

JOYSTICK DRAW

Versione <1.0>

[Tale schema di documento va usato come riferimento per lo sviluppo del progetto, secondo il flusso operativo indicato 3.2.29 del compendio didattico riportato nella pagina successiva. Si prevede quindi che tale documento evolva secondo versioni successive ma SOLO LA VERSIONE FINALE dovrà essere presentata come prova scritta di esame]

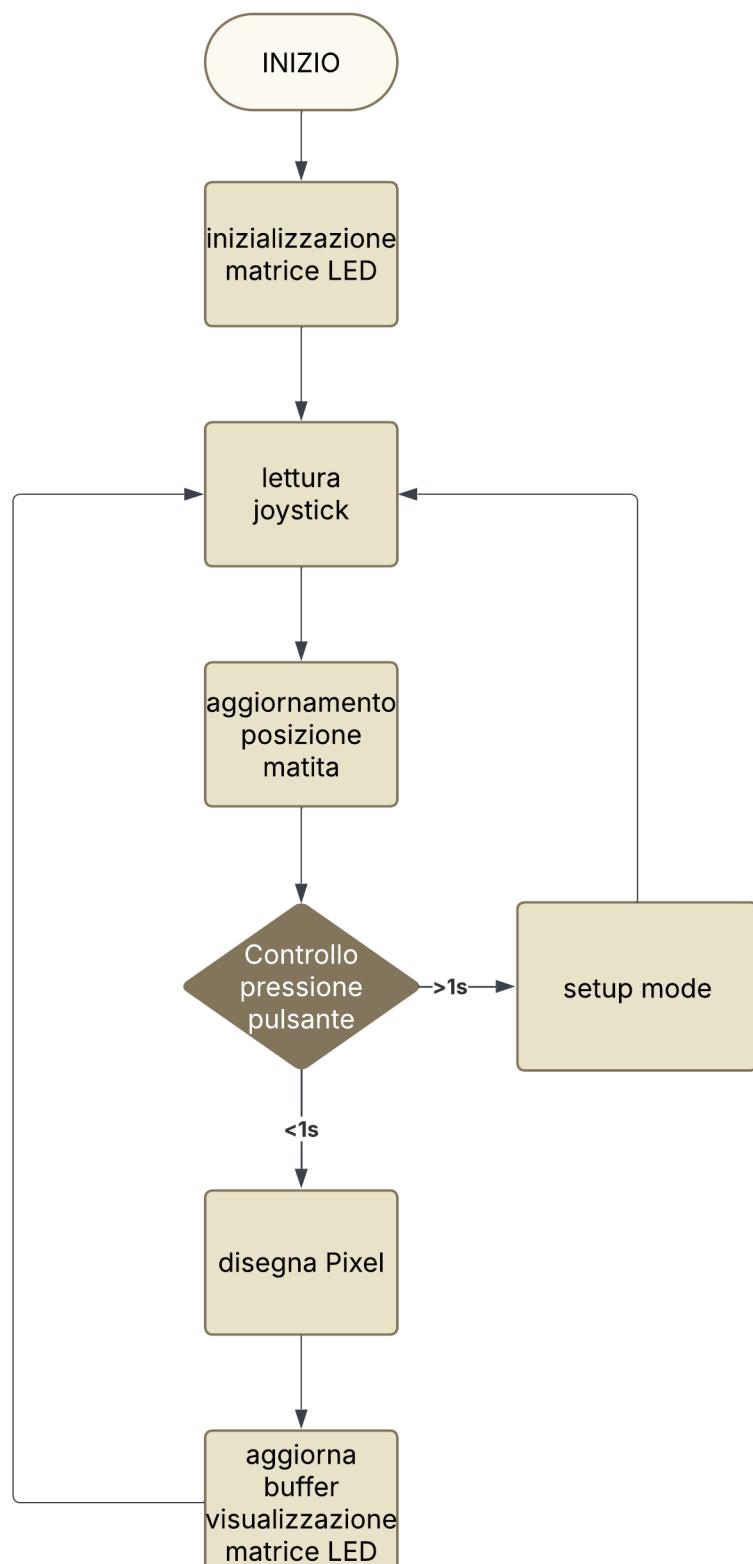
Revisioni

Data	Versione	Descrizione	Autore

INDICE

1. Analisi	3
1.1 Testo assegnato	3
1.2 Analisi del problema	3
1.3 Analisi della soluzione adottata	3
1.4 Analisi delle soluzioni alternative	4
2. Sintesi	5
2.1 Sintesi componente ME1	5
2.2 Sintesi componente ME2	5
.....	5
2.n Sintesi componente MEn	5
3. Sistema complessivo	6
3.1 Manuale utilizzo	6
3.2 Considerazioni finali	6

Come prima cosa , ho creato un flowchart iniziale per analizzare e progettare il problema.



