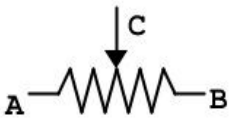
Supponiamo ora di voler cambiare la frequenza di lampeggio dei led, dovremo aprire lo sketch precedente variare le due costanti intervalloRosso e intervalloVerde, ricompilare e dopo esserci collegati col pc alla scheda rifare il trasferimento. Così sarà per tutte le volte che vogliamo fare una modifica. Come possiamo ovviare a questo? Con due potenziometri come quelli per regolare il volume di una radio ed usando due ingressi analogici di Arduino.

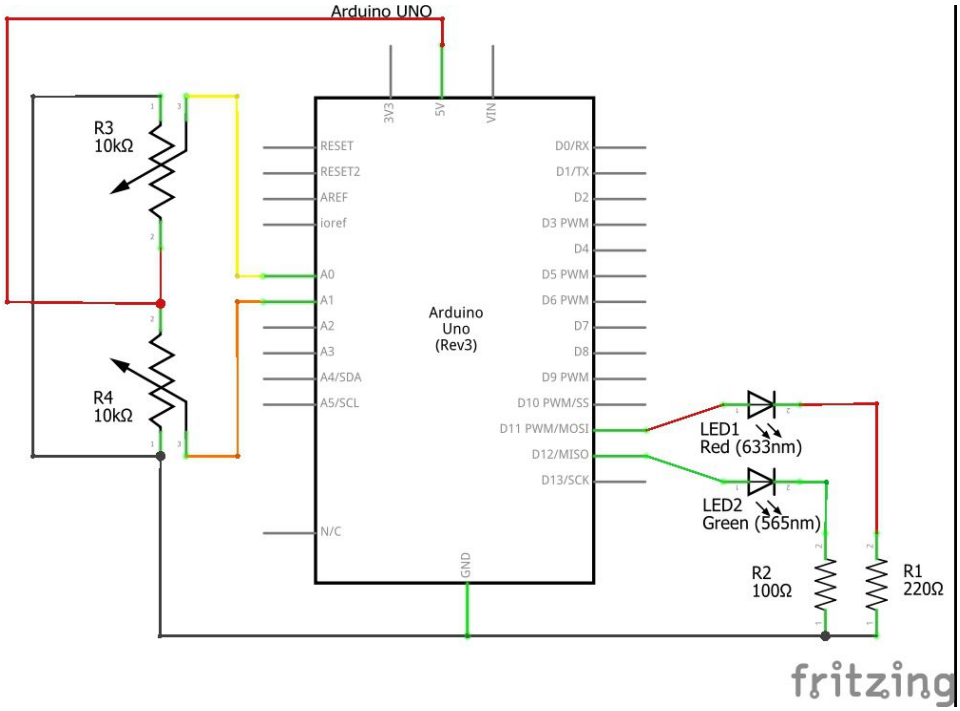
Il potenziometro o resistenza variabile come si può vedere nella figura qui sotto:



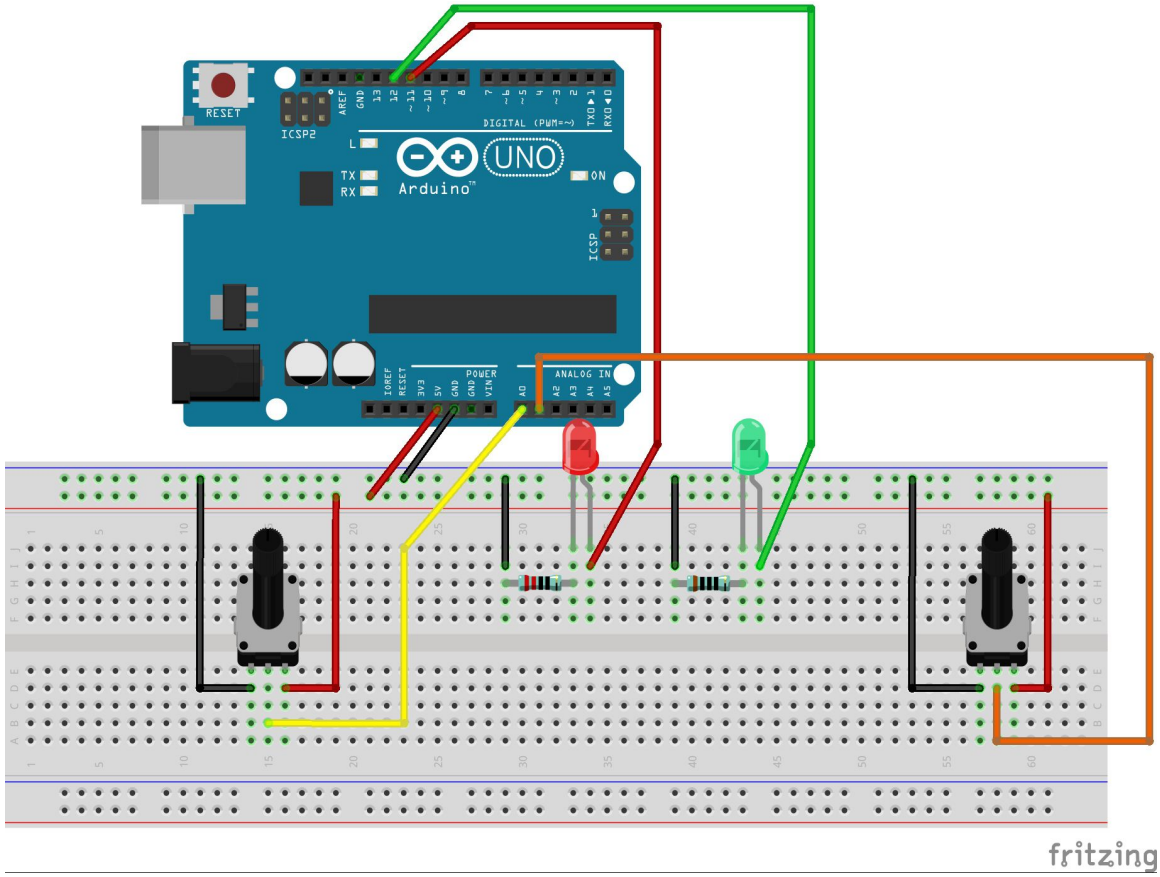
Simbolo resistore variabile,

A- B reofori dell'elemento resistivo
C reoforo del cursore

è simile ad una resistenza normale con l'aggiunta di un contatto mobile, per cui semplificando:
se applichiamo al contatto A la tensione di +5V e a B 0V (gnd) quando il contatto C sarà alla minima distanza da A avremo +5V quando al contrario sarà vicino a B avremo circa 0V in tutte le posizioni intermedie avremo una tensione proporzionale alla distanza da A. In virtù del funzionamento di questo potenziometro potremo ottenere delle variazioni lineari di tensione cioè una variazione analogica adatta quindi ad essere collegata ai pin analogici di Arduino.
Abbiamo già detto che fornendo una tensione continua tra 0V e +5V ad uno di questi pin il valore viene trasformato in un numero che varia tra 0 e 1023 cioè un numero a 10 bit e la variazione minima di tensione che viene rilevata (risoluzione) è: 5 / 1024 = 0.00488 V circa 4.9 mV. Vediamo lo schema



