==0){ //se il led è spento lo accendo
           digitalWrite(LED,HIGH);
           stato_led=1;
        }
        else{               //se il led è acceso lo spengo
           digitalWrite(LED,LOW);
           stato_led=0;
        }
     }
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Stessa identica cosa per il lampeggio veloce, con l'unica differenza, che si utilizza una variabile differente che verrà inizializzata ad un valore minore per avere un lampeggio più veloce.

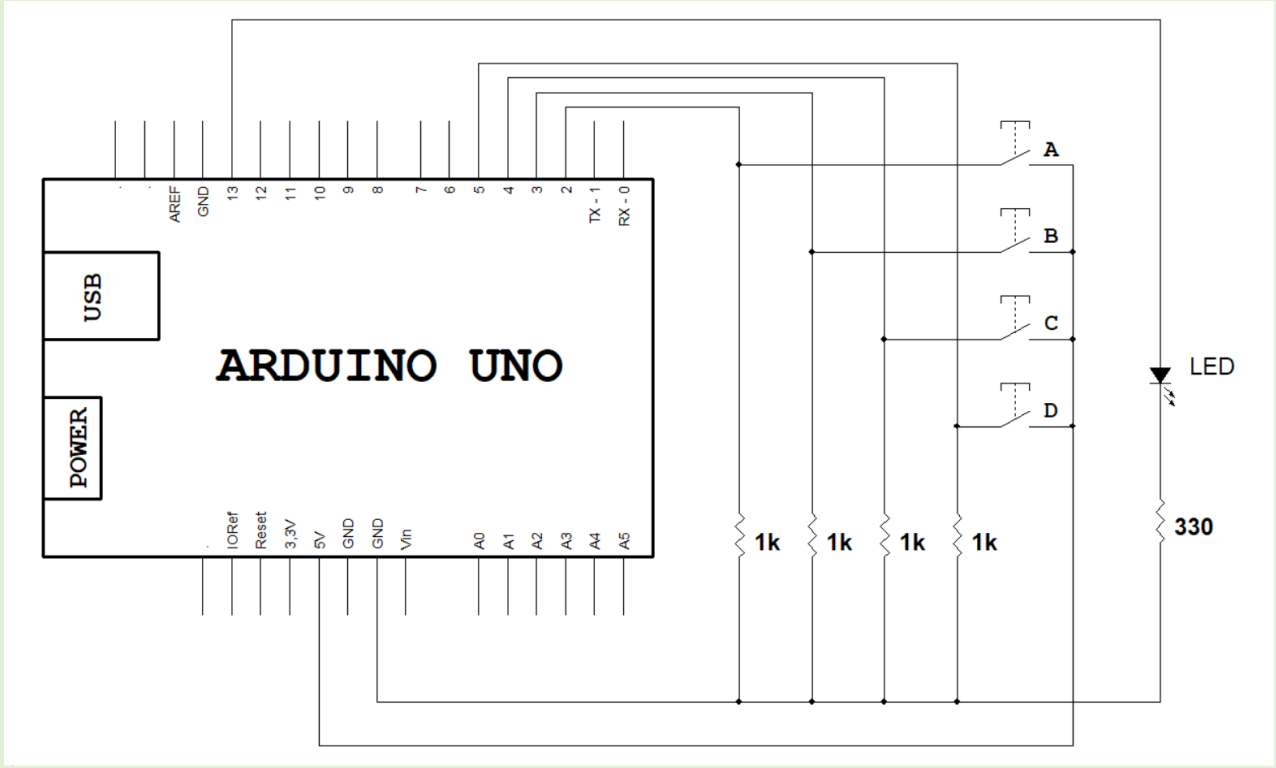
```
void lampeggio_veloce(){
     cont_veloce--;
     if(cont_veloce==0){
        cont_veloce=100;
        if(stato_led==0){
           digitalWrite(LED,HIGH);
           stato_led=1;
        }
        else{
           digitalWrite(LED,LOW);
           stato_led=0;
        }
     }
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Utilizzando questo modo di ragionare proviamo ora a risolvere un problema differente. Dato lo schema di figura, realizzare un programma che riconosca la pressione dei pulsanti secondo la seguente sequenza B-C-D-A.

