

# IMPARA ARDUINO IN UN GIORNO DAL VIVO

Bologna, 5 aprile 2025

# Introduzione alla programmazione Arduino

La programmazione Arduino è uno strumento potente nel mondo dell'elettronica e della robotica.

- Consente agli utenti di creare progetti interattivi utilizzando hardware e software.
- Durante questa lezione, imparerai la sintassi fondamentale, i costrutti chiave e le applicazioni pratiche della programmazione Arduino.
- Aspettati di acquisire conoscenze pratiche che ti consentiranno di avviare i tuoi progetti!

# Nozioni di base sulla sintassi

Comprendere la sintassi è essenziale per scrivere un codice Arduino efficace.

- **#include:** questa direttiva viene utilizzata per includere librerie che forniscono funzionalità aggiuntive

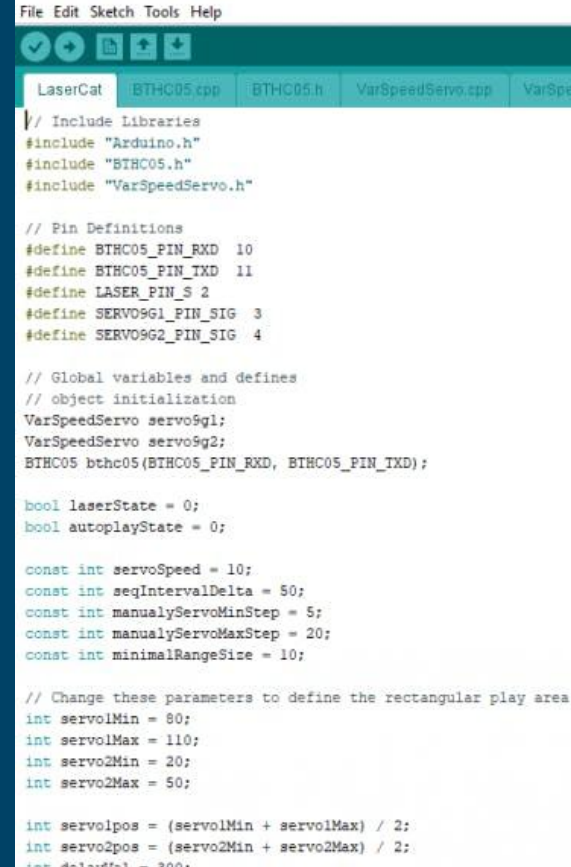
Esempio:

```
#include <Servo.h> //consente di controllare i servomotori.
```

- **#define:** questa direttiva definisce costanti o macro, facilitando la lettura del codice

Esempio:

```
#define LED_PIN 13 // imposta una costante
                  // per il numero di pin del LED.
```



```
File Edit Sketch Tools Help
[Icons]
LaserCat BTHC05.cpp BTHC05.h VarSpeedServo.cpp VarSpeedServo.h
// Include Libraries
#include "Arduino.h"
#include "BTHC05.h"
#include "VarSpeedServo.h"

// Pin Definitions
#define BTHC05_PIN_RXD 10
#define BTHC05_PIN_TXD 11
#define LASER_PIN_S 2
#define SERVO9G1_PIN_SIG 3
#define SERVO9G2_PIN_SIG 4

// Global variables and defines
// object initialization
VarSpeedServo servo9g1;
VarSpeedServo servo9g2;
BTHC05 bthc05(BTHC05_PIN_RXD, BTHC05_PIN_TXD);

bool laserState = 0;
bool autoplayState = 0;

const int servoSpeed = 10;
const int seqIntervalDelta = 50;
const int manuallyServoMinStep = 5;
const int manuallyServoMaxStep = 20;
const int minimalRangeSize = 10;

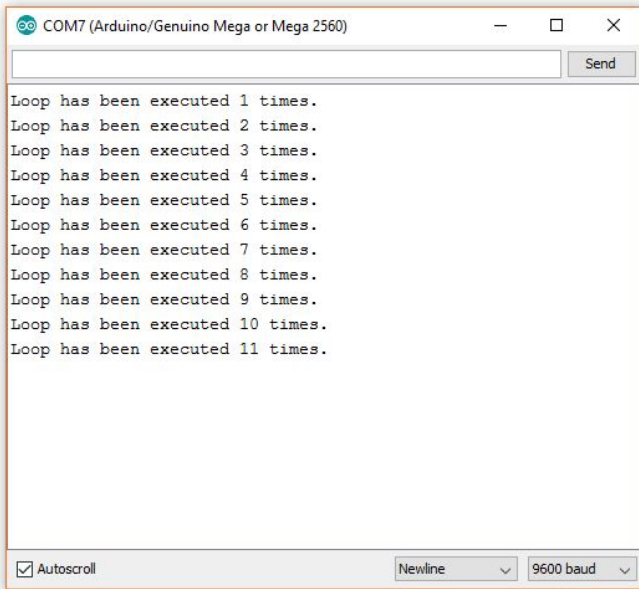
// Change these parameters to define the rectangular play area
int servo1Min = 80;
int servo1Max = 110;
int servo2Min = 20;
int servo2Max = 50;

int servo1pos = (servo1Min + servo1Max) / 2;
int servo2pos = (servo2Min + servo2Max) / 2;
int delayVal = 200;
```