e il circuito sulla breadboard



analizziamo i cambiamenti del programma alla pagina successiva
Ho definito due nuove costanti per i due potenziometri: A0 (pin A0) per il rosso e A1 (pin A1) per il verde. Nel ***setup*** non ho dichiarato se questi pin sono ***INPUT*** o ***OUTPUT*** perché da A0 ad A5 possono essere solamente in lettura e quindi ***INPUT***. Nelle prime due righe del loop eseguo la lettura del valore presente sui due pin con ***analogRead(pin_analogico)*** e scrivo direttamente questo valore nelle due variabili che determinano la frequenza di accensione dei due led e questo è tutto.

```
/* 2 led collegati ad arduino programma 4: lampeggio a frequenza diversa
 *  controllata da potenzionetri collegati a pin analogici
 *  per fare ciò uso la funzione interna millis() che ad ogni istante restituisce
 *  il valore in millisecondi del tempo trascorso dall'ultima
 *  accensione o reset della scheda
 *
 *  led rosso collegato a pin 11
 *  led verde collegato a pin 12
 */
const int ledRosso = 11;
const int ledVerde = 12;
const int potRosso = A0;
const int potVerde = A1;

unsigned long intervalloRosso = 800;            // definisco i tempi iniziali
unsigned long intervalloVerde = 300;            // di lampeggio di ogni led

unsigned long timerRosso;                       // variabile per memorizzare il tempo trascorso
unsigned long timerVerde;                       // una per ogni led

void setup() {
  pinMode(ledRosso,OUTPUT);
  pinMode(ledVerde,OUTPUT);
  timerRosso= millis();                         // inizializzo le due variabili
  timerVerde= millis();                         // con il tempo corrente in millisecondi
}

void loop() {
  intervalloRosso = analogRead(potRosso);       // ridefinisco gli intervalli da 0 a 1023 millisecondi
  intervalloVerde = analogRead(potVerde);       // per entrambi i led

  //Gestisco il primo led (rosso)
  if ((millis()- timerRosso) >= intervalloRosso) // se è passato + tempo di quello previsto per l'attesa
    invertiRosso();                              // chiamo la funzione invertiRosso

  //gestisco il secondo led (verde)
  if ((millis()- timerVerde) >= intervalloVerde) // se è passato + tempo di quello previsto per l'attesa
    invertiVerde();                              // chiamo la funzione invertiVerde
}

/* scrivo qui le funzioni esterne
 * che richiamo dal loop
 */
void invertiRosso() {
  if (digitalRead(ledRosso) ==LOW)              // se il led è spento
      digitalWrite(ledRosso,HIGH);              // lo accendo
  else                                          // altrimenti
      digitalWrite(ledRosso,LOW);               // lo spengo
  timerRosso=millis();                          // memorizzo quando ho cambiato lo stato del led
}

void invertiVerde() {
  if (digitalRead(ledVerde) ==LOW)              // se il led è spento
      digitalWrite(ledVerde,HIGH);              // lo accendo
  else                                          // altrimenti
      digitalWrite(ledVerde,LOW);               // lo spengo
  timerVerde=millis();                          // memorizzo quando ho cambiato lo stato del led
}
```

Con questo metodo possiamo definire un lampeggio che arriva al massimo ad 1 secondo e 23 millesimi di tempo, se vogliamo coprire un arco temporale più esteso, un metodo è quello di eseguire delle proporzioni matematiche per trovare i nuovi valori ed uno più semplice è quello di usare una funzione interna di Arduino che è:

map(valore, da_lim_inf, a_lim_sup, nuovo_da_lim_inf, nuovo_a_lim_sup)

cioè il valore da noi trovato che va da 0 a 1023 lo trasformiamo in un altro valore usando una scala

diversa, ad esempio se definiamo che il ritmo per il led rosso deve arrivare al valore massimo di 5 secondi (5000 millisecondi) scriveremo:

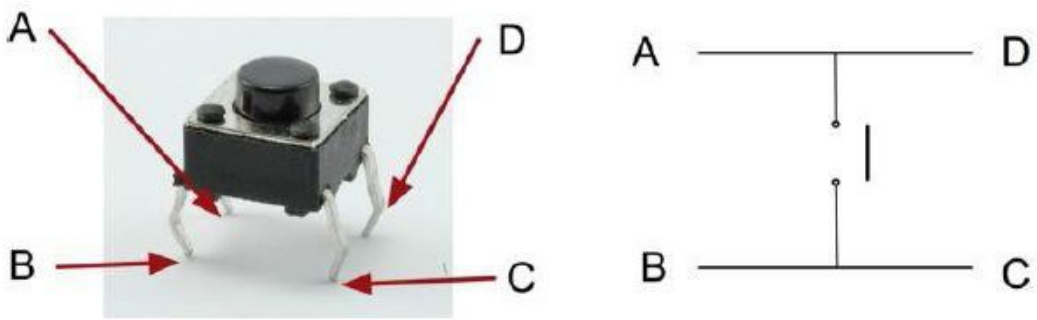
intervalloRosso = ***map(valore, 0, 1023, 0, 5000)***

