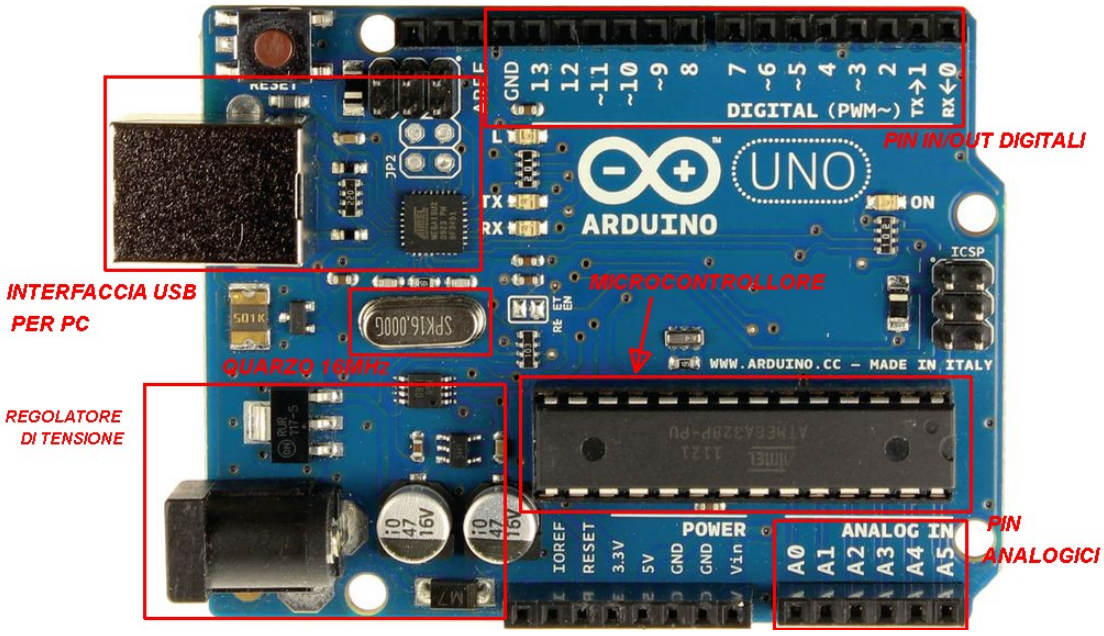
ARDUINO partendo da UNO

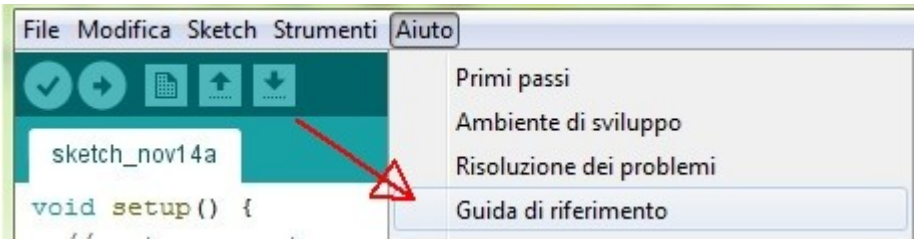
Dopo aver letto il primo tutorial “Arduino partendo da zero”, dovremo sapere qualcosa di più, per questo possiamo indicare questa seconda dispensa con “Arduino partendo da uno”, partiamo infatti dalla consapevolezza di cosa sia il progetto ARDUINO, e di come si possa iniziare a scrivere un semplice programma.
Quello che ci manca per realizzare qualche progetto più complesso, è conoscere qualche istruzione in più, come ad esempio quelle necessarie per la gestione dei pin digitali ed analogici.

Cominciamo con il ricordare che sulla scheda ARDUINO UNO, ci sono 14 pin digitali, numerati da 0 a 13, e 6 pin analogici numerati da A0 ad A5.



Ricordiamo inoltre che sulla scheda sono presenti le tensioni di 5Volt e 3,3Volt ed il riferimento di queste tensioni cioè GND corrispondente a 0Volt.

Ricordiamo anche che sull’I.D.E. *(Integrated Development Environment cioè ambiente di sviluppo integrato)* di Arduino, c’è una chiara e completa guida, accessibile dal menù Aiuto.



Innanzitutto per poter scrivere un pogramma per Arduino, occorre conoscere qualche rudimento di programmazione in linguaggio C o C++, ma al momento cerchiamo di rimanere nelle conoscenze minimali che ci servono per gestire i piedini digitali ed analogici.
Sulla scheda ARDUINO UNO, ed in genere in ogni scheda a microcontrollore, un piedino digitale può essere impostato come ingresso o come uscita;

- come ingresso se vogliamo collegarci un dispositivo digitale, che ha perciò due soli stati, 0 o 1, FALSE o TRUE.
- come uscita se vogliamo invece impostare lo stato di un disposizito digitale, che ha anch’esso due soli stati, 0 o 1, SPENTO o ACCESO.

Il classico esempio di un dispositivo digitale collegabile ad un ingresso, è il pulsante che può essere PREMUTO o NON PREMUTO, cioè due soli stati, ed il classico esempio di un dispositivo collegabile ad un’uscita, è il LED che può essere SPENTO o ACCESO.

