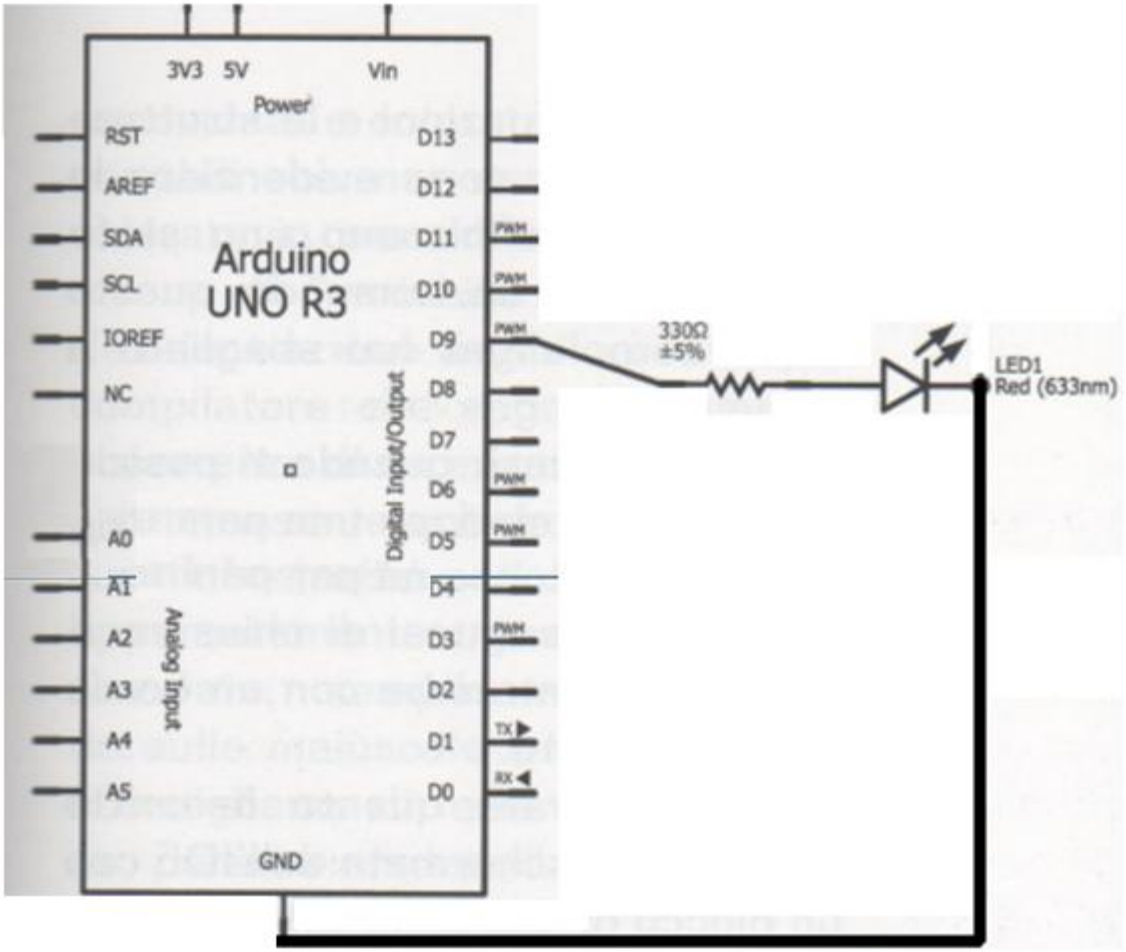