# Cosa sono le variabili e come usarle

Le variabili sono fondamentali per l'archiviazione e la manipolazione dei dati in Arduino.

- **int:** Per numeri interi (numeri interi)  
Esempio: `int count = 10;`
- **float:** per i numeri a virgola mobile (decimali)  
Esempio: `temperatura del galleggiante = 23.5;`
- **char:** per singoli caratteri  
Esempio: `char letter = 'A';`
- **Stringa:** per stringhe di testo  
Esempio: `String message = "Ciao, Arduino!";`

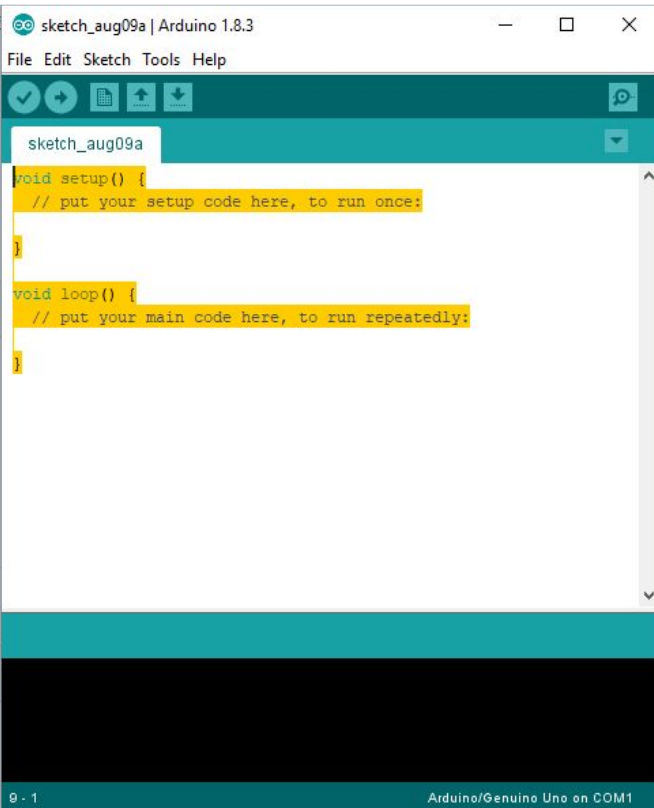


COM7 (Arduino/Genuino Mega or Mega 2560)

```

Loop has been executed 1 times.
Loop has been executed 2 times.
Loop has been executed 3 times.
Loop has been executed 4 times.
Loop has been executed 5 times.
Loop has been executed 6 times.
Loop has been executed 7 times.
Loop has been executed 8 times.
Loop has been executed 9 times.
Loop has been executed 10 times.
Loop has been executed 11 times.
  
```

☒ Autoscroll    Newline    9600 baud



# Le funzioni

Le funzioni sono costrutti che servono per tenere ordinato il codice e per scrivere una sola volta il codice che può essere richiamato ed eseguito più volte.