La prima cosa da farè sarà quella di decidere quali dei 14 piedini disponibili sulla scheda ARDUINO UNO, gestiscono un ingresso e quali un uscita. Questa operazione viene fatta generalmente all'avvio, all'interno di quella parte di codice che viene eseguita una sola volta all'accensione della scheda e cioè il SETUP.

```
void setup() {

}
```

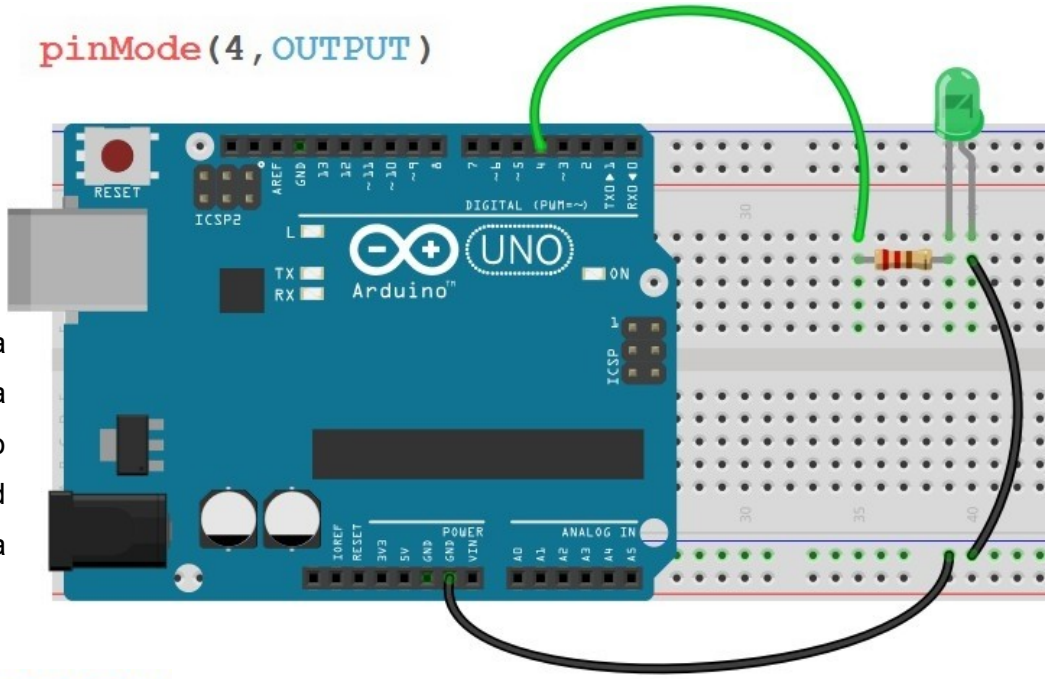
Per decidere se un piedino controllerà lo stato di un pulsante (INGRESSO) o imposterà lo stato di un led (USCITA), si utilizza la funzione **pinMode**.
Nell'esempio a destra vediamo che si è deciso di utilizzare il pin 4 come un'uscita ed il pin 5 come un ingresso.

```
void setup() {
    pinMode(4,OUTPUT);
    pinMode(5,INPUT);
}
```

Nella sintassi di questa funzione va indicato il numero del pin da configurare ed il modo in cui viene configurato.

PIN CONFIGURATO COME USCITA

In questo modo il pin 4 viene configurato in modo da poter accende o spegnere un dispositivo digitale di uscita come ad esempio un LED.



Successivamente se si vorrà accendere il LED bisogna mettere il pin 4 ad un livello logico alto, corrispondente ad una tensione di 5Volt con la funzione **digitalWrite.**

`digitalWrite(4,HIGH)` ***mette l'uscita 4 ad un livello alto (5Volt)per accendere il LED***

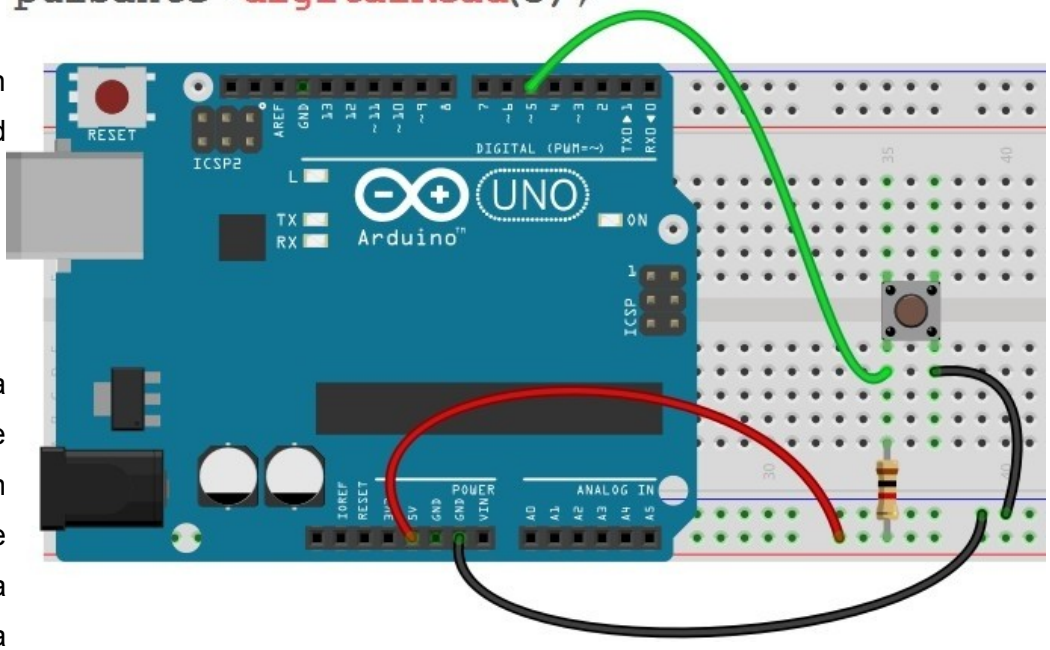
`digitalWrite(4,LOW)` ***mette l'uscita 4 ad un livello basso (0Volt) per spegnere il LED***

Il pin 4 configurato come uscita può erogare al massimo 40mA, pertanto non è possibile pilotare dispositivi di potenza che assorbono correnti maggiori.