1. Analisi

Il progetto sviluppa un sistema di controllo per una matrice LED, utilizzando una scheda Arduino. Il sistema è controllato tramite un joystick analogico e un pulsante, il quale avvia una modalità di disegno sulla matrice. Il programma permette di modificare il colore di un "pennello" virtuale, disegnando pixel sulla matrice e gestendo il comportamento di un LED con effetti di lampeggio. Inoltre, la posizione del pennello è controllata tramite i valori letti dai pin analogici, mentre il pulsante gestisce il passaggio tra la modalità di disegno e la modalità di setup del colore.

1.1 Testo assegnato

Con riferimento alla seguente matrice di 4x8=32 led WS2812 :

```
{  
  "version": 1,  
  "author": "Massimo Trojani",  
  "editor": "wokwi",  
  "parts": [  
    { "type": "wokwi-arduino-nano", "id": "nano", "top": 335.25, "left": 403.85, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb1", "top": 150, "left": 400, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb2", "top": 150, "left": 420, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb3", "top": 150, "left": 440, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb4", "top": 150, "left": 460, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb5", "top": 170, "left": 400, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb6", "top": 170, "left": 420, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb7", "top": 170, "left": 440, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb8", "top": 170, "left": 460, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb9", "top": 190, "left": 400, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb10", "top": 190, "left": 420, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb11", "top": 190, "left": 440, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb12", "top": 190, "left": 460, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb13", "top": 210, "left": 400, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb14", "top": 210, "left": 420, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb15", "top": 210, "left": 440, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb16", "top": 210, "left": 460, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb17", "top": 230, "left": 400, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb18", "top": 230, "left": 420, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb19", "top": 230, "left": 440, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb20", "top": 230, "left": 460, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb21", "top": 250, "left": 400, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb22", "top": 250, "left": 420, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb23", "top": 250, "left": 440, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb24", "top": 250, "left": 460, "attrs": {} },  
    { "type": "wokwi-neopixel", "id": "rgb25", "top": 270, "left": 400, "attrs": {} },  
  ]  
}
```

```
{
  "type": "wokwi-neopixel", "id": "rgb26", "top": 270, "left": 420, "attrs": {} },
  {"type": "wokwi-neopixel", "id": "rgb27", "top": 270, "left": 440, "attrs": {} },
  {"type": "wokwi-neopixel", "id": "rgb28", "top": 270, "left": 460, "attrs": {} },
  {"type": "wokwi-neopixel", "id": "rgb29", "top": 290, "left": 400, "attrs": {} },
  {"type": "wokwi-neopixel", "id": "rgb30", "top": 290, "left": 420, "attrs": {} },
  {"type": "wokwi-neopixel", "id": "rgb31", "top": 290, "left": 440, "attrs": {} },
  {"type": "wokwi-neopixel", "id": "rgb32", "top": 290, "left": 460, "attrs": {} }
],
"connections": [
  ["rgb2:DOUT", "rgb1:DIN", "green", [ "h0" ] ],
  ["rgb4:DOUT", "rgb3:DIN", "green", [ "h0" ] ],
  ["rgb3:DOUT", "rgb2:DIN", "green", [ "h0" ] ],
  ["rgb6:DOUT", "rgb5:DIN", "green", [ "h0" ] ],
  ["rgb7:DOUT", "rgb6:DIN", "green", [ "h0" ] ],
  ["rgb8:DOUT", "rgb7:DIN", "green", [ "h0" ] ],
  ["rgb10:DOUT", "rgb9:DIN", "green", [ "h0" ] ],
  ["rgb11:DOUT", "rgb10:DIN", "green", [ "h0" ] ],
  ["rgb12:DOUT", "rgb11:DIN", "green", [ "h0" ] ],
  ["rgb14:DOUT", "rgb13:DIN", "green", [ "h0" ] ],
  ["rgb15:DOUT", "rgb14:DIN", "green", [ "h0" ] ],
  ["rgb16:DOUT", "rgb15:DIN", "green", [ "h0" ] ],
  ["rgb13:DOUT", "rgb12:DIN", "green", [ "h0.47", "v-14", "h79.53" ] ],
  ["rgb9:DOUT", "rgb8:DIN", "green", [ "h-2", "v-12.41", "h82" ] ],
  ["rgb5:DOUT", "rgb4:DIN", "green", [ "h-2.82", "v-14.94", "h82.82" ] ],
  ["rgb17:DOUT", "rgb16:DIN", "green", [ "h-1.87", "v-16.79", "h81.87" ] ],
  ["rgb18:DOUT", "rgb17:DIN", "green", [ "h0" ] ],
  ["rgb19:DOUT", "rgb18:DIN", "green", [ "h0" ] ],
  ["rgb20:DOUT", "rgb19:DIN", "green", [ "h0" ] ],
  ["rgb22:DOUT", "rgb21:DIN", "green", [ "h0" ] ],
  ["rgb21:DOUT", "rgb20:DIN", "green", [ "h-1.87", "v-15.54", "h81.87" ] ],
  ["rgb30:DOUT", "rgb29:DIN", "green", [ "h0" ] ],
  ["rgb32:DOUT", "rgb31:DIN", "green", [ "h0" ] ],
  ["rgb28:DOUT", "rgb27:DIN", "green", [ "h0" ] ],
  ["rgb27:DOUT", "rgb26:DIN", "green", [ "h0" ] ],
  ["rgb26:DOUT", "rgb25:DIN", "green", [ "h0" ] ],
  ["rgb25:DOUT", "rgb24:DIN", "green", [ "h-1.47", "v-14.02", "h81.47" ] ],
  ["rgb28:DIN", "rgb29:DOUT", "green", [ "h1.7", "v13.53", "h-81.7" ] ],
  ["rgb32:DIN", "nano:8", "green", [ "h8.93", "v34.34", "h-29.44" ] ],
  ["rgb31:DOUT", "rgb30:DIN", "green", [ "h0" ] ],
  ["rgb24:DOUT", "rgb23:DIN", "green", [ "h0" ] ],
  ["rgb23:DOUT", "rgb22:DIN", "green", [ "h0" ] ]
],
"dependencies": {}
}
```