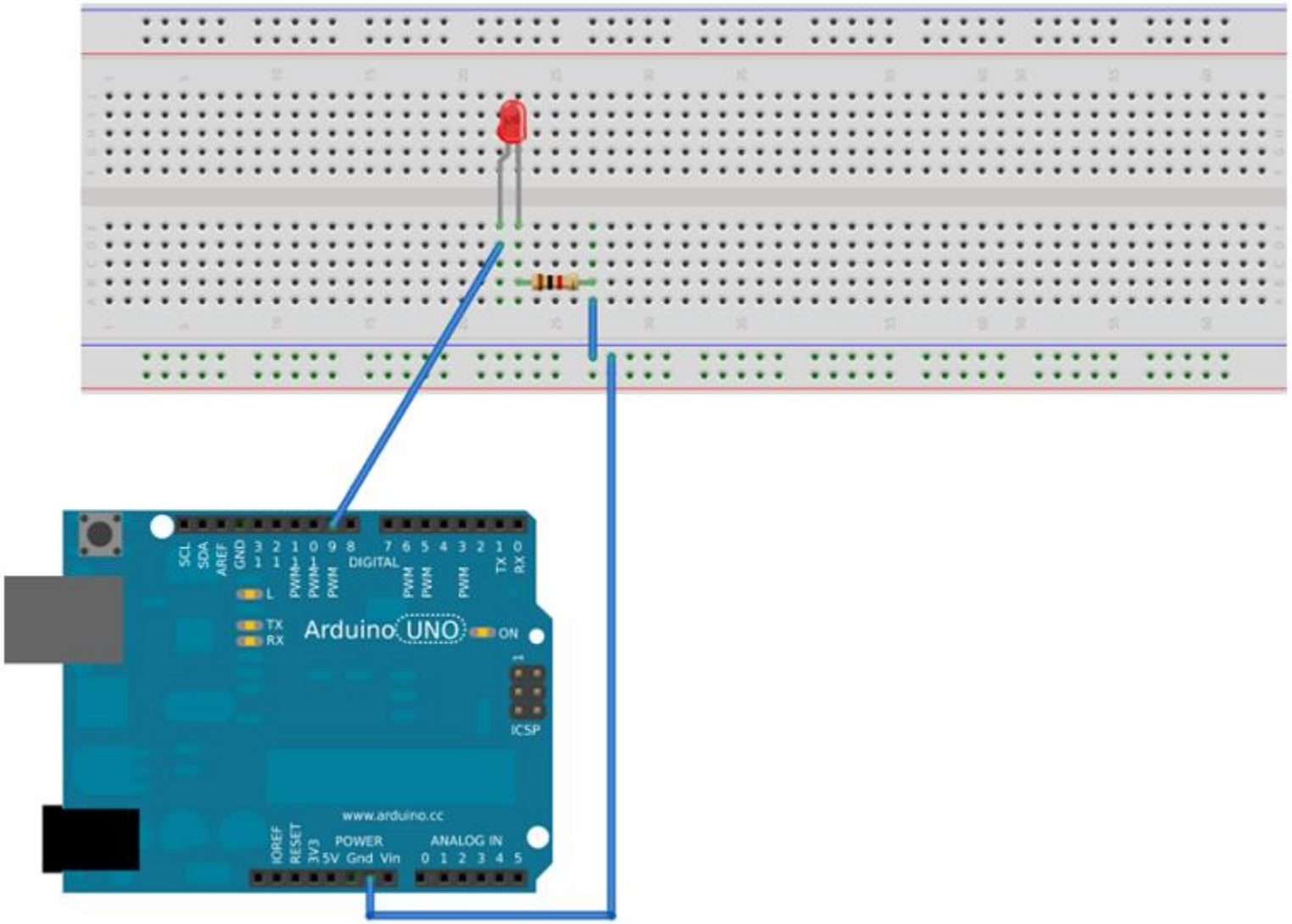
ACCENDERE E SPEGNERE UN DIODO LED