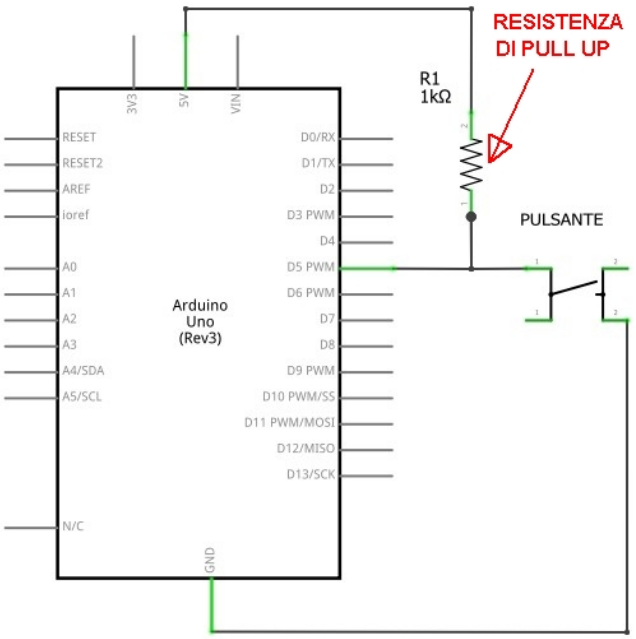
PIN CONFIGURATO COME INGRESSO

```
int pulsante;
pulsante= digitalRead(5);
```

In questo modo il pin 5 viene configurato in modo da poter leggere lo stato di un ingresso digitale, come ad esempio un pulsante.
Nel collegamento in figura se il pulsante viene premuto, il filo verde ed il nero andranno in contatto, e sul pin 5 verrà così portato il livello di tensione 0Volt. Se invece il pulsante non viene premuto, il filo verde collegato al pin 5, risulta collegato solo alla resistenza che porta un livello di tensione di 5Volt.



Visto che la resistenza serve a portare un livello di tensione alto quando il pulsante non è premuto, viene chiamata **resistenza di PULL UP**.



Con questo collegamento se il pulsante non è premuto sul pin 5 ci saranno 5Volt e cioè un livello logico alto.
Per poter leggere con l'istruzione **digitalRead** lo stato del pulsante occorre utilizzare una variabile che conterrà il risultato della lettura.

In questo modo se il pulsante **NON E'** premuto la variabile pulsante **vale 1**, se invece **E'** premuto la variabile pulsante **vale 0**.

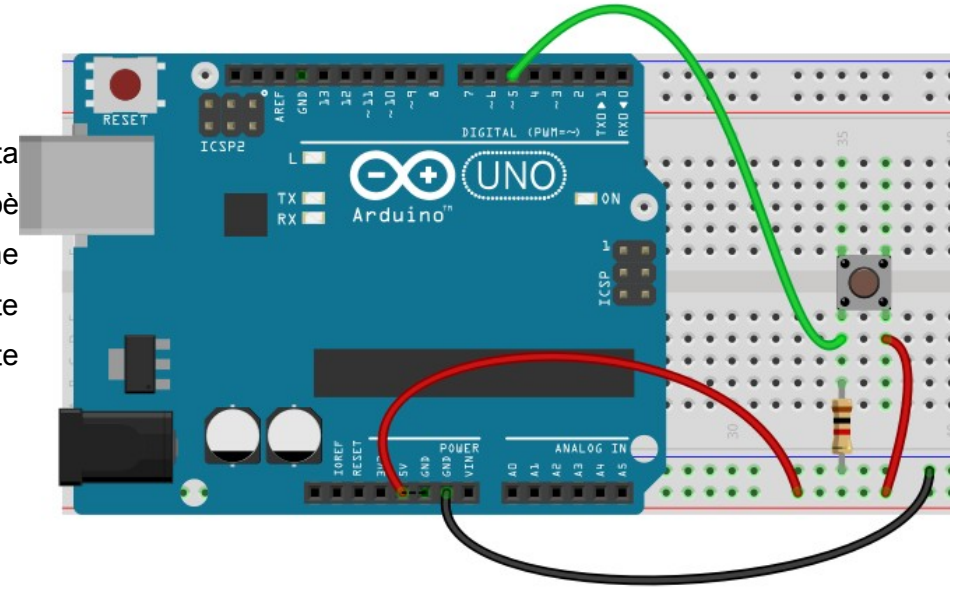
```
int pulsante;
pulsante= digitalRead (5);
```

Un altro modo di configurare un piedino in ingresso, è quello di utilizzare il modo INPUT_PULLUP.

```
pinMode(5, INPUT_PULLUP );
```

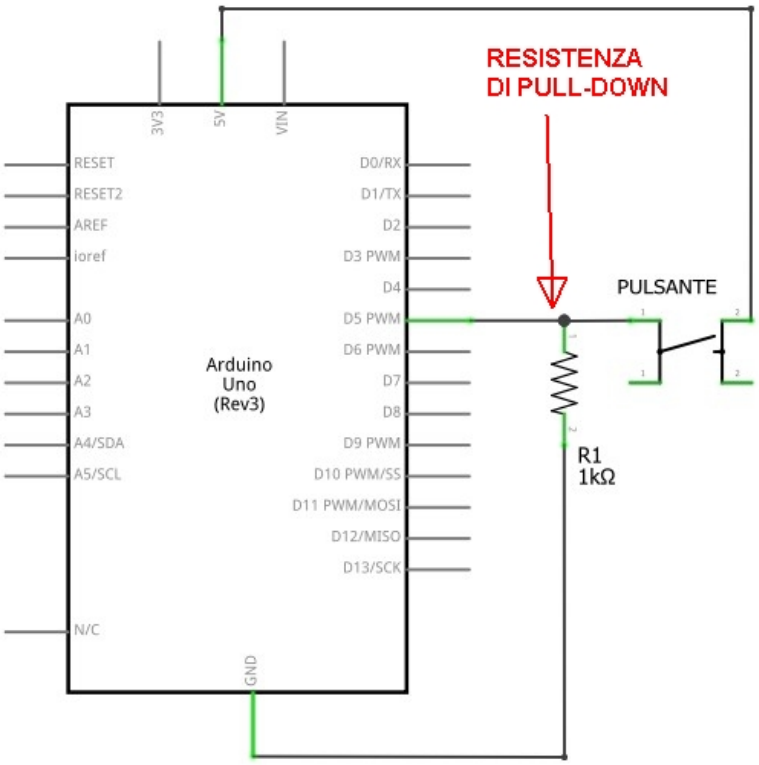
In questo modo viene definito un piedino come ingresso, con una resistenza di PULL-UP già collegata dentro la scheda ARDUINO UNO. Si utilizzerà perciò il primo schema SENZA la resistenza esterna di PULL-UP.

