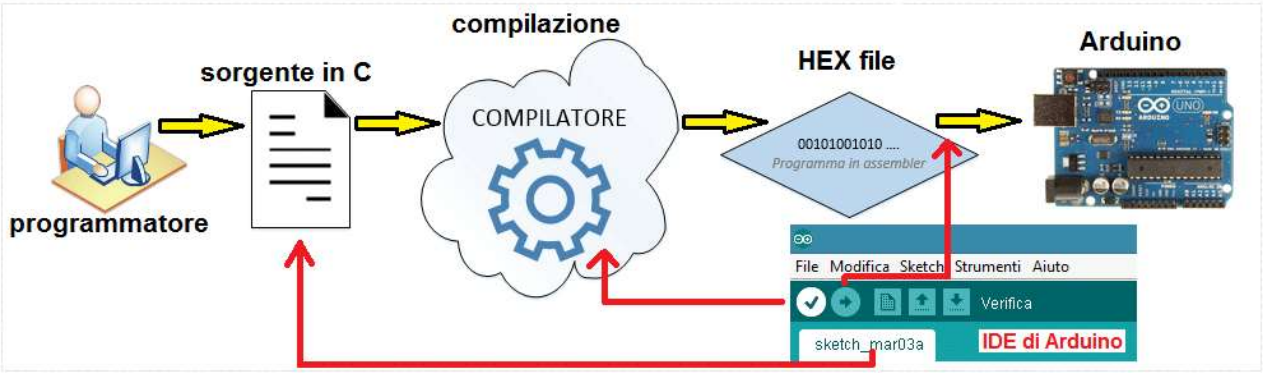
La programmazione della scheda Arduino

Prima di affrontare la programmazione della scheda Arduino è opportuno chiarire che il programma che scriviamo nell'IDE si chiama sorgente, è un testo umanamente comprensibile, ma non per il micro di arduino pertanto è necessario un processo che trasformi il programma sorgente in una sequenza di comandi comprensibile dal micro di Arduino (linguaggio assembler o hex) si dice COMPILAZIONE. Il programma che effettua tale codifica si dice compilatore.



L'ambiente di sviluppo è scaricabile gratuitamente dal sito www.arduino.org, dopo semplici passaggi otterremo una icona sul desktop che consente di mandare in esecuzione l'ambiente di sviluppo per Arduino.

