Il primo sketch

- Il blocco di codice delimitato da /* e */ è considerato un commento, ovvero sarà ignorato dal compilatore
- Tutti gli sketch Arduino devono contenere la definizione della funzione ***void setup() { }*** . È la prima che viene eseguita quando la scheda viene alimentata o resettata
  - ***void*** indica che la funzione non restituisce alcun valore ma vengono solo eseguite le istruzioni al suo interno
  - ***setup*** è il nome della funzione
  - Le parentesi graffe ***{*** e ***}*** racchiudono le istruzioni associate alla funzione

I TIPI DI DATO

- ***boolean***: il tipo di dato più semplice, ha valore true o false
- ***char***: occupa un byte di memoria e contiene un singolo carattere; ogni carattere equivale a un numero compreso tra –128 e 127
- ***byte*** o ***unsigned char***: questo tipo di dato ha le stesse caratteristiche di char, ma l'intervallo di valori che può assumere va da 0 a 255, senza numeri negativi
- ***int***: lo spazio occupato da questo tipo di dato è di 2 byte; equivale ai numeri interi compresi tra –32'768 e 32'767
- ***word*** o ***unsigned int***: in questo caso le caratteristiche sono analoghe a quelle del tipo int; i valori vanno da 0 a 65'535
- ***long***: i byte occupati sono 4; comprendere i numeri interi tra -2'147'483'648 e 2'147'483'647
- ***unsigned long***: valori compresi tra 0 e 4'294'967'295
- ***float***: questo tipo di dato occupa 4 byte; i valori possibili includono i numeri decimali compresi tra 3,4028235E+38 e –3,4028235E+38

La funzione loop

- ***void loop() { }***
- le istruzioni in essa contenute vengono ripetute in ciclo continuo quando la scheda è alimentata
- Utilizzando le costanti si possono ridurre al minimo i valori hardcoded, ovvero inseriti direttamente nel codice
  - Utilizzando const per dichiarare una costante stiamo comunicando al compilatore che il valore di tale parametro non verrà mai modificato durante l'esecuzione del programma
  - ***const int* ledPin = 13;**

Input digitali

I componenti di cui abbiamo bisogno sono:

- un **LED** da collegare al pin 13 (possiamo sfruttare quello già integrato su Arduino Uno);
- un **pulsante** collegato tra il pin 2 e quello contrassegnato con 5V;
  - Questo pulsante è di tipo NO (Normally Open, ovvero normalmente aperto): quando è in posizione di riposo gli impulsi elettrici non possono attraversarlo, mentre premendolo l'elettricità di scorre
  - Il pulsante ha quattro terminali. Posizionandolo a cavallo della fessura centrale si può inserirlo solo in un verso. I due in alto sono sempre collegati tra loro, e lo stesso i due in basso. Premendo il pulsante la corrente è libera di scorrere dall'alto verso il basso e viceversa
- una **resistenza** da 10K Ohm da collegare tra il pin 2 e GND.

Il codice

const int buttonPin = 2;

const int ledPin = 13;

- Definiscono le costanti
  - **buttonPin** il pin a cui è collegato il pulsante
  - **ledPin** il pin a cui è collegato il LED

Il codice

int buttonState = 0;

- variabile **buttonState** il cui valore potrà essere modificato durante l'esecuzione dello sketch
- Il tipo di dato attributio a questa variabile è *int* e il valore iniziale è *0*

Il codice

```
void setup() {

pinMode(ledPin, OUTPUT);

pinMode(buttonPin, INPUT);
}
```

- nella funzione **setup()** l'istruzione **pinMode()** si occupa di impostare i pin interessati come **OUTPUT** o **INPUT**

Il codice

```
void loop(){
    buttonState = digitalRead(buttonPin);
    if (buttonState == HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    else {
        digitalWrite(ledPin, LOW);
    }
}
```

- la funzione loop(), che conterrà le istruzioni da eseguire ciclicamente dopo l’avvio

OPERATORI DI CONFRONTO

- **a == b** (a è uguale a b);
- **a != b** (a è diverso da b);
- **a > b** (a è maggiore di b);
- **a < b** (a è minore di b);
- **a >= b** (a è maggiore o uguale a b);
- **a <= b** (a è minore o uguale a b);

Comunicare da Arduino al computer

- La scheda Arduino è anche in grado di comunicare tramite messaggi seriali durante l’esecuzione dello sketch
- In questo modo può trasmettere o ricevere dati attraverso la connessione USB
- Il livello di interazione può diventare anche piuttosto complesso

la comunicazione seriale

l'istruzione ***Serial.begin()***

- da utilizzare nella funzione **setup()**
  - Abilita la comunicazione
  - Come parametro prevede la velocità in bit al secondo della connessione con il computer
  - Utilizzeremo un valore standard di 9600
- Il convertitore seriale-USB è connesso ai pin digitali 0 e 1 della scheda
  - Non destinare tali pin al collegamento di altri componenti

la comunicazione seriale

l’istruzione ***Serial.println()***

- il messaggio passato alla funzione come parametro tra parentesi verrà trasmesso sulla porta seriale
  - Per trasmettere una stringa di caratteri è necessario delimitarla con le virgolette doppie " all’interno della parentesi
  - Omettendo le virgolette, come in ***Serial.println(anno)***, verrà trasmesso il valore assunto dall’ipotetica variabile

la comunicazione seriale

l’istruzione ***Serial.println()***

- il messaggio passato alla funzione come parametro tra parentesi verrà trasmesso sulla porta seriale
  - Per trasmettere una stringa di caratteri è necessario delimitarla con le virgolette doppie " all’interno della parentesi
  - Omettendo le virgolette, come in ***Serial.println(anno)***, verrà trasmesso il valore assunto dall’ipotetica variabile