La resistenza può essere collegata anche in modalità **PULL-DOWN**, cioè verso GND, come nell'esempio che segue, in questo caso se il pulsante **NON E'** premuto la variabile pulsante **vale 0**, se invece **E'** premuto la variabile pulsante **vale 1**.



Anche in questo caso la lettura dello stato del pulsante viene effettuata con la funzione **digitalRead.**

```
int pulsante;
pulsante= digitalRead(5);
```



PIEDINI 0 ed 1

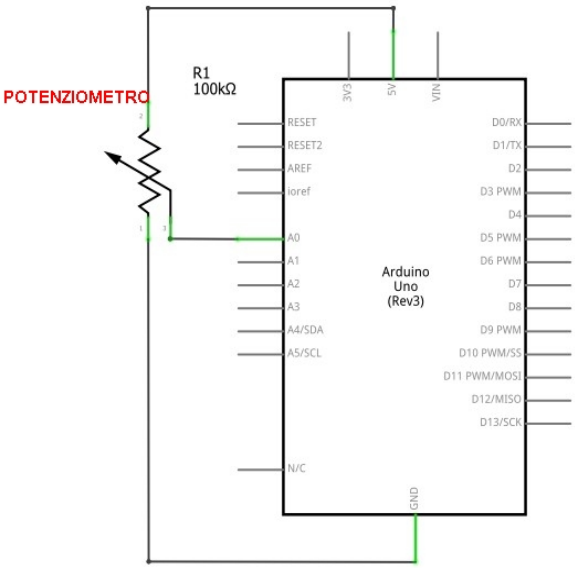
Come si può vedere nell'immagine questi due pin hanno una duplice funzione, sono utilizzati infatti come piedini digitali, o come segnali RX e TX per la trasmissione seriale tra la scheda ed il PC, tramite il connettore USB ed il relativo convertitore presente a bordo della scheda. Questi piedini vengono anche utilizzati in fase di programmazione, pertanto è preferibile non utilizzarli se non necessario.

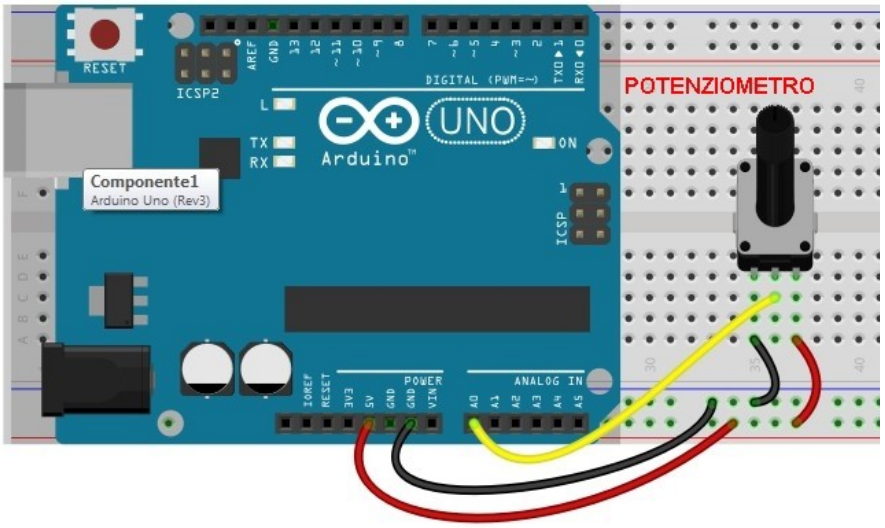
PIN ANALOGICI

Sulla scheda ARDUINO UNO ci sono inoltre 6 ingressi analogici denominati; A0,A1,A2,A3,A4,A5.

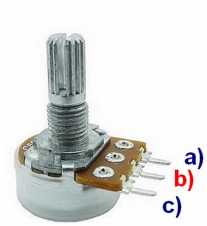
Un ingresso analogico a differenza di quello digitale non può assumere solo due valori 0 e 1, ma **può assumere infiniti valori all'interno di un range predefinito.** Nel nostro caso il range possibile va da 0 a 5 Volt, ma può essere anche differente applicando una tensione sul piedino AREF presente sull'altro connettore.
La tensione applicata sul pin AREF dovrà ovviamente essere sempre inferiore a 5Volt.

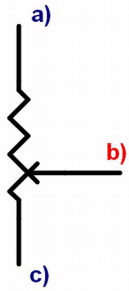
Un piedino analogico viene utilizzato per leggere un valore di tensione, come ad esempio quello che potremmo avere in uscita da un sensore analogico utizzato per misurare una qualsiasi grandezza fisica.
Potremmo ad esempio collegare un sensore di temperatura, un sensore di luminosità, un potenziometro cioè una resistenza variabile, o di altro genere.
Per comprendere il funzionamento di quesi ingressi e le relative funzioni software, supponiamo di collegare a questo piedino il piedino centrale di un potenziometro, come in figura.





In pratica il piedino centrale del potenziometro si può spostare verso uno dei due piedini laterali dello stesso componente, ruotando in senso orario o antiorario. Visto che i due piedini laterali sono collegati a 0Volt ed a 5Volt, il valore della tensione presente sul piedino centrale andrà da 0 a 5Volt in maniera continua.





Con un potenziometro potremmo ad esempio leggere la tensione in ingresso al piedino A0, e scegliere di conseguenza il valore di una variabile interna che poi potrà essere utilizzata in vari modi.
Potremmo ad esempio in base alla posizione del potenziometro, accendere una sequenza di led collegati agli ingressi digitali, o visualizzare un numero variabile su un display.

Analizziamo ora la funzione **analogRead** utilizzata per leggere un ingreso analogico.