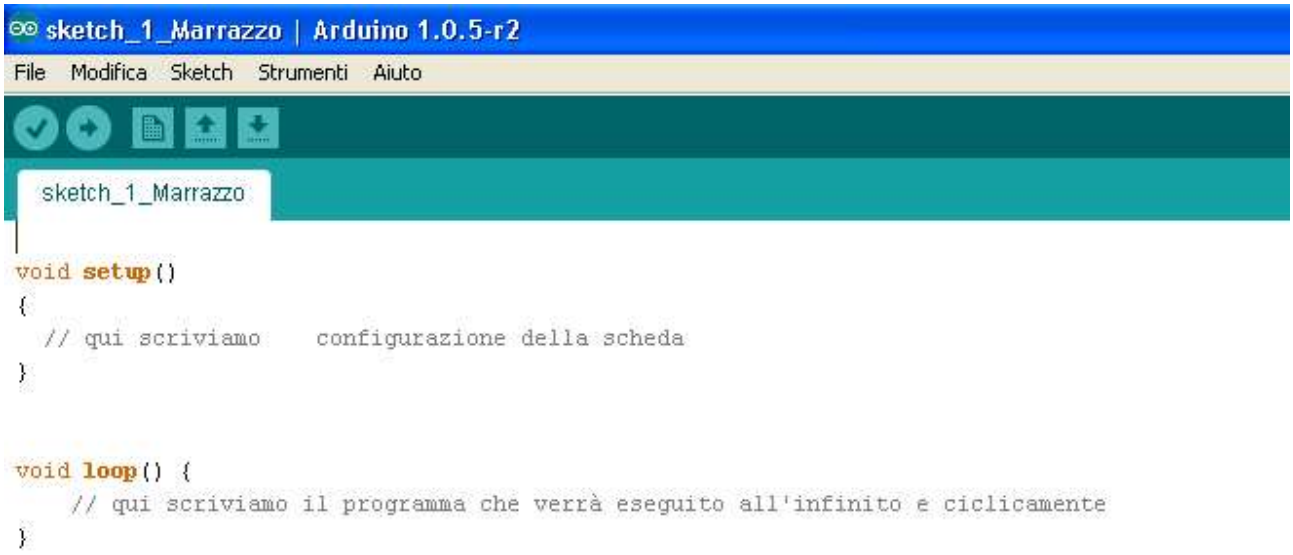
L'ambiente di sviluppo

L'ambiente di sviluppo Arduino consente di scrivere i programmi, chiamati sketch, all'interno di un editor, di compilarli e di copiarli nella memoria della scheda Arduino. II termine derivante dalla lingua inglese, significa bozza o composizione per pianoforte e rappresenta il programma che viene eseguito sulla scheda Arduino. Il codice sorgente è formato da due sezioni:

- setup()
- loop() .

setup( ) è la funzione che viene eseguita solo all'apertura dello sketch, circa 5 secondi dopo l'accensione o il reset della scheda. In questo blocco possiamo effettuare le inizializzazioni necessarie al programma, come ad esempio 1'impostazione dei valori iniziali delle variabili, le modalità di uso dei pin (input, output ecc.) e inizializzazione delle librerie utilizzate
loop ( ) è il programma principale, le istruzioni contenute al suo interno vengono eseguite ciclicamente fino allo spegnimento della scheda. Quando lo colleghiamo a una fonte di alimentazione, come ad esempio un cavo USB connesso al computer oppure una batteria, viene avviato il programma caricato.

Notiamo che entrambe sono caratterizzate dalla presenza {} **Curly braces** note anche semplicemente come "parentesi" o "parentesi graffe" definiscono l'inizio e la fine dei blocchi di codice e dei blocchi di istruzioni.



Alcune delle operazioni più comuni sono accessibili attraverso una serie di pulsanti posti immediatamente al di sotto alla barra dei menu e che rispettivamente indicano:

- Verifica : esegue la verifica del codice scritto e relativa compilazione (primo da sin.);
- Carica : esegue il caricamento sulla board del firmware compilato(secondo da sin.);
- Nuovo : permette di creare un nuovo sketch (terzo da sin.);
- Apri : permette di aprire uno sketch esistente(quarto da sin.);
- Salva : permette di salvare lo sketch correntemente aperto Quinto da sin.);

Pulsati per accesso diretto alle principali funzioni

Il Linguaggio di programmazione

Come precedentemente accennato il linguaggio usato per la creazione degli sketch deriva dal linguaggio C dal quale linguaggio eredita gran parte della sintassi delle istruzioni. Come abbiamo visto uno sketch si compone di due sezioni principali, una chiamata setup ( ) e una seconda chiamata loop ( ). Possono tuttavia essere create anche altre funzioni personalizzate, scrivendole alla fine del blocco loop ( ). Per capire meglio l'uso di queste due funzioni analizziamo il semplice sketch dell'esempio che segue:

```
int pulsante = 3;
int led = 8;
void setup()
{
    pinMode(pulsante, INPUT);
    pinMode(led, OUTPUT);
}
void loop()
{
      if (digitalRead(pulsante) == HIGH)
      digitalWrite(led, HIGH);
      if (!digitalRead(pulsante) == HIGH)
      digitalWrite(led, LOW);

}
```