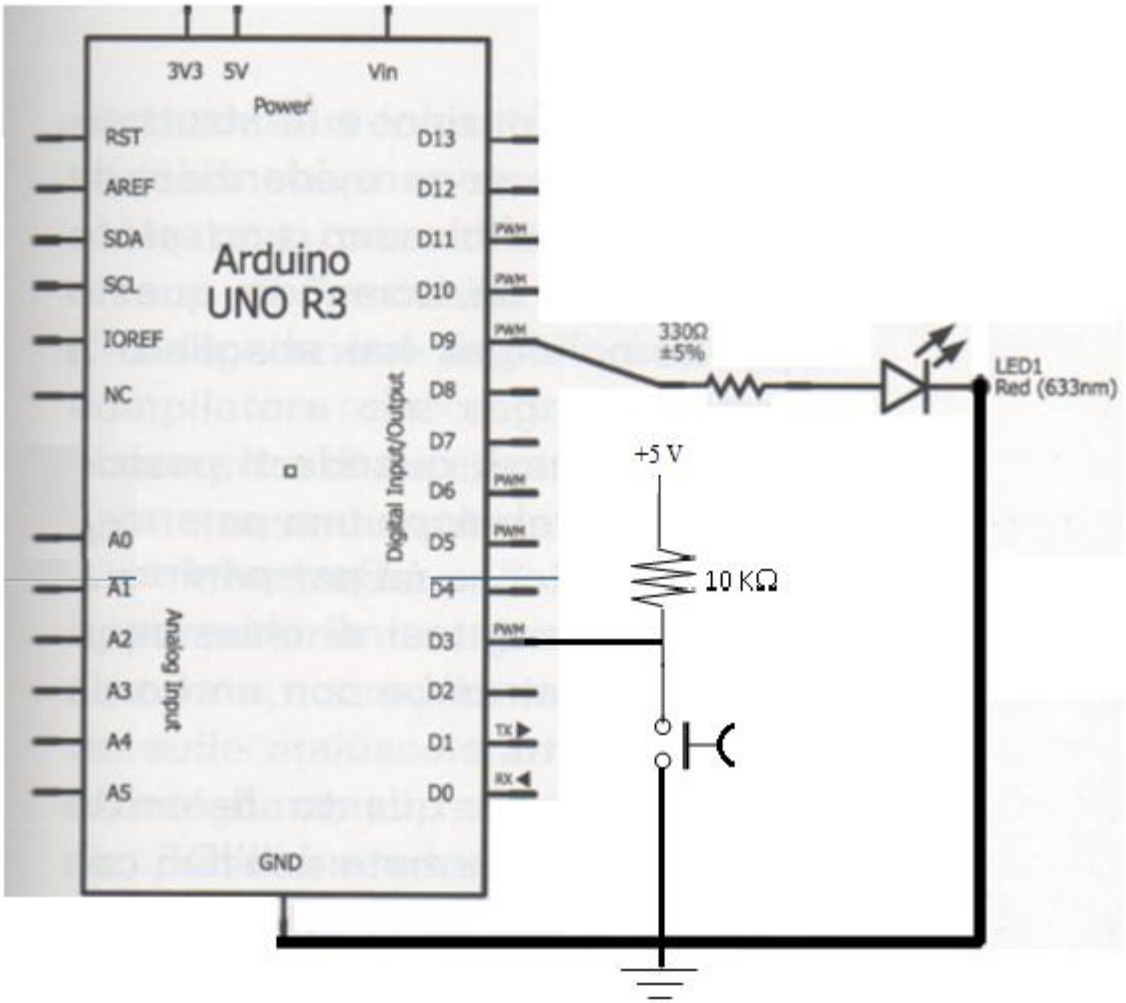
ACCENDERE E SPEGNERE UN DIODO LED



Accendere e spegnere un diodo LED

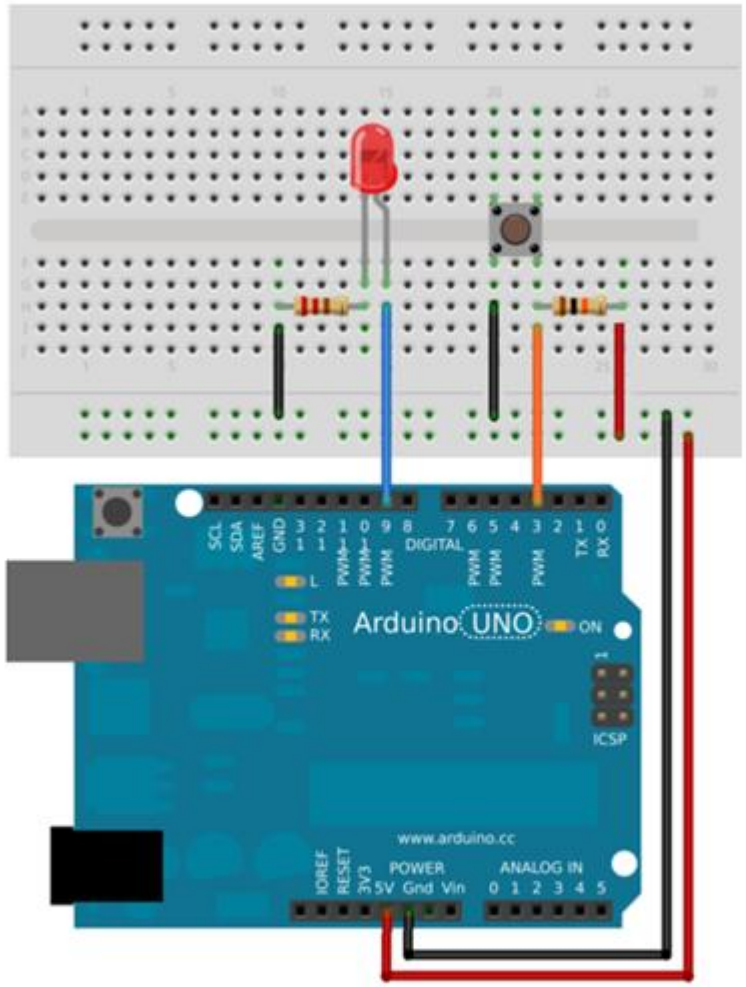
```
/*
  Blink
  Turns on a LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 9;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Accendere e spegnere un diodo LED con un pulsante



Accendere e spegnere un diodo LED con un pulsante

Schema elettrico



ACCENDERE E SPEGNERE UN DIODO LED CON UN PULSANTE

```
int led = 9;                          // # define led 9
int pul = 3;                          // # define pul 3
 // impostiamo i due pin digitali come output e input
void setup()
            { pinMode(led, OUTPUT);
              pinMode(pul, INPUT); }

void loop()
{
If (digitalRead(pul)== LOW)           // se è premuto il pulsante
       {  digitalWrite(led, HIGH); }  // accende il LED
else { digitalWrite(led, LOW);}       // altrimenti spegne il LED
}
```

Il programma legge a ogni ciclo lo stato del pulsante e, se questo è a livello basso vuol dire che è stato pigiato e accende il led altrimenti lo spegne

VARIANTE: COMMUTAZIONE CON IL PULSANTE

E' possibile accendere e spegnere alternativamente il LED ad ogni pressione del tasto. Per far questo occorre modificare lo sketch in questo modo:

```
int led = 9;
int pul = 3;

int stato_pul=1;
Int stato_led=1;

void setup()
            { pinMode(led, OUTPUT);
              pinMode(pul, INPUT); }

