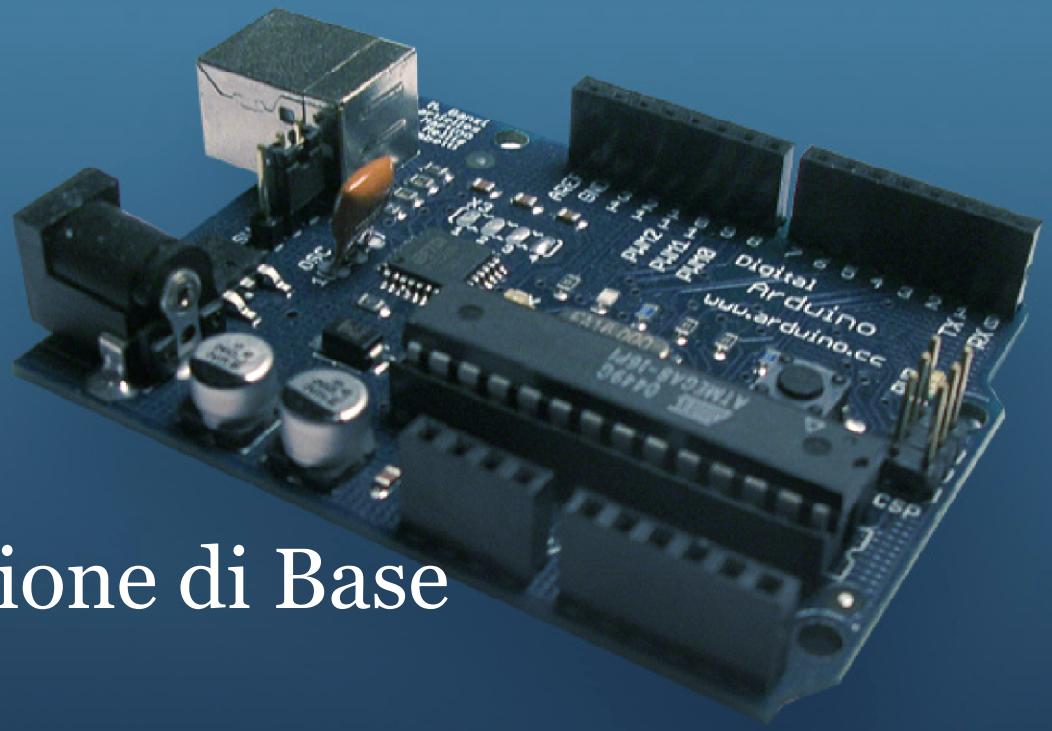
Arduino

Physical Computing I/O board



Programmazione di Base

Variabili

└ Array

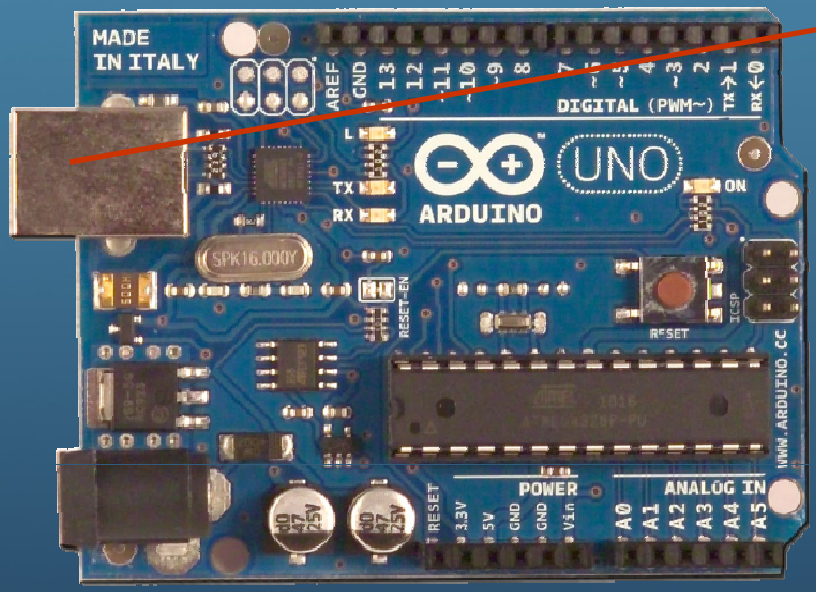
Funzioni

Strutture di controllo

Author: Ing. Sebastiano Giannitto (ITIS “M.BARTOLO” –PACHINO)