ed al capitolo 4.3.4.4 delle dispense, realizzare un sistema di disegno comandato da un joystick analogico del tipo: <https://docs.wokwi.com/parts/wokwi-analog-joystick> con le seguenti regole:

a) partendo dall'angolo in basso a sinistra, è possibile spostare la matita (rappresentata da un

led lampeggiante ad una frequenza di 4 impulsi al secondo) nella matrice seguendo la direzione indicata dal joystick, un pixel alla volta oppure, se si mantiene la direzione per più di un secondo, ad una velocità di due pixel al secondo (=un pixel ogni 500 ms)

b) cliccando brevemente sul bottone di "selezione" del joystick viene scritto il pixel alla posizione corrente con il colore attualmente impostato

c) tenendo premuto per più di un secondo il bottone di selezione del joystick si entra in modalità "setup" del colore della matita secondo le seguenti regole:

c1) La componente "Rosso" aumenta di intensità spostando il joystick in direzione Nord-Ovest e diminuisce nella direzione opposta ovvero Sud-Est

c2) La componente "Verde" aumenta di intensità spostando il joystick in direzione Nord e diminuisce nella direzione opposta ovvero Sud

c3) La componente "Blu" aumenta di intensità spostando il joystick in direzione Nord-Est e diminuisce nella direzione opposta ovvero Sud-Ovest

Si esce da questa modalità cliccando sul bottone di selezione

Si consiglia di seguire il percorso di analisi e progettazione indicato in figura 3.2.29 delle dispense, secondo il paradigma:

- Minore risorse impiegate per la macchina
- Maggior sforzo per la progettazione e sviluppo.

In particolare:

a) componenti HW-SW dovranno essere "Automi" indipendenti che comunicano con variabili globali

b) Le cadenze temporali (ritardi ecc.) dovranno essere governati dall'ISR del clock

c) Non sono consentiti calcoli laboriosi nelle ISR. Tutte le elaborazioni pesanti devono essere svolte nella funzione loop ed essere "sincronizzate" con gli eventi in background.

