# Funzioni Arduino - Riferimento Approfondito

Questo documento raccoglie una panoramica sintetica ma approfondita delle funzioni utilizzate nei vari sketch Arduino, utile come riferimento durante la scrittura o lettura di progetti basati su Arduino Uno. Per ogni funzione vengono indicati scopo, eventuale appartenenza a librerie, parametri in ingresso e comportamento.

## Funzioni Obbligatorie negli Sketch Arduino

- `void setup()`
  Funzione eseguita una sola volta all'avvio dello sketch. Serve per inizializzare variabili, impostazioni dei pin, comunicazione seriale, ecc.

- `void loop()`
  Funzione eseguita in ciclo continuo dopo `setup()`. Contiene il codice principale del programma Arduino.

## Funzioni Fondamentali di Arduino

- `pinMode(pin, mode)`
  Imposta la modalità del pin.
  Argomenti: `pin` (numero del pin), `mode` (`INPUT`, `OUTPUT`, `INPUT_PULLUP`)

- `digitalWrite(pin, value)`
  Imposta il livello logico di un pin (`HIGH` o `LOW`), si puo usare con un led ad esempio per il controllo dello stato.

- `digitalRead(pin)`
  Restituisce il livello logico attuale del pin (`HIGH` o `LOW`).

- `analogRead(pin)`
  Legge in output un valore analogico da un pin (range 0–1023)

$$\frac{\textit{Resolution of the ADC}}{\textit{System Voltage}} = \frac{\textit{ADC Reading}}{\textit{Analog Voltage Measured}}$$

- `analogWrite(pin, value)`
  Scrive un valore PWM (0–255) su un pin abilitato.

- `delay(time in ms)`
  Pausa di `ms` millisecondi.

- `delayMicroseconds(time in µs)`
  Come `delay()`, ma in microsecondi.
- `millis()`
  Restituisce il tempo trascorso dall'avvio, in millisecondi.
- `micros()`
  Come `millis()`, ma in microsecondi.
- `pulseIn(pin, value)`
  Misura la durata (in µs) di un impulso `HIGH` o `LOW` su un pin.

# Funzioni da Librerie Standard (math.h)

- `round(x)`
  Arrotonda `x` al numero intero più vicino. (non in math.h ma utile da sapere)
- `ceil(x)`
  Arrotonda `x` per eccesso al numero intero più vicino.
- `floor(x)`
  Arrotonda `x` per difetto al numero intero più vicino.
- `powf(x, y)`
  Restituisce `x` elevato alla `y`, versione per float.
- `sqrtf(x)`
  Restituisce la radice quadrata di `x`, versione per float.

# Funzioni Personalizzate e di Libreria

# Funzioni personalizzate con parametri e descrizione

### `void plot_both(void)`

Plotta una coppia di segnali su una finestra grafica (solitamente due array correlati, es. tempo e segnale).

- `void`: Nessun argomento richiesto.

### `void plot_2_signals(double sig_src1_arr, double sig_src2_arr, uint32_t sig_length)`

Plotta due array di segnale, ad esempio per confronto tra segnale originale e filtrato.

- `double sig_src1_arr`: Primo array di input (ad esempio: segnale originale).
- `double sig_src2_arr`: Secondo array di input (ad esempio: segnale elaborato o filtrato).
- `uint32_t sig_length`: Numero di campioni presenti in ciascun array; entrambi devono avere la stessa lunghezza.

### `void calc_running_sum(double sig_src_arr, double sig_dest_arr, uint32_t sig_length)`

Calcola la somma cumulativa di un segnale (integrale discreto).

- `double sig_src_arr`: Array di input del segnale originale.
- `double sig_dest_arr`: Array dove viene salvato il risultato della somma cumulata.
- `uint32_t sig_length`: Numero di campioni del segnale.

### `void calc_first_difference(double sig_src_arr, double sig_dest_arr, uint32_t sig_length)`

Calcola la prima differenza tra campioni successivi (derivata discreta).

- `double sig_src_arr`: Array di input del segnale originale.
- `double sig_dest_arr`: Array dove viene salvata la differenza tra campioni successivi.
- `uint32_t sig_length`: Numero di campioni del segnale.

### `void moving_average(double sig_src_arr, double sig_out_arr, uint32_t sig_length, uint32_t filter_pts)`

Applica una media mobile su un segnale, utile per il filtraggio del rumore.