3^ parte

Com’è fatto Arduino



Programmazione da USB

Nota:

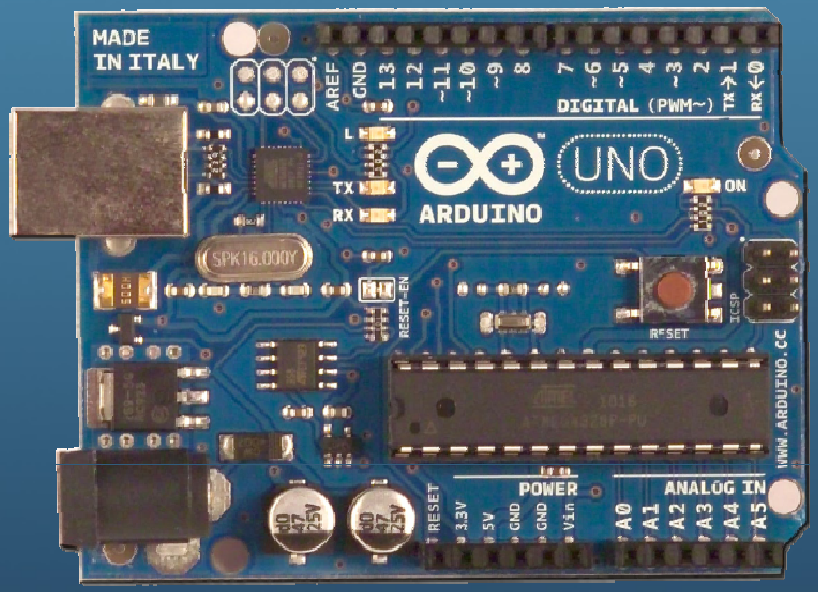
Firmware: Insieme di istruzioni che controllano il funzionamento di un microcontrollore. E’ il «programmino» che andremo a caricare sul nostro Arduino.

Software: Insieme di istruzioni che controllano parte del nostro computer. Sono i programmi che girano sul nostro calcolatore (Word, Excel…).

Incluso il software Arduino che avete installato, chiamato anche «Integrated Development Environment (IDE) Arduino».

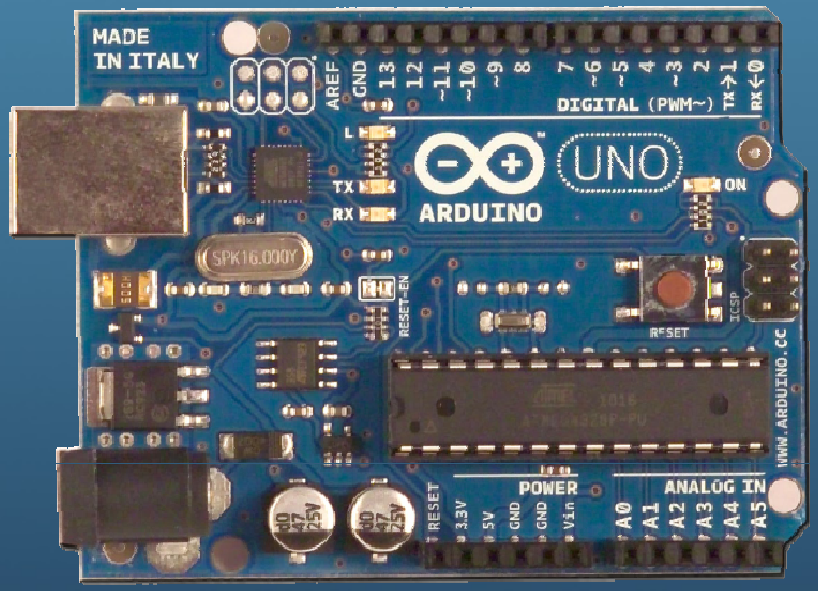
Procesing è un altro IDE che serve tuttavia a creare software per PC/MAC.