Se la sequenza è corretta si accende il led per 1 secondo.

Se la sequenza è errata il led fa due lampeggi dal 200ms.



PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Il collegamento su breadboard è quello in figura, ed il programma dovrà prevedere i seguenti 6 stati.

Stato 0 = attesa 1° pulsante corretto

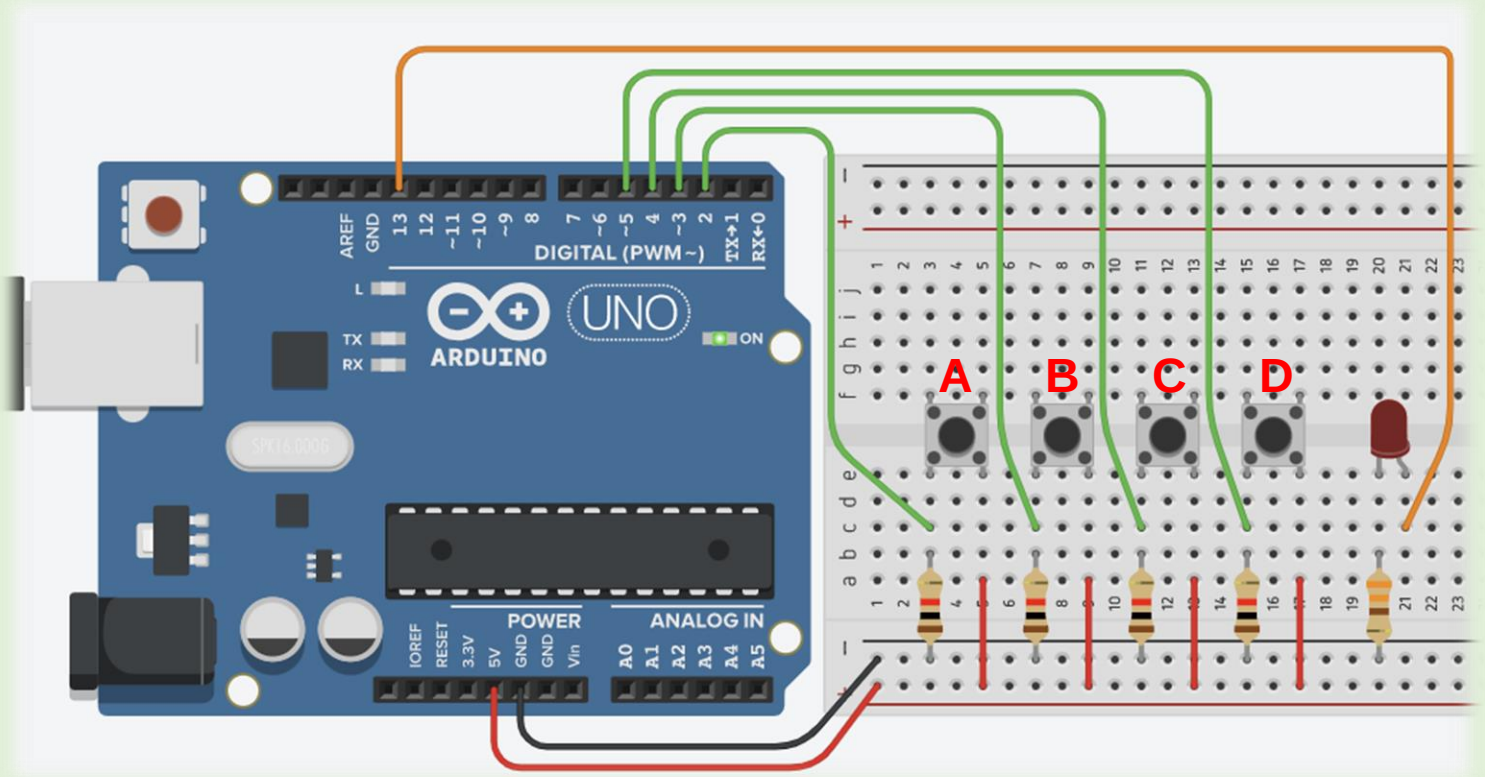
Stato 1 = attesa 2° pulsante corretto

Stato 2 = attesa 3° pulsante corretto

Stato 3 = attesa 4° pulsante corretto

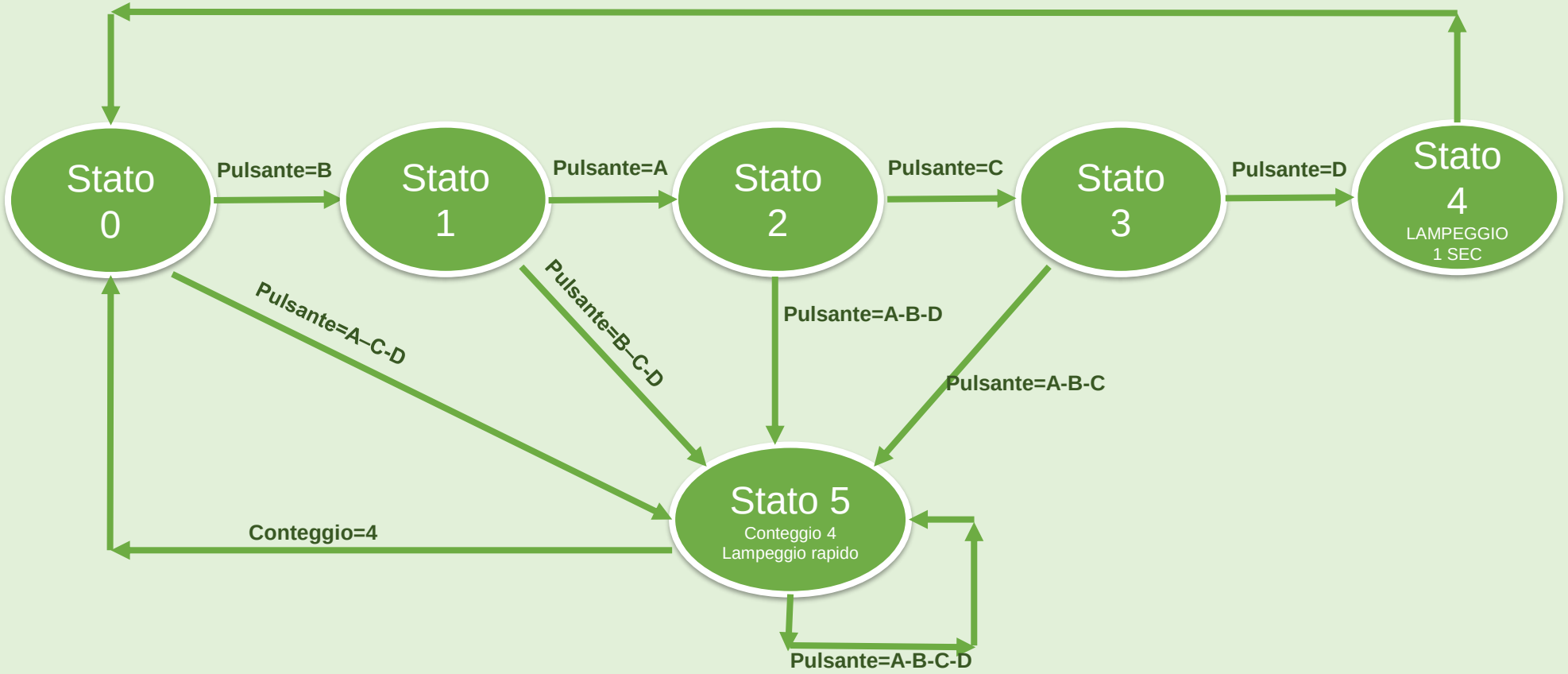
Stato 4 = accensione e spegnimento led

Stato 5 = pulsante errato



PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Il collegamento su breadboard è quello in figura, ed il programma dovrà prevedere i seguenti 6 stati.



PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Proviamo a tradurre il grafico precedente in programma.

Innanzitutto definiamo due variabili, una per memorizzare lo stato in cui si trova il sistema, l'altra per contare 4 pressioni consecutive di pulsanti errati.

Nel ciclo setup vanno configurati in ingresso ed uscita i pin utilizzati e vanno Inizializzati i valori delle due variabili.