void loop()
{  stato_pul= digitalRead(pul);
   If (stato_pul==LOW)          // se è premuto il pulsante
       {  stato_led=1-stato_led;
          if (stato_led ==1) { digitalWrite(led, HIGH); } // accende il LED
          else { digitalWrite(led, LOW);}          // altrimenti spegne il LED
          delay(500);
       }
}
```

Il programma a ogni ciclo legge lo stato del pulsante e se questo è attivo , inverte lo stato del led. Il ritardo introdotto alla fine consente che i rimbalzi del pulsante possano esaurirsi. Però, se si tiene premuto il tasto oltre il mezzo secondo, il led continua a cambiare stato ogni mezzo secondo.

VARIANTE: COMMUTAZIONE CON IL PULSANTE

Per risolvere questo inconveniente occorre affidare il controllo del led alla transizione negativa del pulsante dallo stato alto a quello basso e non al permanere del pulsante allo stato basso. Per stabilire se è avvenuta la transizione occorre confrontare lo stato corrente del pulsante con quello precedente. In tal modo solo se lo stato attuale è bassoe lo stato precedente è alto allora si è verificata una transizione negativa e deve cambiare lo stato del led.

```
int led = 9;
int pul = 3;

int stato_att=1;
int stato_pre=1;
Int stato_led=0;
void setup()
            {pinMode(pul, INPUT); pinMode(led, OUTPUT); }

void loop()
{  stato_att= digitalRead(pul);
   If ((stato_att==LOW)  &&(stato_pre==HIGH))
       {  stato_led=1-stato_led;
          if (stato_led ==1) { digitalWrite(led, HIGH); }
              else { digitalWrite(led, LOW);}
        } stato_pre=stato_att;
          delay(50);
}
```

Variare luminosità di un LED tenendo premuto un pulsante

```
int led = 9;
int pul = 3;

int stato_pul=1;
int lum=0;
Int durata=0;

void setup()
            {pinMode(pul, INPUT); pinMode(led, OUTPUT); }

void loop()
{  stato_pul= digitalRead(pul);
   If (stato_pul==LOW)
        {  lum++;
             If (lum>255) { lum=0; durata=1000; }
              else { durata=20; }
          analogWrite(led,lum);
            delay(durata);
         }
}
```

Leggere un segnale Analogico

Acquisire un segnale analogico ti permette di utilizzare Arduino per interagire con il mondo esterno non più in bianco o nero (0 e 1) ma con 1024 sfumature di grigio.

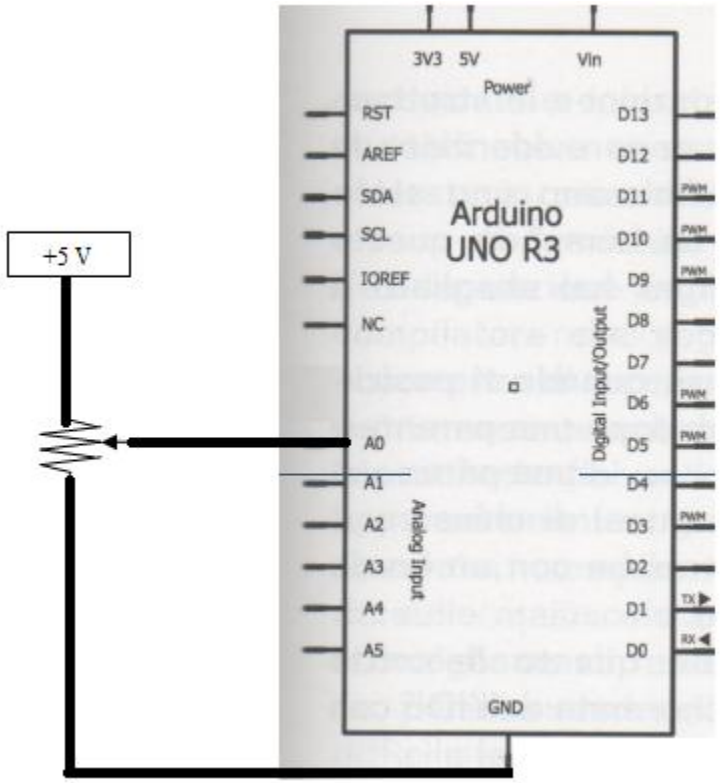
Ecco come si fa a leggere un segnale analogico e dove potrai leggere quei valori, partiamo con lo schema:

Come si vede in figura il collegamento del potenziometro è molto semplice e richiede:

- una scheda Arduino
- un potenziometro da 10K hom

Occorre collegare gli estremi del potenziometro (quelli su cui è sempre presente una resistenza di 10K ohm indipendentemente dalla posizione della manopola) rispettivamente al +5V e Gnd di arduino, non è importante quale va al +5v e quale al pin Gnd.
Collegare, quindi, il terminale centrale del potenziometro al pin A0 (ossia il pin 0 analogico).

Leggere un segnale Analogico



Leggere un segnale Analogico

Il codice che utilizzeremo per i nostri esperimenti è il seguente:

```
/*  Commento
 AnalogReadSerial
 Reads an analog input on pin 0,  prints the result to the serial monitor
  */


void setup() {
  Serial.begin(9600); /* indico ad Arduino che utilizzerò la comunicazione
                         seriale a 9600 baud*/
}

void loop() {
  int sensorValue = analogRead(0);
  Serial.println(sensorValue, DEC);
}
```

Leggere un segnale Analogico

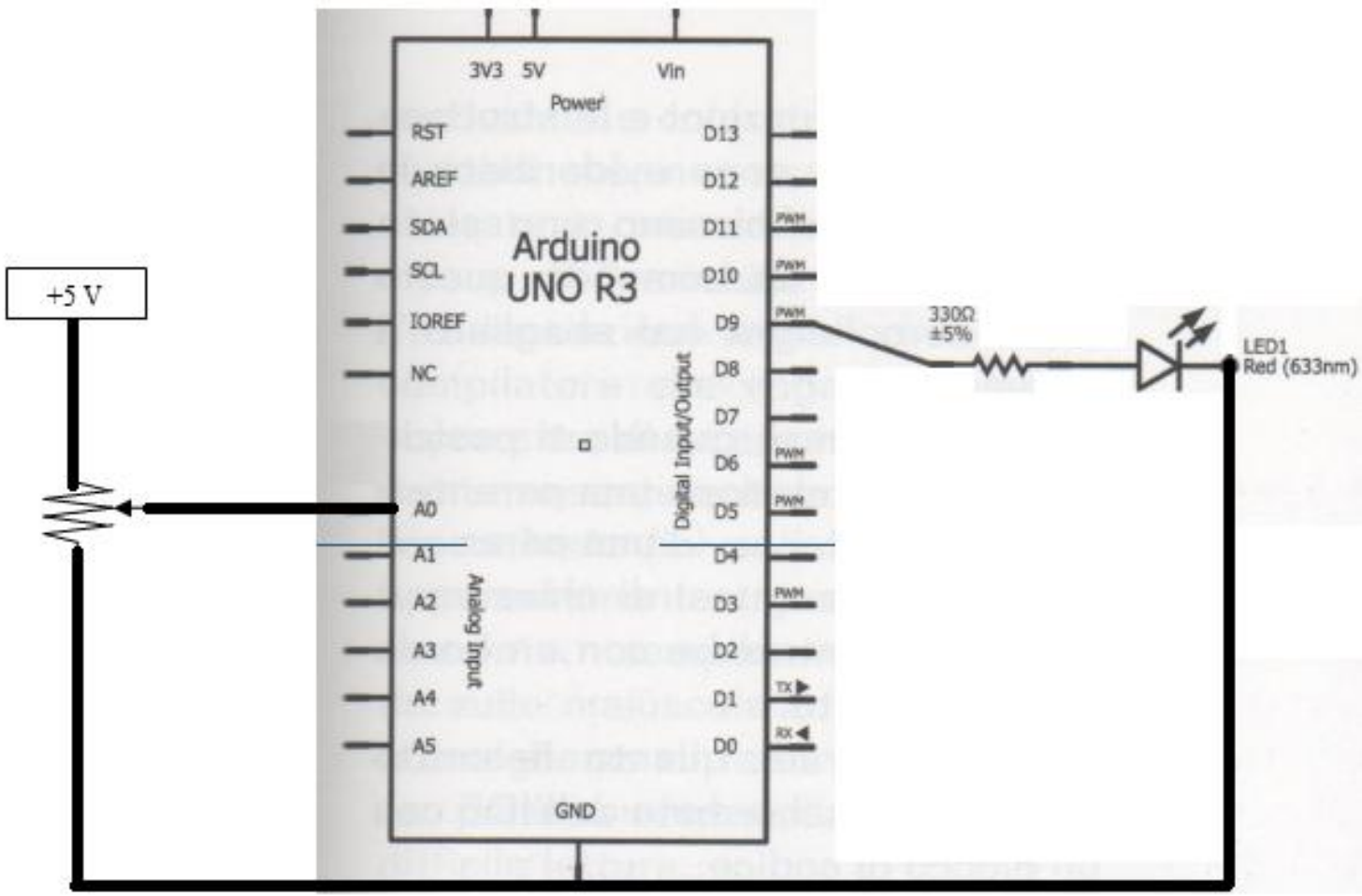
Osservazioni:

Utilizzando una sintassi compressa dichiaro la variabile **sensorValue** e ne imposto il valore con il risultato letto sul pin 0 