1.2 Analisi del problema

Elementi computazionali:

Specifiche numeriche calcolabili:

- **Matrice di LED:** La matrice è composta da 4 colonne e 8 righe, quindi ci sono 32 LED ($4 * 8 = 32$) da gestire. Ogni LED ha 3 componenti (RGB), quindi in totale ci sono 96 valori (32 LED * 3 componenti per LED).
- **Intervallo di tempo:** Il lampeggio del LED per simulare la matita avviene a 4 impulsi al secondo (intervallo di 250 ms). Questo valore deve essere gestito correttamente per mantenere il tempo di lampeggio costante.
- **Velocità di movimento:** La velocità di movimento della matita è:
 - 1 pixel ogni 125 ms (8 pixel al secondo) quando il joystick è spostato.
 - 2 pixel al secondo quando il joystick è mantenuto premuto nella stessa posizione per più di un secondo.

Valori di ingresso ed uscita e casi estremi:○ **Ingressi:**

Joystick X e Joystick Y: Valori analogici letti tramite ADC, che indicano la direzione in cui spostare la matita.

Pulsante di selezione : Un valore digitale che indica quando il pulsante è premuto per scrivere un pixel o entrare in modalità setup.

○ **Uscite:**

- La matrice di LED, che visualizza la posizione e il colore della matita.
- Eventuali segnali di stato (ad esempio, colore della matita).

○ **Casi estremi:**

- **Posizione fuori dai limiti della matrice:** Se la posizione della matita supera i limiti della matrice, è necessario bloccare il movimento o reimpostare la posizione.
- **Valori estremi del joystick:** Il joystick fornisce valori analogici che possono essere molto vicini a 0 o 1023. È necessario trattare questi valori in modo da evitare letture anomale a letture fuori scala.

Vincoli imposti da eventuali componenti fisici esterni:○ **Hardware (Arduino Nano, matrice LED WS2812):**

- L'Arduino Nano ha un numero limitato di pin e risorse di calcolo. La matrice di LED WS2812 richiede una gestione precisa del timing per la comunicazione (protocollo di data di tipo PWM).
- L'ADC dell'Arduino Nano ha una risoluzione di 10 bit, il che implica una lettura da 0 a 1023. I valori letti dai pin del joystick devono essere mappati per determinare i movimenti di pixel.

○ **Joystick:**

- I valori analogici del joystick devono essere interpretati correttamente per determinare i movimenti su due assi (X e Y). Ogni movimento deve essere tradotto in un cambiamento di posizione sulla matrice di LED.

○ **Componente del LED (WS2812):**

- La matrice di LED è a indirizzo individuale, quindi ogni LED richiede un controllo preciso del colore. Le operazioni di scrittura dei pixel devono essere eseguite in modo da non interferire con altre operazioni in esecuzione sull'Arduino.

Flusso logico del sistema:

Inizializzazione:

- L'Arduino Nano imposta i pin e inizializza l'ADC per leggere i valori del joystick.
- Viene configurato il timer per gestire gli intervalli di tempo per il lampeggio del LED (4 Hz) e il movimento della matita.
- La matrice di LED viene inizializzata.

Movimento della matita:

- Quando il joystick è spostato, la posizione della matita viene aggiornata di un pixel in direzione della lettura del joystick.
- Se il joystick è mantenuto premuto per più di un secondo, la velocità di movimento della matita aumenta (2 pixel al secondo).
- Se il pulsante di selezione viene premuto brevemente, il pixel nella posizione corrente viene "disegnato".

Modalità setup del colore:

- Se il pulsante di selezione viene tenuto premuto per oltre 1 secondo, entra in modalità setup del colore.
- In modalità setup, il movimento del joystick modifica i valori RGB della matita. Le direzioni del joystick devono essere mappate sui cambiamenti dei singoli colori (rosso, verde, blu).
- Quando il pulsante di selezione viene rilasciato, il sistema esce dalla modalità setup.

Limitazioni:

- Il movimento della matita è limitato dalla dimensione della matrice (4x8), quindi è necessario verificare che la posizione del cursore non vada oltre i confini.
- La velocità di aggiornamento dei LED deve essere bilanciata per evitare il sovraccarico di calcoli in tempo reale.
- La modalità di setup deve bloccare il movimento, in modo tale da poter cambiare il colore del Led senza contemporaneamente spostare l'aggiornamento del cursore.
- Il protocollo WS2812 è un protocollo di comunicazione basato su un segnale di tipo PWM (Pulse Width Modulation) per controllare i singoli LED RGB della matrice. Ogni LED nella matrice richiede 24 bit (8 bit per ogni componente: Rosso, Verde, Blu) per essere controllato, con un ciclo di comunicazione preciso.