```
#define LEDVERDE 13
#define P1 2
#define P2 3
#define P3 4
#define P4 5

//sequenza pulsante P2-P1-P3-P4
int stato,cont;

void setup() {
  pinMode(P1,INPUT);
  pinMode(P2,INPUT);
  pinMode(P3,INPUT);
  pinMode(P4,INPUT);
  pinMode(LEDVERDE,OUTPUT);
  stato=0;
  cont=0;
}
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Nel ciclo loop troveremo subito un ciclo switch-case sulla variabile stato ed all'interno dei singoli stati verranno testati i pulsanti, il pulsante corretto causerà un passaggio allo stato successivo, un pulsante errato incrementerà il contatore e causerà il passaggio allo stato 5 (stato di errore).

Se la sequenza è giusta, ci sarà un passaggio dallo stato 0 allo stato 4 dove il led verrà acceso per un secondo per poi tornare allo stato 0.

```
void loop() {
   switch(stato){
    case 0:
        if(digitalRead(P2)) {stato=1; while(digitalRead(P2)){} }
        if(digitalRead(P1)) {stato=5; cont++; while(digitalRead(P1)){} }
        if(digitalRead(P3)) {stato=5; cont++; while(digitalRead(P3)){} }
        if(digitalRead(P4)) {stato=5; cont++; while(digitalRead(P4)){} }
        break;
    case 1:
        if(digitalRead(P1)) {stato=2; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {stato=5; cont++; while(digitalRead(P2)){} }
        if(digitalRead(P3)) {stato=5; cont++; while(digitalRead(P3)){} }
        if(digitalRead(P4)) {stato=5; cont++; while(digitalRead(P4)){} }
        break;
    case 2:
        if(digitalRead(P3)) {stato=3; while(digitalRead(P3)){} }
        if(digitalRead(P1)) {stato=5; cont++; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {stato=5; cont++; while(digitalRead(P2)){} }
        if(digitalRead(P4)) {stato=5; cont++; while(digitalRead(P4)){} }
        break;
    case 3:
        if(digitalRead(P4)) {stato=4; while(digitalRead(P4)){} }
        if(digitalRead(P1)) {stato=5; cont++; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {stato=5; cont++; while(digitalRead(P2)){} }
        if(digitalRead(P3)) {stato=5; cont++; while(digitalRead(P3)){} }
        break;
    case 4:
        digitalWrite(LEDVERDE,HIGH);    delay(1000);
        digitalWrite(LEDVERDE,LOW);     stato=0;
        break;
    case 5:
        if(digitalRead(P1)) {cont++; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {cont++; while(digitalRead(P2)){} }
        if(digitalRead(P3)) {cont++; while(digitalRead(P3)){} }
        if(digitalRead(P4)) {cont++; while(digitalRead(P4)){} }
        break;
   }//switch
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Dopo il ciclo switch-case sempre all'interno del loop, va inserito il controllo del conteggio della pressione dei pulsanti. Se il conteggio ha raggiungo il valore 4, vuol dire che sono stati premuti 4 pulsanti di cui almeno uno sbagliato.