Questa funzione restituisce un valore compreso tra 0 e 1023 a seconda del valore di tensione in ingresso.

Se la tensione in ingresso è massima il valore restituito è 1023, nel nostro caso il valore massimo della tensione di ingresso è 5Volt a cui corrisponde il valore 1023.

Questo perché al suo interno utilizza un circuito che converte la tensione analogica in un numero digitale, utilizzando 10 bit, e con 10 bit si possono avere al massimo 2^{10} = 1024 combinazioni da 0 a 1023.

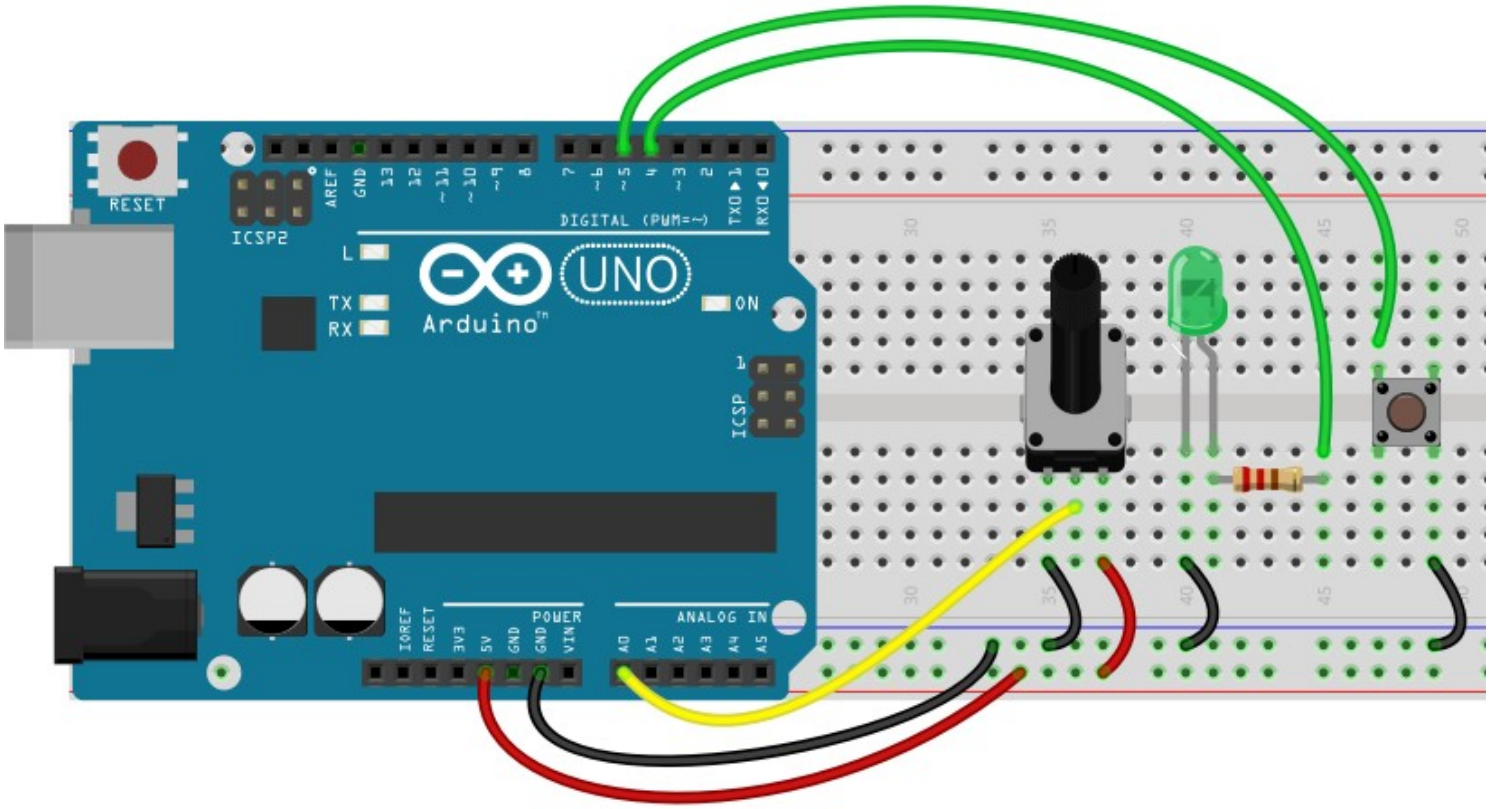
Se vogliamo conoscere il valore della tensione perciò, sarà sufficiente fare una semplice proporzione:

5Volt : 1023 = Vin : Valore letto

Il valore restituito dalla funzione necessita di una variabile intera che va dichiarata prima di chiamare la funzione.

```
int valore;
valore= analogRead(A0);
```

A questo punto possiamo provare a sintetizzare in un unico circuito quanto detto finora, realizzando un programma per far lampeggiare un LED, se un pulsante viene premuto, variando la velocitù del lampeggio mediante un potenziometro.