Due funzioni speciali in Arduino: **setup()** e **loop()**

La struttura di un programma Arduino è definita da due funzioni principali:

- **setup():** Questa funzione viene eseguita una volta all'inizio e viene utilizzata per inizializzare le impostazioni
- **loop():** Questa funzione viene eseguita continuamente dopo l'impostazione, controllando il flusso del programma

# Istruzioni condizionali: if e while

Le istruzioni condizionali consentono il processo decisionale nel codice.

- **if** Istruzione:  
- Esegue un blocco di codice se una condizione è vera

Esempio:

if (temperatura > 30)

- **while** Istruzione:  
- Ripete un blocco di codice mentre una condizione è vera

Esempio:

while (count < 10)

## Conditional Statement

If  $p$ , then  $q$ .  
 $p$  implies  $q$ .  
 $p \rightarrow q$

$p$  - hypothesis or antecedent  
 $q$  - conclusion or consequent



# Iterazione con ciclo For

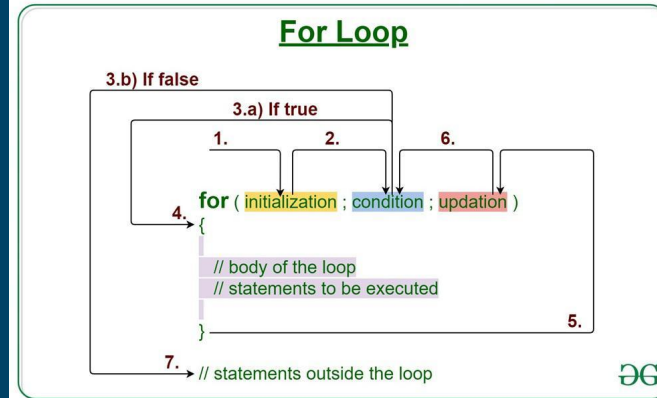
Il ciclo for è un potente strumento per ripetere le azioni:

- **for (inizializzazione; condizione; incremento)**

Esempio:

```
for (int i = 0; i < 5; i++)
```

- Usalo quando sai quante volte vuoi ripetere un'attività.



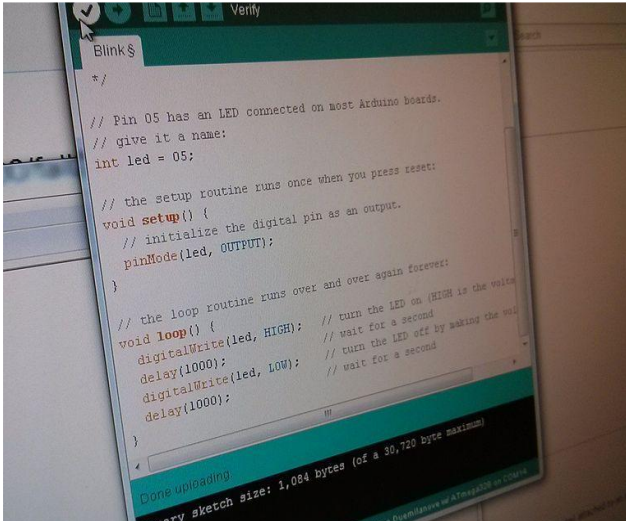
# Funzioni native di Arduino

Arduino offre una varietà di funzioni native per diverse attività.

- **pinMode()**: Imposta se un pin è ingresso o uscita
- **digitalRead()**: Legge il valore da un pin digitale
- **digitalWrite()**: Scrive un valore HIGH o LOW su un pin digitale
- **analogRead()**: Legge il valore da un pin analogico
- **analogWrite()**: Scrive un valore analogico (o PWM) su un pin
- **delay()**: Mette in pausa il programma per un numero specificato di millisecondi.