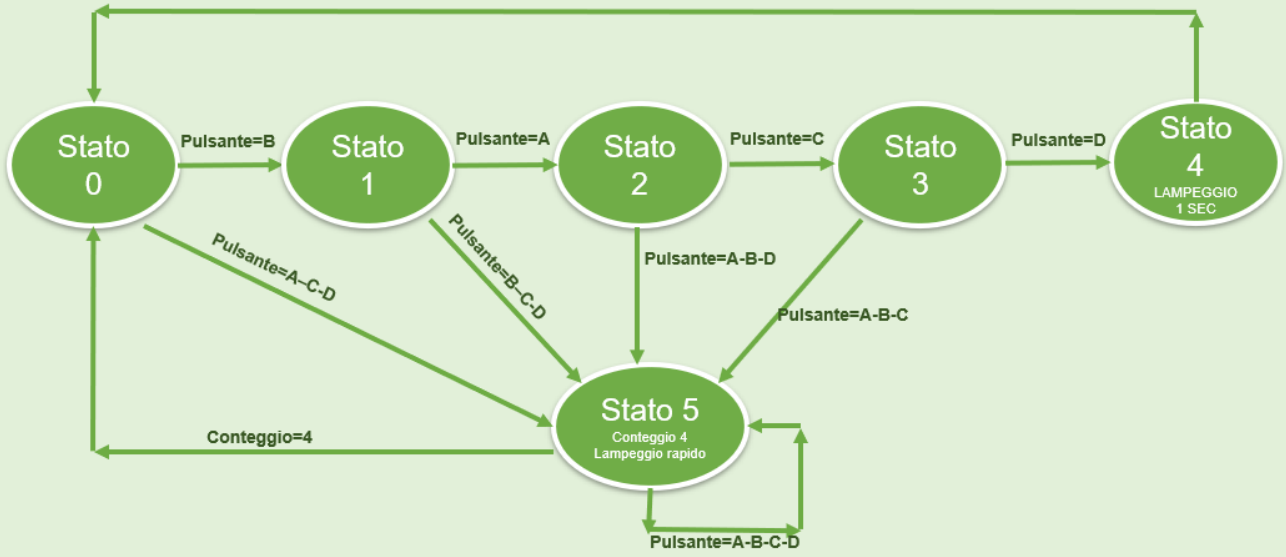
In questo caso si avrà il doppio lampeggio rapido e il passaggio allo stato 0.

```
//codice errato
if(cont==4){
      digitalWrite(LEDVERDE,HIGH);
      delay(200);
      digitalWrite(LEDVERDE,LOW);
      delay(200);
      digitalWrite(LEDVERDE,HIGH);
      delay(200);
      digitalWrite(LEDVERDE,LOW);
      cont=0;
      stato=0;
 }//if
```

PROGRAMMAZIONE A STATI FINITI CON ARDUINO

Un'ottimizzazione può essere quella di inserire il controllo del conteggio dentro al case 5.

Nell'immagine il ciclo loop completo che realizza esattamente la situazione indicata sotto con i 6 stati.




```
void loop() {
   switch(stato){
    case 0:
        if(digitalRead(P2)) {stato=1; while(digitalRead(P2)){} }
        if(digitalRead(P1)) {stato=5; cont++; while(digitalRead(P1)){} }
        if(digitalRead(P3)) {stato=5; cont++; while(digitalRead(P3)){} }
        if(digitalRead(P4)) {stato=5; cont++; while(digitalRead(P4)){} }
        break;
    case 1:
        if(digitalRead(P1)) {stato=2; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {stato=5; cont++; while(digitalRead(P2)){} }
        if(digitalRead(P3)) {stato=5; cont++; while(digitalRead(P3)){} }
        if(digitalRead(P4)) {stato=5; cont++; while(digitalRead(P4)){} }
        break;
    case 2:
        if(digitalRead(P3)) {stato=3; while(digitalRead(P3)){} }
        if(digitalRead(P1)) {stato=5; cont++; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {stato=5; cont++; while(digitalRead(P2)){} }
        if(digitalRead(P4)) {stato=5; cont++; while(digitalRead(P4)){} }
        break;
    case 3:
        if(digitalRead(P4)) {stato=4; while(digitalRead(P4)){} }
        if(digitalRead(P1)) {stato=5; cont++; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {stato=5; cont++; while(digitalRead(P2)){} }
        if(digitalRead(P3)) {stato=5; cont++; while(digitalRead(P3)){} }
        break;
    case 4:
        digitalWrite(LEDVERDE,HIGH);   delay(1000);
        digitalWrite(LEDVERDE,LOW);    stato=0;
        break;
    case 5:
        if(digitalRead(P1)) {cont++; while(digitalRead(P1)){} }
        if(digitalRead(P2)) {cont++; while(digitalRead(P2)){} }
        if(digitalRead(P3)) {cont++; while(digitalRead(P3)){} }
        if(digitalRead(P4)) {cont++; while(digitalRead(P4)){} }
        if(cont==4){  //codice errato
          digitalWrite(LEDVERDE,HIGH);  delay(200);   digitalWrite(LEDVERDE,LOW);    delay(200);
          digitalWrite(LEDVERDE,HIGH);  delay(200);   digitalWrite(LEDVERDE,LOW);
          cont=0;       stato=0;
        }//if
        break;
   }//switch
}//loop
```