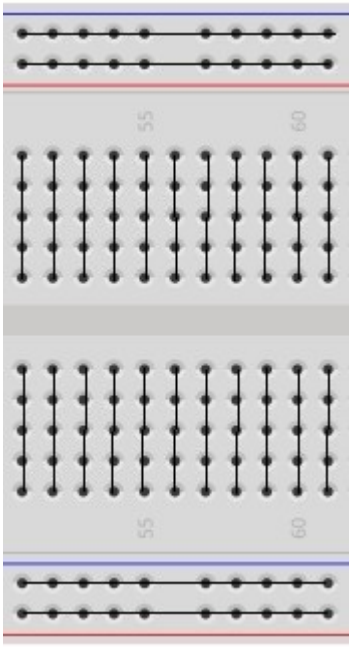


Piedino 4 OUTPUT, uscita per accendere e spegnere il LED

Piedino 5 INPUT, ingresso per leggere lo stato del pulsante

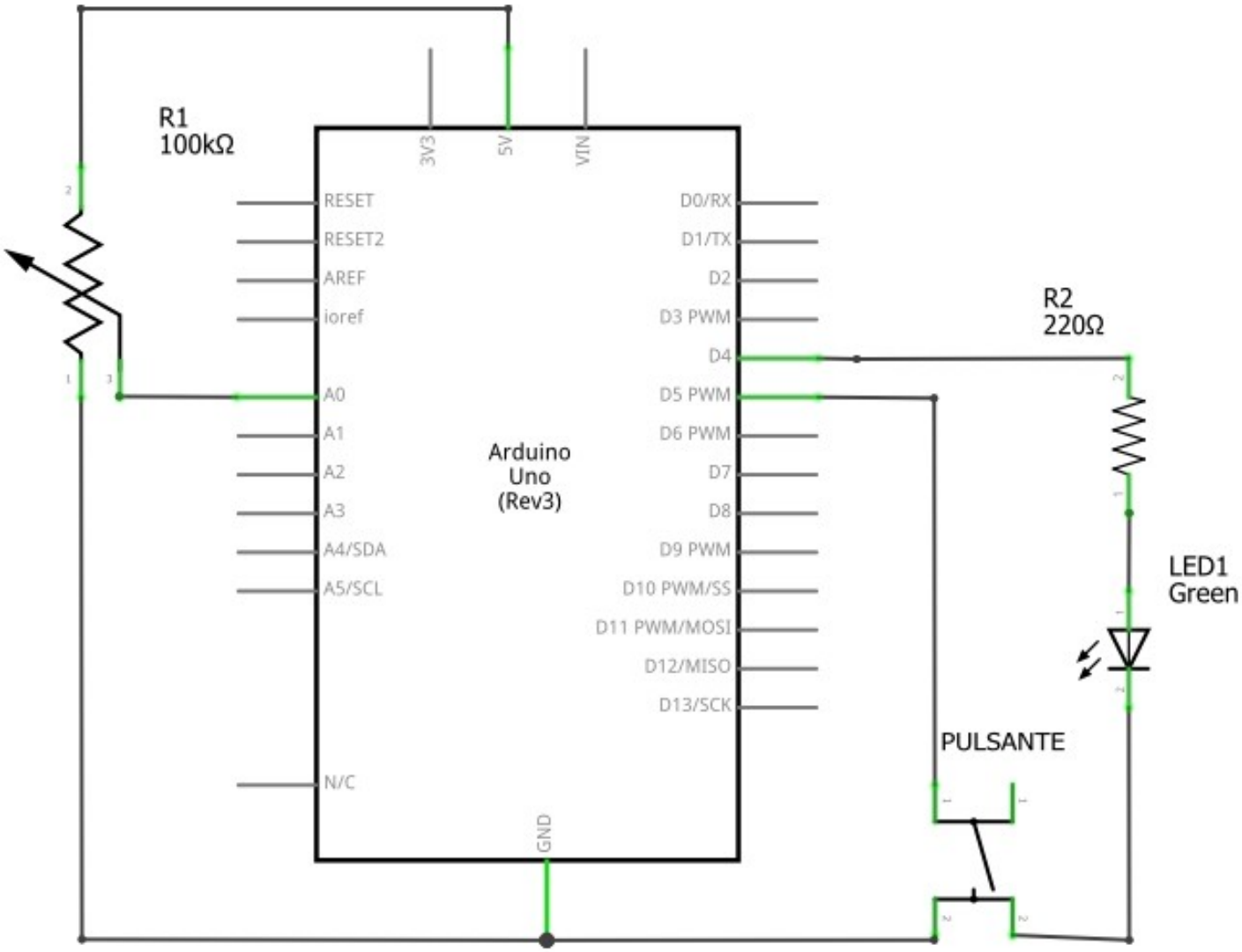
Piedino A0 ingresso analogico per leggere il valore di tensione in base alla rotazione del potenziometro.

Per comprendere il collegamento del precedente circuito come quello dei precedenti, bisogna conoscere la breadboard, e cioè la basetta sperimentale visualizzata negli schemi precedenti.



In una breadboard i tanti fori dove vengono inseriti i piedini dei componenti, sonocollegati come nell'immagine di lato.

Se passiamo dallo schema di montaggio precedente allo schema elettronico otterremo lo schema che segue:



La prima da cosa da fare è realizzare il diagramma di flusso.