Nelle prime due righe vengono definite due variabili di tipo intero (pulsante e led) che contengono l'indirizzo del pin digitale della scheda relativi a un pulsante collegato al piedino 3 e a un LED collegato col piedino 8, fatto questo è indispensabile definire se sono di INPUT o OUTPUT, infatti successivamente nella funzione di inizializzazione (setup) vengono configurati i due piedini, uno in input e l'altro in output, in quanto il LED rappresenta il nostro attuatore (OUTPUT) mentre il pulsante il nostro sensore (input), a questo punto la configurazione della scheda è conclusa. Nella sezione (programma principale) loop viene verificato il valore letto dal piedino collegato al pulsante, se il pulsante è premuto viene acceso il LED altrimenti viene spento. Come possiamo notare la lettura dal piedino 3 avviene mediante la funzione digitalRead (n°piedino), mentre l'output verso il piedino a cui è collegato un LED avviene mediante la funzione digitalWrite (n°piedino).
Riassumendo, oltre a tutte le istruzioni del linguaggio C, rappresentate dai blocchi (parentesi graffe), selezione (if), cicli (for, while, do) e tipi di variabili, abbiamo le seguenti funzioni riservate tipiche di questo linguaggio necessarie per la gestione dell'I/O. Di seguito sono elencate le funzione suddivise in analogico e digitale

Ingressi / uscite digitali

pinMode(pin, mode)
Utilizzato nella funzione setup ( ), serve per configurare un determinato pin e stabilire se deve essere un ingresso o uscita.

digitalRead(pin)

L'istruzione permette di leggere lo stato di un pin di input e restituisce un valore HIGH se al pin è applicato una tensione o un valore LOW se non è applicato nessun segnale. Il pin può essere specificato come una variabile o una costante.

digitalWrite(pin, valore)
Attiva o disattiva un pin digitale, quindi l'istruzione pone il pin di uscita a livello logico HIGH o LOW. Il pin può essere specificato come una variabile o una costante.

Ingressi / uscite analogici

analogRead(pin)
Legge il valore di tensione applicato a un pin di input analogico con una risoluzione pari a 10 bit. Questa funzione restituisce un numero intero compreso tra O e 1023.

Serial.prìntln()
Invia su seriale il valore dell'argomento con il carattere di invio a capo alla fine (\ n). L'argomento può essere una variabile oppure una costante (numerica o stringa, quest'ultima va racchiusa tra doppi apici). Ad esempio per stampare il contenuto della variabile temperatura:

```
Serial.println(temperatura);
```

Serial.print()
Come println () ma non va a capo dopo aver inviato il dato.

Serial.read()
Restituisce il valore inviato mediante serial monitor. (l'istruzione inArrivo=Serial.read( ) legge il byte in arrivo dal PC tramite porta seriale)

Serial.available()
Indica se sono arrivati dei dati sulla porta seriale, e in caso affermativo restituisce il numero di byte arrivati e disponibili per la lettura.

flush()
Cancella la coda dei dati arrivati sulla porta seriale.

Prima di concludere è opportuno ricorda che nello sketch è possibile o conveniente scrivere un commento al programma:

/ * ... * / Biock comments - Blocco commenti
All'inizio di ogni programma è conveniente scrivere un commento al programma stesso. Una serie di righe di commenti costituiscono un blocco. Il blocco del commento è un'area di testo ignorato dal programma e viene utilizzato per aiutare a capire le parti del programma.
Iniziano con / * e finiscono con * / e possono estendersi su più righe.

```
/ *   Questo è un blocco di apertura del commento.
      Non dimenticare il blocco di commento di chiusura.
      Essi devono essere sempre bilanciati!
*/
```

Poiché i commenti sono ignorati dal processore di Arduino non occupano spazio di memoria, quindi possono essere usati con generosità e possono anche essere usati per "commentare" blocchi di codice per il debug dello skecth.(Mentre è possibile completare ogni singola riga del programma

con un blocco di commento, non è permesso inserire un secondo blocco di commenti al suo interno).

// Line comments - Singola linea di commento

I commenti d: singola linea iniziano con // e terminano con la prossima linea di codice. Come i commenti a blocchi, essi sono ignorati dal programma e non occupano spazio di memoria.

// Questo è un commento singola linea

I commenti di singola linea sono spesso utilizzati dopo una dichiarazione valida per fornire maggiori informazioni su ciò che la dichiarazione compie o per fornire un promemoria.