La lista completa dei comandi predefiniti di Arduino sul sito ufficiale:

<https://docs.arduino.cc/language-reference/>

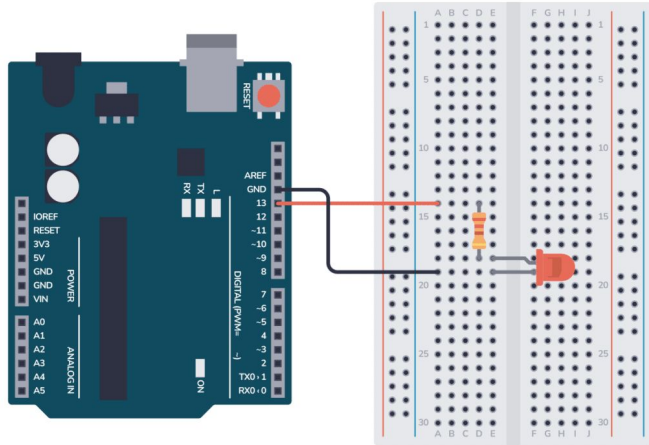




# Iniziare con blink.ino

Introduzione a **blink.ino**: L'esempio di blink.ino è un classico progetto per principianti nella programmazione di Arduino.

- Dimostra le basi del controllo di un LED
- Il programma attiva e disattiva il LED, rafforzando i concetti chiave di programmazione
- Questo semplice progetto è un trampolino di lancio per applicazioni più complesse



# Utilizzo delle librerie di servomotori



L'utilizzo delle librerie di servomotori consente un controllo preciso dei servomotori nei progetti.

Le caratteristiche principali includono l'impostazione dell'angolo e il controllo della velocità.

Esempio:

```
#include <Servo.h>
Servo myServo;
myServo.attach(9);
mioServo.write(90); // sposta a 90 gradi
```

**Nota Bene:** Staccare sempre il servo quando non è in uso per risparmiare energia. Usa fonti di alimentazione adeguate per i tuoi servo.

# Nozioni di base sulla comunicazione seriale

Comprensione della comunicazione seriale: la comunicazione seriale è essenziale per lo scambio di dati tra Arduino e altri dispositivi.

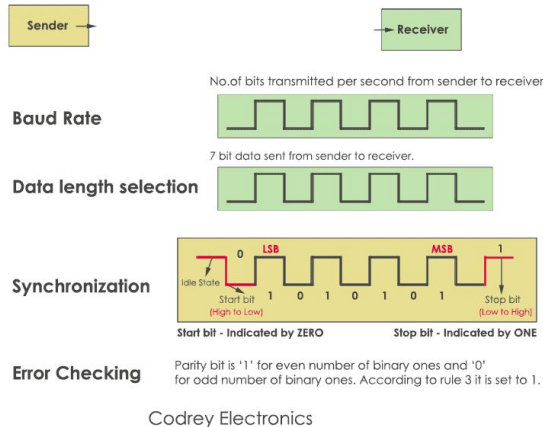
Consente l'invio e la ricezione di dati tramite la connessione USB.

Le funzioni principali includono:

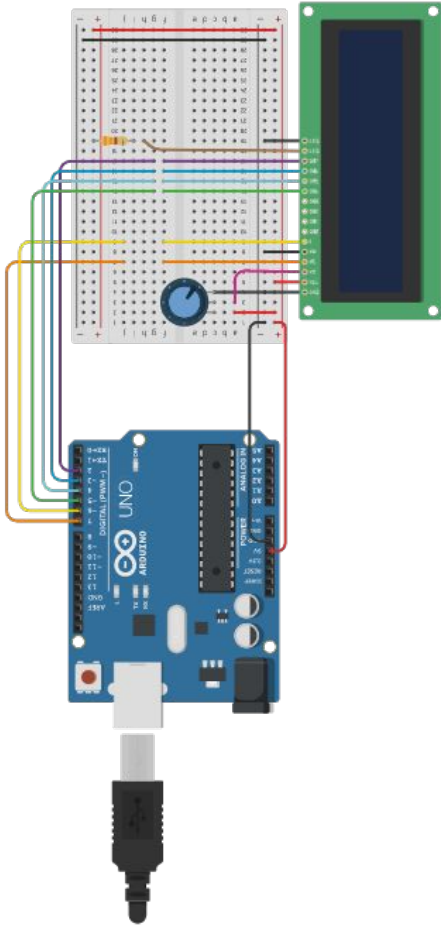
- **Serial.begin():** Inizializza la comunicazione seriale
- **Serial.print():** Invia i dati al monitor seriale
- **Serial.read():** Legge i dati in entrata

Documentazione ufficiale:

<https://docs.arduino.cc/language-reference/en/functions/communication/serial/>



# Gestione display LCD



Lavorare con gli LCD migliora i tuoi progetti fornendo un output visivo.