dove valore sarà la lettura analogica di potRosso

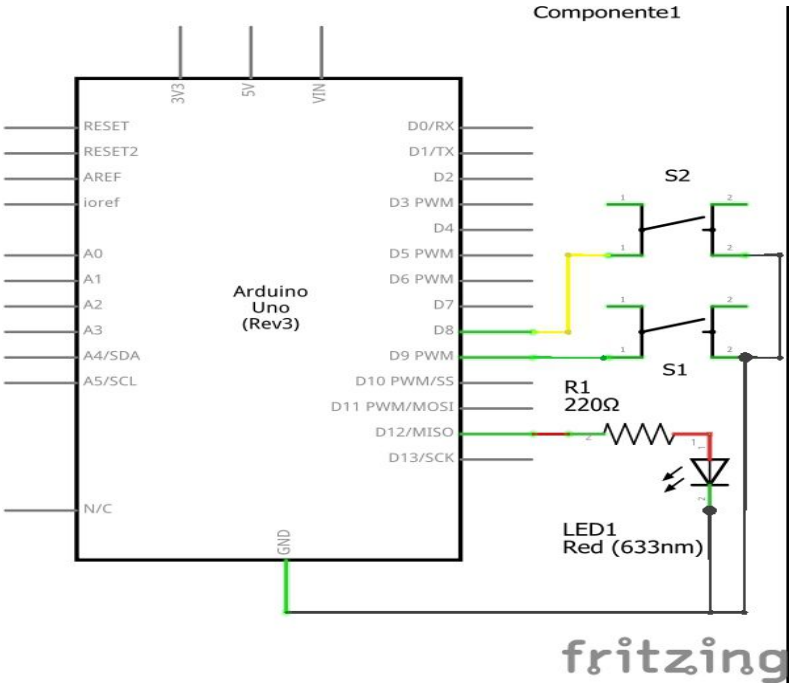
```
1  /* 2 led collegati ad arduino programma 5: lampeggio a frequenza diversa
9
10 const int ledRosso = 11;
11 const int ledVerde = 12;
12 const int potRosso = 0;
13 const int potVerde = 1;
14
15 unsigned long intervalloRosso = 800; //definisco i tempi iniziali di lampeggio di ogni led
16 unsigned long intervalloVerde = 300;
17
18 unsigned long timerRosso; // variabile per memorizzare il tempo trascorso
19 unsigned long timerVerde; // 1 per ogni led
20
21 void setup() {
27
28 void loop() {
29   int valPotRosso = analogRead(potRosso);
30   int valPotVerde = analogRead(potVerde);
31
32   intervalloRosso = map(valPotRosso, 0, 1023, 0, 5000);
33   intervalloVerde = map(valPotVerde, 0, 1023, 0, 5000);
34
35   //Gestisco il primo led (rosso)
36   if ((millis()- timerRosso) >= intervalloRosso) // se è passato + tempo di quello previsto per l'attesa
37     invertiRosso();                              // chiamo la funzione invertiRosso
38
39   //gestisco il secondo led (verde)
40   if ((millis()- timerVerde) >= intervalloVerde)  // se è passato + tempo di quello previsto per l'attesa
41     invertiVerde();                               // chiamo la funzione invertiVerde
42 }
43
44 // scrivo qui le funzioni esterne che richiamo dal loop
45 void invertiRosso() {
52
53 void invertiVerde() {
```

alle righe 29 e 30 ho definito due nuove variabili per salvare il valore dei potenziometri che verranno poi rimappati alle righe 32 e 33.