analogico, per leggere il valore del pin 0 analogico utilizziamo il comando **analogRead(0)**,

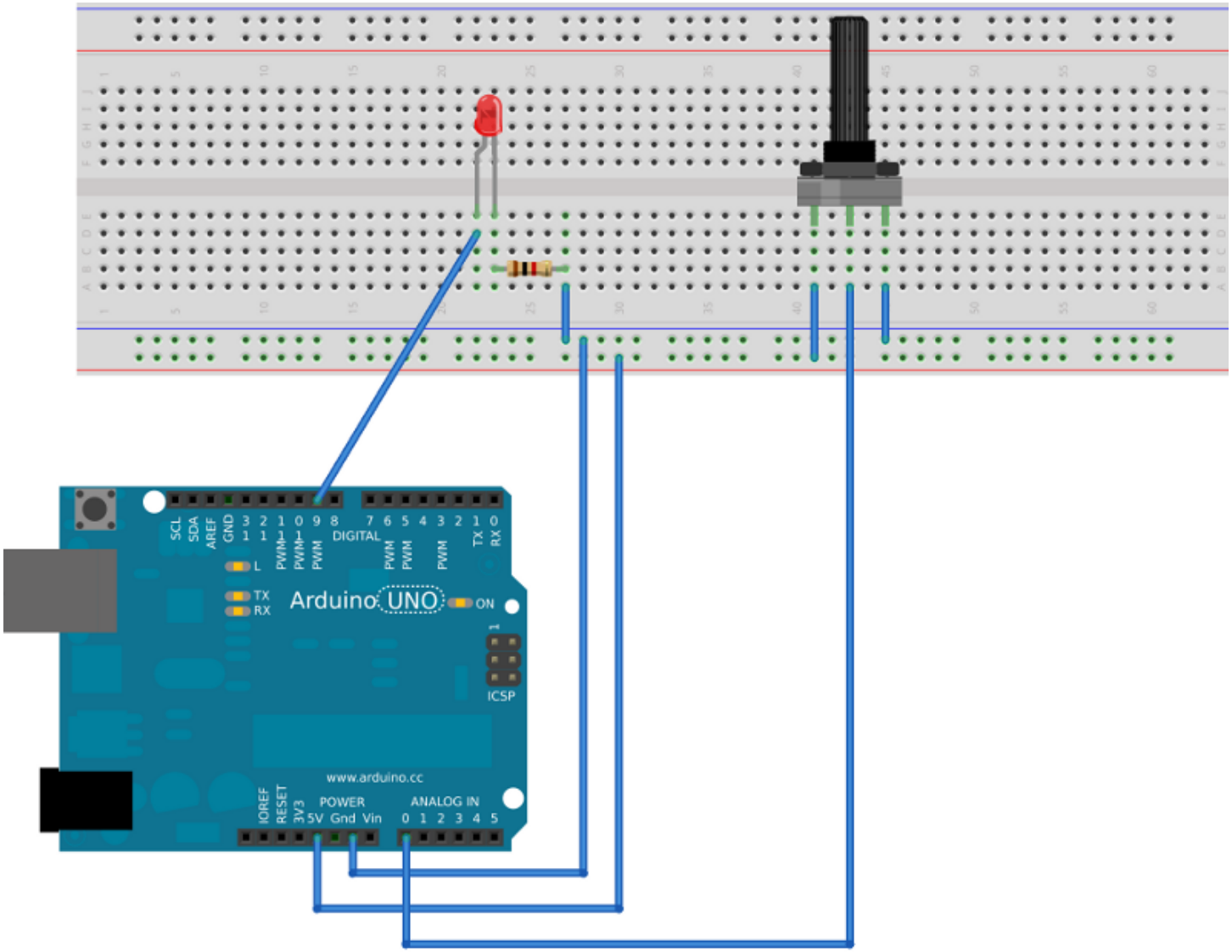
In questo comando per riferirci al pin 0 analogico (A0), il comando **analogRead** leggerà ad ogni ciclo della funzione *loop* il valore analogico rilevato sul pin 0.

Tale valore varia tra 0 e 1023.

Variare la luminosità di un led con un potenziometro



Variare la luminosità di un led con un potenziometro



VARIARE LA LUMINOSITÀ DI UN LED CON UN POTENZIOMETRO

Il codice che utilizzeremo per i nostri esperimenti è il seguente:

```
 int valore = 0;
 int POT = 0;                    // #define POT 0
 int LED = 3;                      // #define LED 3

 void setup()
{
    pinMode(Led, OUTPUT); // pin led
    Serial.begin(9600); // inizializza il collegamento seriale
 }

void loop() {
  valore= analogRead(POT);
  Serial.println(valore/4, DEC);
  analogWrite(LED,valore/4); delay(100);
}
```

Variare il ritmo di lampeggio di un LED con un potenziometro

Il codice che utilizzeremo per i nostri esperimenti è il seguente:

```
 int valore = 0;
 int POT = 0;                    // #define POT 0
 int LED = 3;                      // #define LED 3

 void setup()
{    pinMode(Led, OUTPUT); // pin led
     Serial.begin(9600); // inizializza il collegamento seriale
 }

void loop() {
  valore= analogRead(POT);
  digitalWrite(LED,HIGH);
  delay(valore);
  digitalWrite(LED,LOW);
  delay(valore);
}
```

Differenze tra SPI e I2C

SPI supporta la comunicazione full-duplex con un throughput molto più alto rispetto ad I2C. non è limitato a parole di 8 bit, in modo da poter inviare messaggi di ogni genere e con contenuti e scopi arbitrari. L'interfaccia SPI non richiede resistenze di pull-up, il che si traduce in un minore consumo di energia. Tuttavia, I2C è più semplice avendo meno linee il che significa che meno piedini sono richiesti per interfacciarsi ad un circuito intergrato. Quando si comunica con più di un dispositivo slave, I2c possiede il vantaggio di un indirizzamento in banda, al posto di avere una linea chip select per ogni slave. I2c supporta inoltre lo slave acknowledgment che significa che si è certi del dispositivo con cui si sta comunicando. Con SPI, un master può inviare dati a vuoto e non saperlo. In generale SPI è consigliato per applicazioni che comunicano lunghi flussi di dati e non solo parole come locazioni di memoria.

Approfondimenti