Com’è fatto Arduino



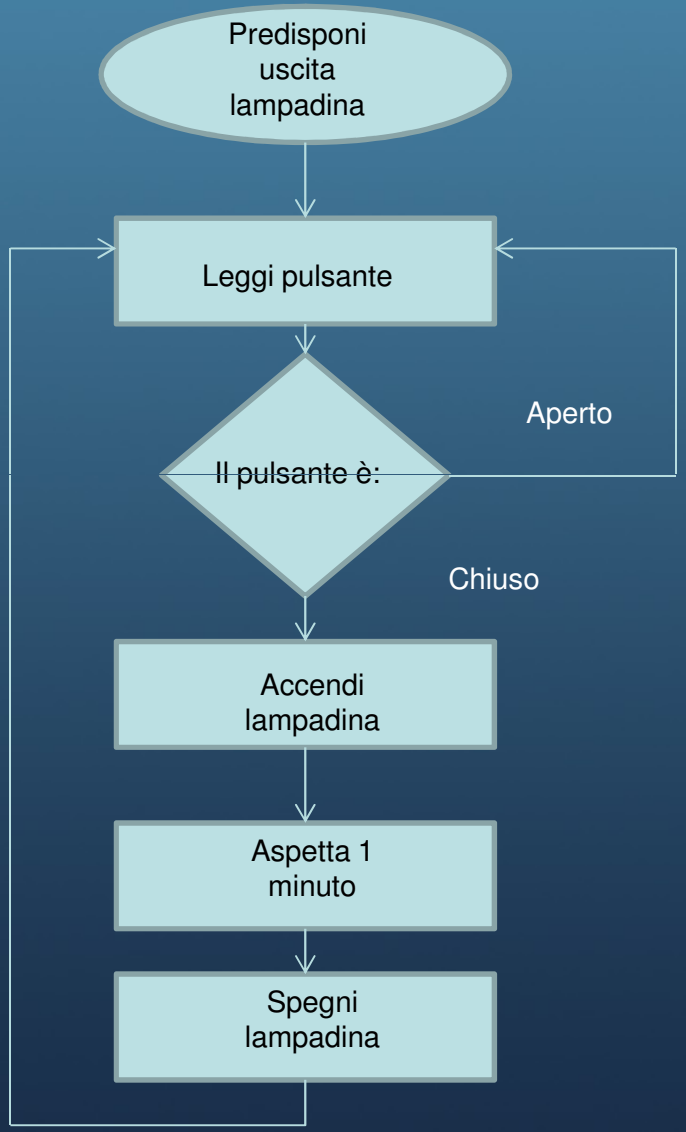
I programmi di Arduino si chiamano sketch o bozzetti.
Collegando gli opportuni sensori , Arduino disporrà di tatto, udito, olfatto e vista ai quali si potrà aggiungere l’orientamento, il calcolo delle distanze , la vista agli infrarossi e altro ancora (giroscopi e accelerometri).
Se un progetto richiede molti collegamenti che superano le porte di I/O disponibili in una sola scheda, allora si potrà intraprendere la strada a più schede Arduino UNO che colloquiano tra di loro e si dividono i compiti.
Arduino dispone di soli 32KB e in questo spazio dovrà starci tutto il nostro programma

Com’è fatto Arduino



- Descrivere il flusso logico
- Convertirlo in una serie di istruzioni
- Compilare il programma
- Trasferirlo sul microcontrollore.

…ma cosa succede se premo il pulsante quando la lampadina è già accesa ?



Esempio di programma

```
/* http://www.arduino.cc/en/Tutorial/Button  */
const  int  buttonPin = 2;     // the number of the pushbutton pin
const  int  ledPin =  13;      // the number of the LED pin

// variables will change:
int  buttonState = 0;         // variable for reading the pushbutton status

void setup()
{
  pinMode(ledPin, OUTPUT); // initialize the LED pin as an output
  pinMode(buttonPin, INPUT); // initialize the pushbutton pin as an input
}

void loop()
{
    buttonState = digitalRead(buttonPin); // read the state of the pushbutton value
  // check if the pushbutton is pressed. if it is, the buttonState is HIGH:
  if (buttonState == HIGH)   { digitalWrite(ledPin, HIGH);  // turn LED on:  }
              else                    { digitalWrite(ledPin, LOW);    // turn LED off:  }
}
```

Esempio di programma

Il programma inizia con la **dichiarazione delle variabili**.