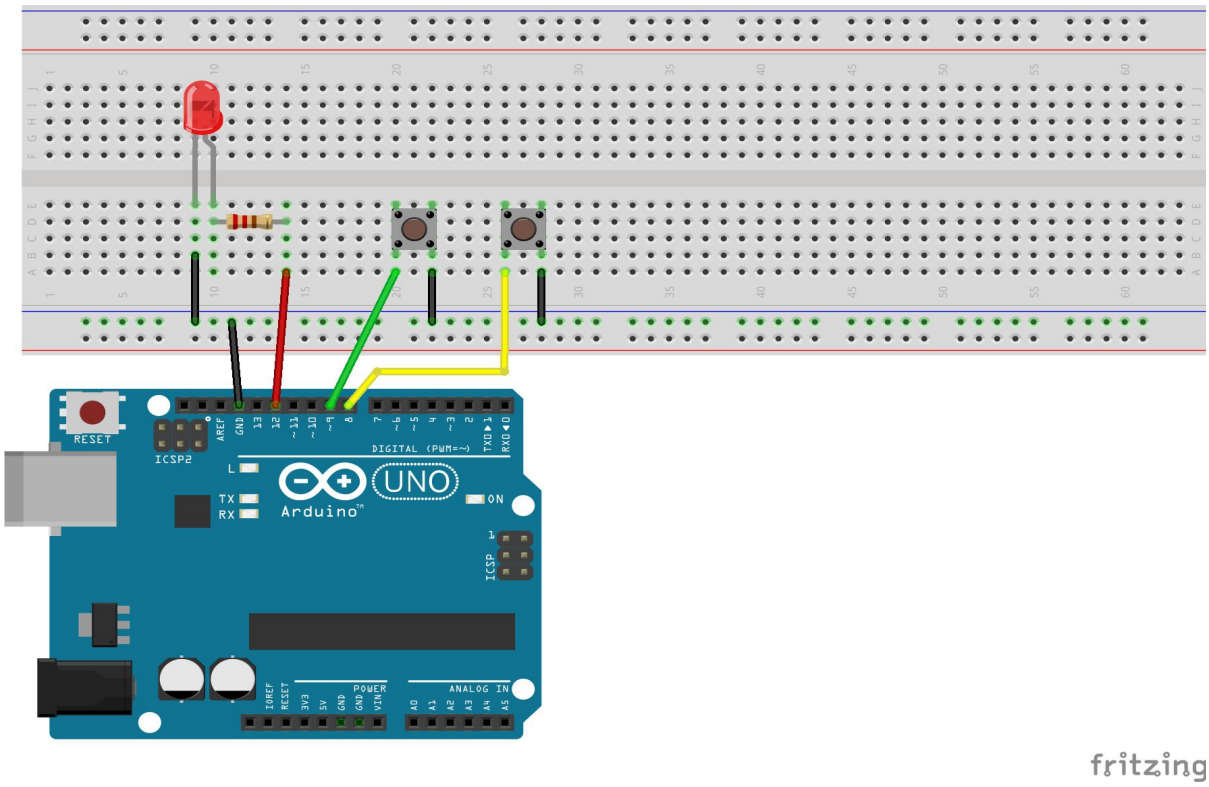
Costruiamo ora un semplice circuito per accendere e spegnere un led usando due pulsanti. Per fare questo prendiamo 2 pulsanti dal kit e come si può notare hanno quattro contatti ma questi sono collegati a coppie come dall'immagine seguente per cui bisogna prestare attenzione al loro posizionamento sulla breadboard



quindi realizziamo il seguente circuito:



che sulla breadboard diventa:



ed ora scriviamo il programma.

```
/* 1 led 2 pulsanti versione 1
 *  il led è collegato al pin 12
 *  i due pulsanti al pin 9 e pin 8
 *  l'altro contatto è verso massa
 */
int ledPin = 12;
int buttonApin = 9;
int buttonBpin = 8;

void setup(){
  pinMode(ledPin, OUTPUT);              // il pin del led è OUTPUT
  pinMode(buttonApin, INPUT_PULLUP);    // i pin dei pulsanti sono INPUT e usando
  pinMode(buttonBpin, INPUT_PULLUP);    // la loro resistenza interna vanno a +5V
}

void loop(){
  if (digitalRead(buttonApin) == LOW){  // se il pulsante A è premuto porto a 0V il pin, il pin sarà LOW
    digitalWrite(ledPin, HIGH);         // quindi accendo il led
  }
  if (digitalRead(buttonBpin) == LOW){  // se il pulsante B è premuto porto a 0V il pin, il pin sarà LOW
    digitalWrite(ledPin, LOW);          // quindi spengo il led
  }
}
```