Impostazione dei piedini di ARDUINO

Test del pulsante

Accensione del LED

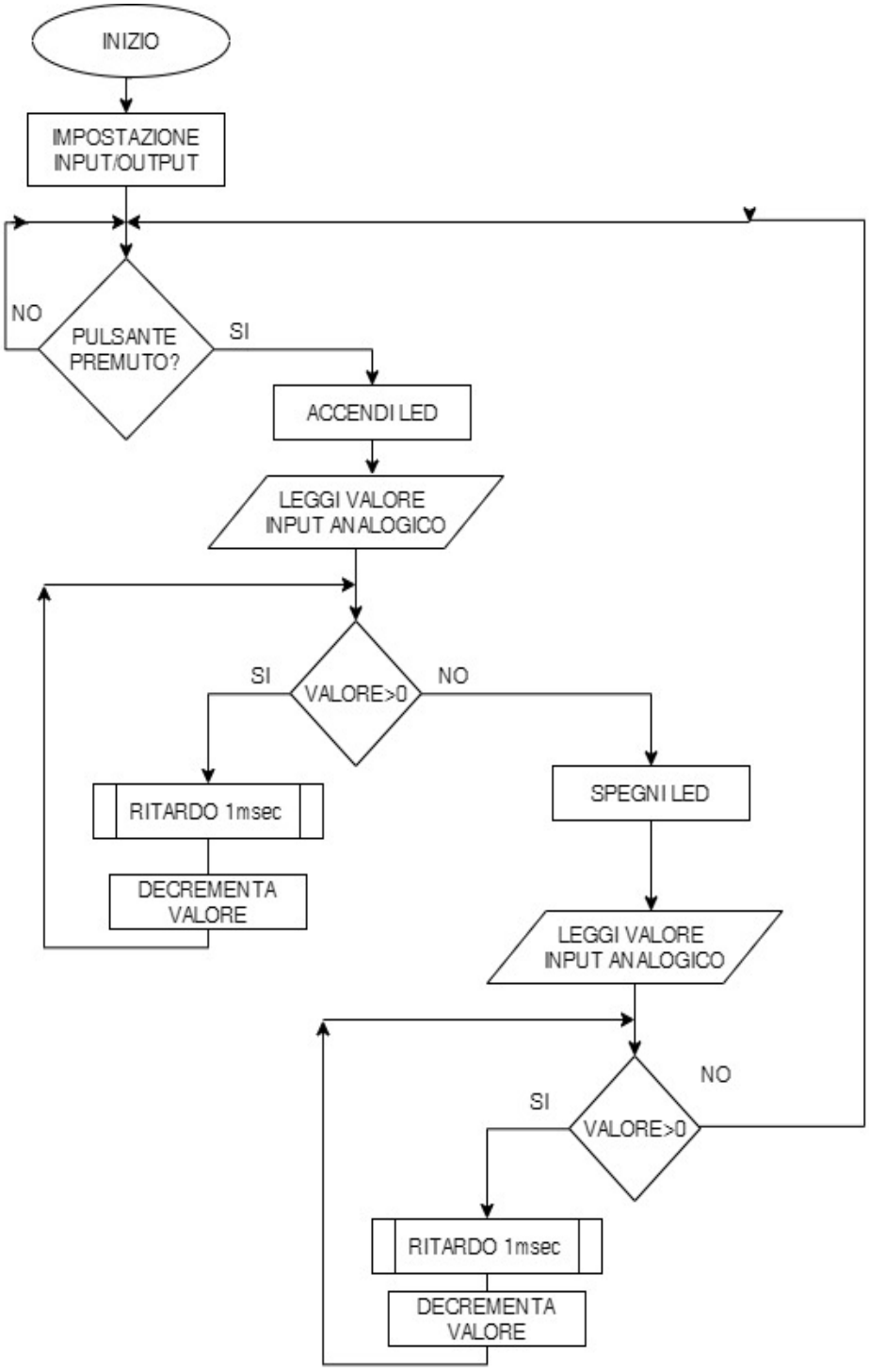
Lettura dell'ingresso analogico

Ritardo di un msec ripetuto un numero di volte pari al valore letto sull'ingresso analogico.

Spegnimento del LED

Lettura dell'ingresso analogico

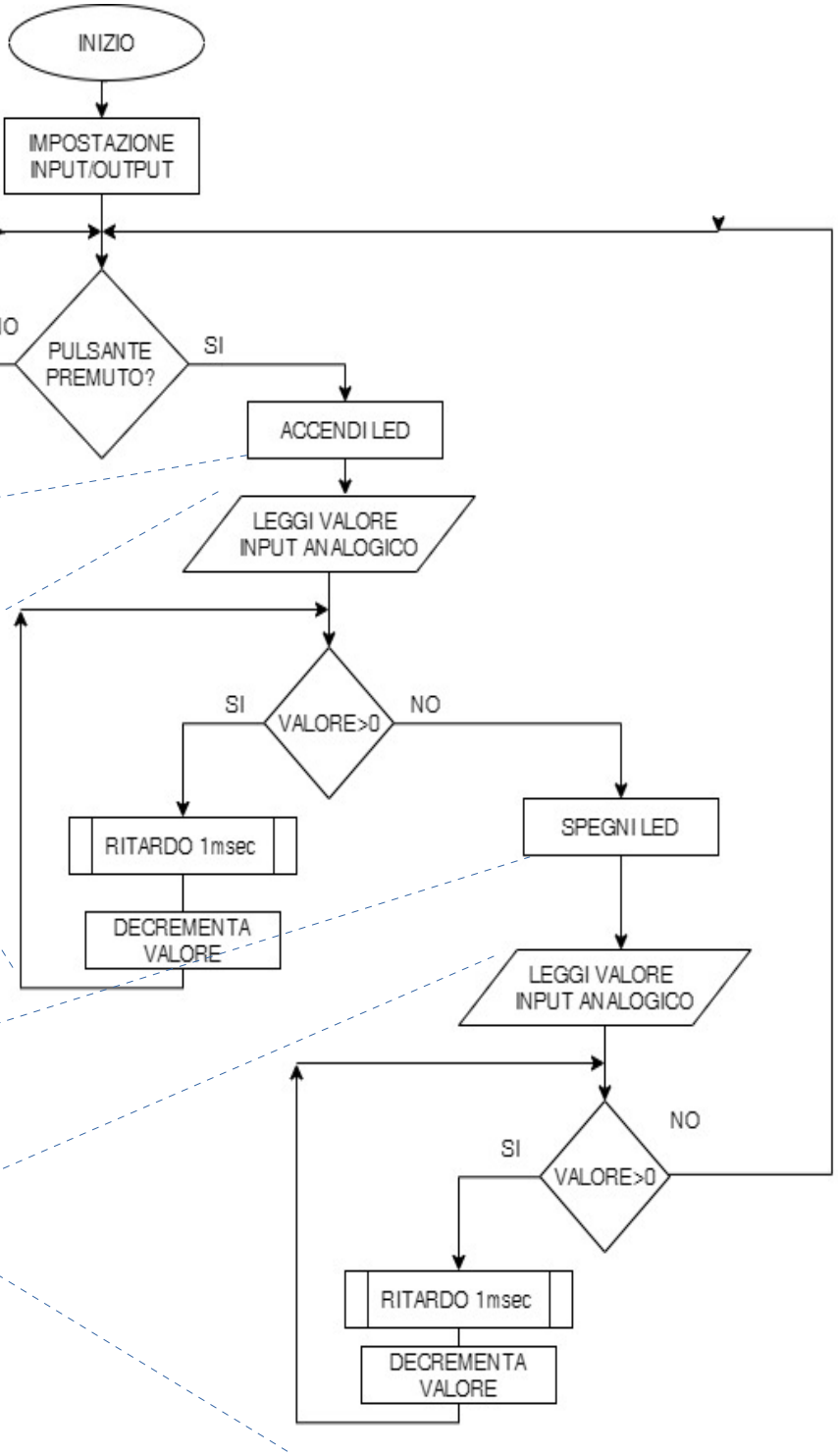
Ritardo di un msec ripetuto un numero di volte pari al valore letto sull'ingresso analogico.