Seguono due blocchi distinti:
-**Void setup();** contenente le istruzioni di inizializzazione che impostano la scheda settando I pin di input e di output
-**Void loop();** contenete il programma principale , che si ripete ciclicamente

/* e */ delimitano I commenti che possono occupare più righe
// segna l'inizio di un commento che però deve essre contenuto in una sola riga
Ogni istruzione deve terminare con il **;**

Le funzioni più usate

pinMode (3,OUTPUT) ; configura il pin 3come OUTPUT
digitalWrite(3,LOW); porta il pin3 al livello LOW (HIGH)
Val=digitalRead(3); legge il pin 3 e assegna alla variabile val il valore LOW o HIGH
analogWrite(3,127); genera un'onda quadra con dc del 50 %
Val= analogRead(3); legge il valore (0÷1023) sul pin 3 (A3)e lo assegna a val
Delay(ms); genera un ritardi di ms millisecondi
delayMicrosecond(µs); genera un ritardo espresso in µs in microsecondi
Val=millis(); pone in val il numero di millisecondi trascorsi dall'inizio dello sketch.
Val=pulseIn(3,HIGH); pone nella variabile val la durata in µ della parte alta (HIGH) dell'impulso che si presenta sul pin di ingresso 3
Tone(3,frq,duration); genera sul pin 3 un tono di frequenza in Hz e durata in ms specificate
noTone(3); interrompe la generazione del tono sul pin 3
Serial.begin(speed); imposta la velocità di trasmissione della porta seriale
Serial.println(data); trasmetti dati tramite laporta seriale
Val= Serial.available(); indica il numero di byte in attesa di essere letti nel buffer della porta seriale
Val=Serial.read(); legge il primo byte del buffer della porta seriale

Le funzioni più usate

shiftOut(dataPin,clockPin,bitOrder,value);

Viene usata per espandere le uscite della scheda tramite uno shift register SIPO (serial input parallel output)collegato con l'ingresso dati seriale e con il clock a 2 dei pin digitali (dataPin e clockPin). bitOrder (LSBFIRST o MSBFIRST) speciica l'ordine con cui escono i bit e con value il byte da inviare-

COSTANTI

Oltre alle costanti numeriche **const int buttonPin = 2;**
alle stringhe o caratteri chiusi tra apici , Arduino usa le costanti HIGH/LOW, INPUT/OUTPUT, true/false.
Const char x[5]="Ciao"

Programmazione di base
Le Variabili

Dichiarazione delle varibili:

```
int ledPin = 13;
int ledPin1 = 10;
float interval = 100.5;
char carattere= 'c';
```

Tipo variabile NOME VARIABILE (= *Valore iniziale*);

Tipi di variabili presenti:

boolean: variabile binaria, può essere **TRUE** o FALSE;

char: usa un byte di memoria e può contenere un qualsiasi carattere. Ad esso è assegnato un valore tra -127 e 128;

byte: contiene 8 bit, valore tra 0 e 255;

int: numero intero tra -32768 e 32767 (2 byte);

word: contiene 16 bit, valore tra 0 e 65535;

long: numero intero tra -2147483,648 e 2147483,647 (4 byte);

float e **double**: numero con virgola mobile tra 3,4028235E+38 and as low as -3,4028235E+38

string: colonna di char;

array: colonna di variabili (int, float…)

Programmazione di base
Le Variabili

```
boolean running = false;
char myChar = 'A';
byte b = B00010010;
int ledPin = 13;
word w = 10000;
long speedOfLight = 186000L;
float sensorCalbrate = 1.117; (variabile con decimali)
```

STRING:

```
char Str2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
char* myStrings[]={"This is string 1", "This is string
2", "string 3"};
```

ARRAY:

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

Programmazione di base
Le Variabili

```
boolean primo;
int arrivo;
int record=130;
…

Primo = record > arrivo

/* se il numero contenuto in arrivo è < 130  la variabile
logica primo è vera. */
```

Esempi di variabili: array

Si può immaginare un array come una sorta di casellario, le cui caselle sono dette **celle** dell'array stesso. Ciascuna delle celle si comporta come una variabile tradizionale; tutte le celle sono variabili di uno stesso tipo preesistente, detto **tipo base** dell'array. Si parlerà perciò di tipi come "array di interi", "array di stringhe", "array di caratteri" e così via.