- `double sig_src_arr`: Array di input del segnale originale.
- `double sig_out_arr`: Array in cui viene salvato il segnale filtrato.
- `uint32_t sig_length`: Numero totale di campioni nel segnale.
- `uint32_t filter_pts`: Numero di punti della finestra di media mobile.

### `void recursive_moving_average(double sig_src_arr, double sig_out_arr, uint32_t sig_length, uint32_t filter_pts)`

Calcola una media mobile ricorsiva (filtro IIR) su un segnale.

- `double sig_src_arr`: Array di input del segnale originale.
- `double sig_out_arr`: Array dove viene salvato il risultato filtrato.
- `uint32_t sig_length`: Numero totale di campioni nel segnale.
- `uint32_t filter_pts`: Costante di filtro (più alta = filtraggio più forte).

### `void calc_sig_dft(double sig_src_arr, double sig_dest_rex_arr, double sig_dest_imx_arr, uint32_t sig_length)`

Calcola la DFT (Discrete Fourier Transform) di un segnale reale.

- `double sig_src_arr`: Array del segnale originale (spazio/tempo).
- `double sig_dest_rex_arr`: Array destinazione per la parte reale della DFT.
- `double sig_dest_imx_arr`: Array destinazione per la parte immaginaria della DFT.
- `uint32_t sig_length`: Numero di campioni del segnale (e della DFT).

### `void calc_sig_idft(double idft_out_arr, double sig_src_rex_arr, double sig_src_imx_arr, uint32_t idft_length)`

Calcola la trasformata inversa della DFT a partire da parte reale e immaginaria.

- `double idft_out_arr`: Array dove viene salvato il segnale ricostruito.
- `double sig_src_rex_arr`: Array della parte reale della DFT.
- `double sig_src_imx_arr`: Array della parte immaginaria della DFT.
- `uint32_t idft_length`: Numero di campioni (uguale alla lunghezza della DFT).

### `void get_dft_output_mag(void)`

Calcola la magnitudo (modulo) della DFT a partire da parte reale e immaginaria.

- `void`: Nessun argomento richiesto.

### `void signal_mean(double sig_src_arr, uint32_t sig_length)`

Calcola il valore medio (media aritmetica) di un segnale.

- `double sig_src_arr`: Array del segnale.
- `uint32_t sig_length`: Numero di campioni nel segnale.

### `void signal_variance(double sig_src_arr, double sig_mean, uint32_t sig_length)`

Calcola la varianza di un segnale, data la sua media.

- `double sig_src_arr`: Array del segnale.
- `double sig_mean`: Valore medio del segnale (precalcolato).
- `uint32_t sig_length`: Numero di campioni nel segnale.

`void signal_std(double variance)`

Calcola la deviazione standard a partire dalla varianza.

- `double variance`: Valore della varianza del segnale.

# Formule Matematiche e Operatori

- `media = somma / N`: calcolo della media aritmetica.
- `varianza = somma_quad - N · media`$^2$: calcolo della varianza.
- `std = sqrt(var)`: deviazione standard.
- `abs(x)`: valore assoluto (libreria `stdlib.h` o `math.h`).
- `sqrt(x)`: radice quadrata (`math.h`).
- `cos(x), sin(x)`: funzioni trigonometriche standard (radiani).

# Elaborazione del Segnale

- `media mobile`: media su finestra mobile di lunghezza fissa.
- `output[i] = input[i] - input[i-1]`: differenza prima (filtro derivativo).
- `output[i] = alpha · input[i] + (1 - alpha) · output[i-1]`: filtro passa basso ricorsivo.
- `convoluzione`: moltiplicazione e somma tra finestra e kernel (spesso implementata con due cicli annidati).
- `DFT`: trasformata discreta di Fourier tramite somme complesse di seni e coseni.

# PWM e Output

- `analogWrite(pin, duty)`: genera segnale PWM.
- `dutyCycle = (onTime / periodo) · 100`: formula per duty cycle in percentuale.

# Debug e Output Seriale

- `Serial.begin(speed)`: Imposta la velocità di trasmissione dati in bit al secondo (baud) per la trasmissione seriale.
- `Serial.print(...)`: stampa continua su monitor seriale.
- `Serial.println(...)`: stampa con a capo.