L'installazione prevede il cablaggio dell'LCD ad Arduino e l'inclusione della libreria appropriata.

Esempio:

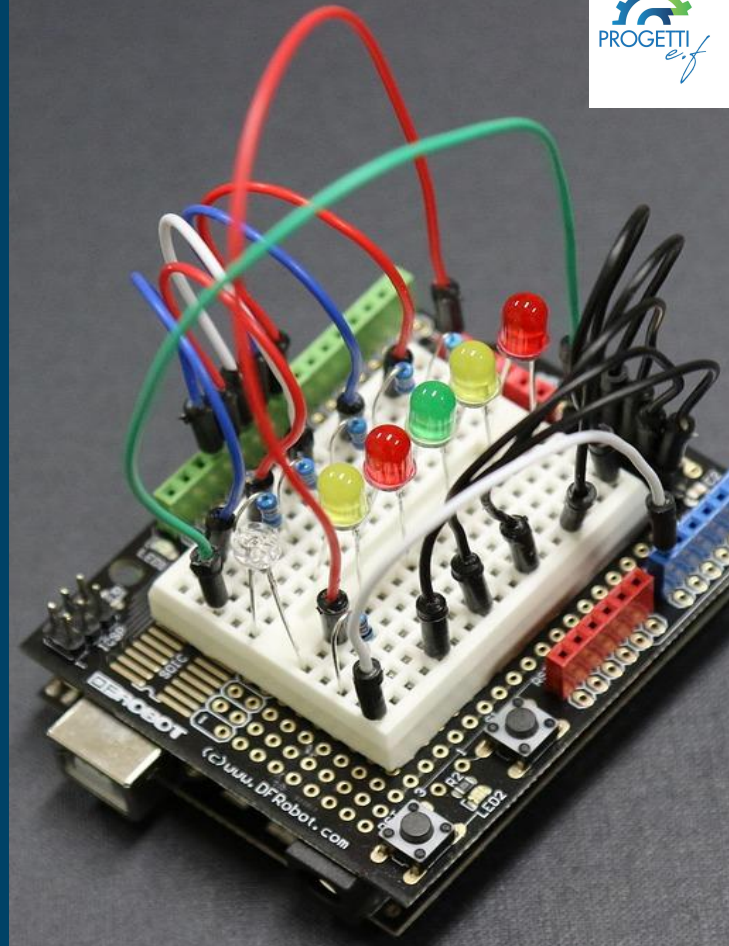
```
#include <LiquidCrystal.h>
int seconds = 0;
LiquidCrystal lcd_1(7, 6, 5, 4, 3, 2);
void setup() {
  lcd_1.begin(16, 2);
  lcd_1.print("hello world!");
}
```

<https://docs.arduino.cc/libraries/liquidcrystal/>

# Sintesi della sezione: Punti chiave

Ricapitoliamo i costrutti e le funzioni importanti trattati:

- Nozioni di base sulla sintassi: **#include**, **#define**, **setup()**, **loop()**
- Istruzioni condizionali: **if**, **while**
- Iterazione: ciclo **for**
- Funzioni native: **digitalRead**, **digitalWrite**, **delay**, ecc.
- Introduzione a **blink.ino** e applicazioni pratiche di servomotori, comunicazione seriale e gestione LCD.



# Considerazioni finali e domande e risposte

La programmazione Arduino apre un mondo di possibilità nell'elettronica e nella robotica.

- Abbraccia il processo di apprendimento e sperimenta i tuoi progetti
- Ora, apriamo la parola a tutte le domande che potresti avere per chiarire la tua comprensione e facilitare la discussione.