La dimensione dell'array (ovvero il numero celle di cui esso è composto) viene considerato parte della definizione del tipo array.

Ciascuna delle celle dell'array è identificata da un valore di **indice**. L'indice è generalmente numerico e i valori che gli indici possono assumere sono numeri interi che partono da 0. Si potrà quindi parlare della cella di indice 0, di indice 1, e, in generale, di indice N, dove N è un intero compreso fra 0 e la dimensione dell'array.

Dichiarazione di un array di 10 interi:

```
int myArray[10];
```

Uso dell'array:

```
myArray[0] = 10;
myArray[1] = 5;
...
```

Cosa sono le funzioni

Le funzioni sono parti di codice a cui viene associato un nome.

Le funzioni hanno principalmente tre funzioni:

- Permettere una maggiore leggibilità del codice
- Evitare di riscrivere lo stesso codice
- Possibilità di costruire librerie da poter utilizzare in differenti progetti

Una funzione ha solitamente questa struttura:

```
Tipo_dato_uscita NomeFunzione(tipi_ingressi NomeVariabile…)
{
      Corpo della funzione
}
```

Arduino permette sia l'utilizzo di funzioni da librerie, sia la creazione di nuove funzioni.

Strutture di controllo

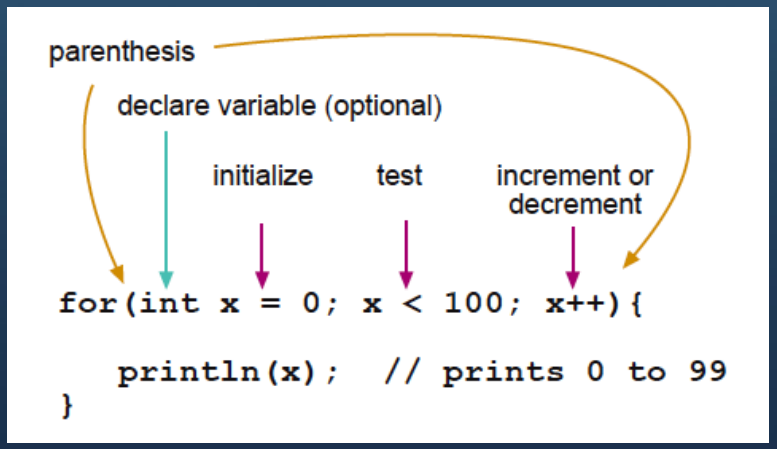
- `if…else`

Se la variabile corrisponde alla prima condizione eseguo azione 1, altrimenti eseguo l'azione due.

```
if (condizione 1)
   azione 1;
else
   azione 2;
```

- Ciclo `for`

Serve per eseguire N volte le funzioni tra parentesi



Strutture di controllo

- **Switch… case**

Il controllo switch si comporta come più if in cascata. È utilizzato per quei casi in cui la variabile può assumere più valori che devono essere controllati tutti (es. Che tasto ho premuto?)

```
switch (var) {
    case 1:
      //do something when var equals 1
      break;
    case 2:
      //do something when var equals 2
      break;
    default:
      // if nothing else matches, do the default
      // default is optional
  }
```

Strutture di controllo

- **Ciclo `while`**

Esegue continuamente le funzioni tra parentesi fino a che la condizione rimane invariata.

```
while(var < 200)
{
  // do something repetitive 200 times
  var++;
}
```

- **Ciclo Do... while**

È come il ciclo while, ma in questo caso la condizione è controllata alla fine.

```
do
{
x = readSensors();  // check the sensors
} while (x < 100);
```

Strutture di controllo

- **continue**

Permette di far proseguire un ciclo do, for o while saltando le istruzioni presenti prima della fine della struttura. Si può utilizzare come parte di un test condizionato

```
for(x=0; x<255; x++)
{
if (x>40 && x<120) { continue;} /* salta le istruzioni per valori compresi tra 41 e 119
println(x);
delay(50);
}
```

- **goto**

Permette di trasferire l'esecuzione del programma al punto definito come etichetta.

- **return**

Restituisce un valore ad una funzione precedentemente chiamata.

```
int checkSensor(){
if (analogRead(0) >400  {return 1; }
 else {return 0; }
}
/* la funzione checkSensor confronta il valore del sensore con 400 se
maggiore restituisce 1 altrimenti 0 */
```