La parte in grassetto si ripete due volte allo stesso modo, perciò vedremo più avanti come si potrà semplificare il programma.

Per ora limitiamoci a tradurre il diagramma di flusso con il linguaggio di programmazione utilizzato nell'ambiente di sviluppo di ARDUINO.

```
int pulsante;
int valore;
void setup() {
   pinMode(4,OUTPUT);
   pinMode(5,INPUT);
}
void loop() {
 do{
     pulsante=digitalRead(5);
 }while(pulsante==0);

digitalWrite(4,HIGH);

valore=analogRead(A0);
while(valore>0){
   delay(1);
   valore=valore-1;
}

digitalWrite(4,LOW);

valore=analogRead(A0);
while(valore>0){
   delay(1);
   valore=valore-1;
}

}
```



```
//dichiarazione di variabili
int pulsante;
int valore;

//impostazioni input/output
void setup() {
   pinMode(4,OUTPUT);
   pinMode(5,INPUT);
}

//loop continuo
void loop() {
 //lettura del pulsante
 do{
     pulsante=digitalRead(5);
 }while(pulsante==0);
 digitalWrite(4,HIGH);  //accensione del LED
 valore=analogRead(A0); //lettura ingresso analogico
 while(valore>0){       //ritardo variabile
    delay(1);
    valore=valore-1;
 }//while
 digitalWrite(4,LOW);  //spegnimento del LED
 valore=analogRead(A0); //lettura ingresso analogico
 while(valore>0){       //ritardo variabile
    delay(1);
    valore=valore-1;
 }//while
}//loop
```

Come detto in precedenza c'è una parte che si ripete due volte allo stesso modo nel programma, ed è quella relativa al ritardo variabile.

Le due parti che si ripetono allo stesso modo, sono quelle nel riquadro rosso, per ottimizzare il programma possiamo perciò dichiarare una funzione che contiene le stesse righe di codice e che viene richiamata nel programma.

La funzione va dichiarata prima del ciclo loop, e la chiameremo RITARDO.

La funzione non restituisce alcun valore, esegue solo le funzioni di lettura valore analogico ed esecuzione del ritardo.

```
//loop continuo
void loop() {
 //lettura del pulsante
 do{
     pulsante=digitalRead(5);
 }while(pulsante==0);
 digitalWrite(4,HIGH);  //accensione del LED
 valore=analogRead(A0); //lettura ingresso analogico
 while(valore>0){       //ritardo variabile
    delay(1);
    valore=valore-1;
 }//while
 digitalWrite(4,LOW);  //spegnimento del LED
 valore=analogRead(A0); //lettura ingresso analogico
 while(valore>0){       //ritardo variabile
    delay(1);
    valore=valore-1;
 }//while
}//loop
```

Perciò il ciclo loop, cambierà e prima di esso verrà dichiarata e definita la funzione ritardo.

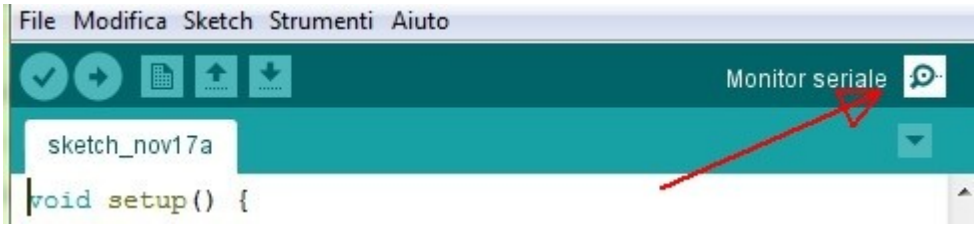
A destra il nuovo programma ottimizzato.

```
//dichiarazione di variabili
int pulsante;
int valore;

//impostazioni input/output
void setup() {
   pinMode(4,OUTPUT);
   pinMode(5,INPUT);
}

void ritardo(){
 valore=analogRead(A0); //lettura ingresso analogico
 while(valore>0){       //ritardo variabile
    delay(1);
    valore=valore-1;
 }//while
}//ritardo

//loop continuo
void loop() {
 //lettura del pulsante
 do{
     pulsante=digitalRead(5);
 }while(pulsante==0);
 digitalWrite(4,HIGH);  //accensione del LED
 ritardo();
 digitalWrite(4,LOW);  //spegnimento del LED
 ritardo();
}//loop
```

MONITOR SERIALE E DEBUG

Uno strumento molto utile dell'IDE di ARDUINO, è il monitor seriale, che è molto utile per effettuare il debug del programma.

Il Monitor seriale è attivabile con l'apposita icona, solo se la scheda è collegata.



Si aprira una finestra che ci consentirà mediante l'istruzione **serialPrint** o **serialWrite**, il valore delle variabili che scegliamo.

Per utilizzare il monitor seriale, occorre inizializzare nel setup, la comunicazione seriale con l'istruzione **serial.begin(9600)**, dove 9600 è la velocità della seriale espressa in bit per secondo (bps).

Di seguito un esempio di come scrivere sul monitor seriale.

```
int valore1=5;

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.print(valore);//non va accapo
  Serial.println(valore);//va accapo
}
```

Con questo programma si conclude la seconda parte del tutorial su ARDUINO.

Le funzioni che questa scheda può effettuare, come anche i dispositivi hardware che si possono pilotare sono molteplici.

Nella guida online del suo ambiente di sviluppo si trovano diverse librerie, ma è anche possibile aggiungere librerie esterne già pronte o costruite da noi.