

ARDUINO: STARTER PACK

Prof. Nicholas Fattori

Come programmare con Arduino IDE

Nelle compilazione di un programma troviamo:

- variabili, ossia zone di memoria per depositare dati
- istruzioni, ossia comandi che possiamo suddividere in

istruzioni di struttura

setup() → inizializzazione del programma

loop() → zona del corpo del programma

istruzioni di controllo

if... else... → permette di prendere decisioni se la condizione indicata tra () viene soddisfatta o meno, l'if è eseguibile anche senza else

for → ripete una serie di istruzioni fino a quando una condizione risulta vera

switch... case... break.... default → istruzione composta, che lancia parti di programma a seconda di cosa si trova dopo switch come variabile

while → serie di istruzioni ripetute fino ad una condizione di verità

do... while → identica alla precedente, ma si usa quando si vuole eseguire il codice almeno una volta prima che la condizione sia valutata

operatori matematici (+, -, *, /, =) booleani (&& "and", || "or" e ! "not")

operatori computazionali come ++ e -- (per incrementare o decrementare una variabile)

- funzioni suddivise in

funzioni di I/O

funzioni di comunicazione

funzioni di tempo

funzioni matematiche

TIPOLOGIE DI VARIABILI (più in uso)

- byte: occupa un byte di memoria (8bit) e può essere un numero tra 0 e 255
- int: 2 byte di memoria, può essere un numero tra -32768 e 32767
- long: usa 4 byte, può essere un numero compreso tra -2.147.483.648 e 2.147.483.647
- float: usa 4 bytes, serve per numeri con virgola
- double: numeri a doppia precisione in virgola mobile
- char: un byte, se per numero da -128 a +127, se come testo un qualunque carattere ASCII

Abbiamo poi l'array, che si tratta di un elenco di variabili accessibili tramite un indice. E' di tipo int o char seguito da [] con all'interno un valore numerico (a seconda di quanti elementi).

STRUTTURA CODICE

```
/*prima parte  
dichiarazione librerie e variabili  
*/
```

```
void setup() /*seconda parte o parte di setup, eseguita solo all'avvio del programma  
*/  
{
```

```
} /* fine del setup */
```

```
void loop() /*terza parte o parte di loop, parte principale del programma, eseguita  
ripetutamente fino al termine dell'alimentazione o alla pressione del pulsante reset */  
  
{
```

```
} /* fine del loop */
```

REGOLE

- Le variabili e le costanti devono essere dichiarate prima
- Ogni istruzione termine con un “ ; ”
- Ad ogni parentesi aperta deve corrispondere una parentesi chiusa
- Le parentesi tonde e quadre delimitano gli operatori di un'istruzione mentre le graffe delimitano una serie di istruzioni riferibili a una condizione o una parte di programma
- La combinazione // indica l'inizio di una zona di note fino alla fine della riga

PAROLE CHIAVE

HIGH e LOW per gestione porte di Arduino
LOW implica che alla variabile o alla porta sia associato il valore 0 mentre HIGH implica il valore associato 1

INPUT e OUTPUT

definiscono se una porta deve essere considerata di entrata o di uscita

TRUE e FALSE

servono a verificare l'esito di una condizione

IL CONCETTO DI LIBRERIE

Esistono moltissime funzioni gestite da librerie, anche scritte da utenti, che si possono utilizzare per particolari attività. La libreria va dichiarata e inclusa nella parte

iniziale del programma (prima del setup).

Per inserire una libreria `#include <nome_della_libreria.h>`

Per inserirne una nuova nell'archivio, bisogna seguire questo percorso nell'IDE:
Sketch->importa libreria->add library selezionando la cartella o file zip e premendo
apri

GESTIRE LE PORTE DI ARDUINO

`pinMode(porta, utilizzo)`

istruzione per configurare una delle 14 porte digitali

prima si mette il numero di porta da 0 a 13 ed a seguire dopo la , il tipo di utilizzo
(INPUT o OUTPUT)

`digitalWrite(porta, valore)`

attiva (HIGH) o disattiva (LOW) una porta digitale di OUTPUT

`analogWrite(porta, valore)`

è possibile utilizzare una porta digitale come una porta analogica di OUTPUT

COMUNICAZIONE SERIALE

Con `Serial.begin(Velocità)` in fase di `setup()` per comunicare con monitor seriale in IDE o altri device.

`Serial.print(valore)` —> invia al monitor seriale il contenuto della variabile e mantiene lì il cursore

`Serial.println(valore)` —> invia al monitor seriale il contenuto della variabile con segnale di ritorno

`val= Serial.available()` —> inserisce in val la lunghezza del testo digitato su tastiera

`val= Serial.read()` —> legge quello che viene battuto in tastiera collegata ad Arduino

FUNZIONI DI TEMPO

`millis()` —> fornisce i millisecondi trascorsi dall'inizio del programma

`delay(pausa)` —> mette in pausa il programma per tot millisecondi