Ogni bit della comunicazione WS2812 ha una durata di circa 1.25 microsecondi:

- **Bit alto:** Circa 0.8 μ s.
- **Bit basso:** Circa 0.4 μ s.

La durata di trasmissione per un singolo LED è quindi di circa 24 bit * 1.25 μ s = 30 μ s. Con 32 LED, la durata totale della trasmissione è di circa 960 μ s (0.96ms). È importante che l'aggiornamento dei LED non interferisca con altre operazioni in corso, come la gestione del timer.

1.3 Analisi della soluzione adottata

Il sistema si evolve attraverso vari **stati** che sono determinati dalla posizione del joystick e dalla pressione del pulsante di selezione. Il diagramma di stato complessivo descrive il comportamento del sistema e i passaggi tra i vari stati in funzione degli input.

- **Stato iniziale** : All'avvio, il sistema è in attesa di azioni da parte dell'utente (movimento del joystick o pressione del pulsante).
- **Stato di movimento** : Quando il joystick viene spostato, la matita si muove sulla matrice. Se il joystick viene mantenuto in posizione per più di un secondo, il movimento accelera.
- **Stato di scrittura** : Quando il pulsante di selezione viene premuto, il pixel nella posizione corrente della matita viene scritto nella matrice.
- **Stato di configurazione colore (Setup Mode)**: Quando il pulsante di selezione viene tenuto premuto per più di un secondo, il sistema entra in modalità configurazione del colore. In questo stato, i movimenti del joystick regolano i valori RGB della "matita".

Moduli di Elaborazione Indipendenti

Il sistema è progettato con moduli di elaborazione indipendenti, che operano su variabili di input e restituiscono variabili di output. Ogni modulo è responsabile per una specifica funzione del sistema. I moduli principali sono:

Modulo di Lettura del Joystick:

- **Ingresso**: Valori analogici letti dai pin JOY_X_PIN e JOY_Y_PIN.
- **Uscita**: Direzione di movimento (dx, dy) o valori di configurazione del colore.

Il modulo legge i valori analogici del joystick e determina la direzione in cui deve muoversi la matita o modifica i colori in modalità configurazione.

Modulo di Controllo della Matita (Movimento):

- **Ingresso**: Direzione del joystick (dx, dy) e stato di modalità di movimento.
- **Uscita**: Posizione della matita (pencilX, pencilY).

Gestisce il movimento della matita sulla matrice di LED, aggiornando la posizione in base agli input del joystick e alla modalità di velocità.

Modulo di Scrittura del Pixel:

- **Ingresso**: Posizione della matita (pencilX, pencilY) e colore della matita (RGB).
- **Uscita**: Aggiornamento della matrice di LED per scrivere il colore del pixel nella posizione attuale.

Registra un pixel alla posizione corrente nella matrice di LED con il colore selezionato.

Modulo di Configurazione del Colore:

- **Ingresso**: Valori analogici letti dal joystick (per configurare i colori) e stato della modalità di configurazione.
- **Uscita**: Nuovi valori di colore RGB per la "matita".

Permette di cambiare i colori della matita regolando le componenti RGB in base alla posizione del joystick.

In particolare, secondo le seguenti regole richieste:

c1) La componente "Rosso" aumenta di intensità spostando il joystick in direzione Nord-Ovest e diminuisce nella direzione opposta ovvero Sud-Est

c2) La componente "Verde" aumenta di intensità spostando il joystick in direzione Nord e diminuisce nella direzione opposta ovvero Sud

c3) La componente "Blu" aumenta di intensità spostando il joystick in direzione Nord-Est e diminuisce nella direzione opposta ovvero Sud-Ovest

Si esce da questa modalità cliccando sul bottone di selezione

Modulo di Gestione del Pulsante:

- **Ingresso:** Stato del pulsante di selezione (SELECT_PIN).
- **Uscita:** Attivazione della modalità di scrittura pixel o configurazione colore.

Gestisce la pressione del pulsante per passare tra le modalità di scrittura pixel e configurazione del colore.

Analisi delle Specifiche Tecniche dei Componenti

Macro e funzioni per manipolazione registri

Macro scritturaPin:

Ho definito questa macro per scrivere un valore HIGH o LOW a un pin specifico. Utilizzando l'operatore bitwise OR (|=) e bitwise AND (&=), la macro modifica direttamente il registro del port di output senza passare per la funzione digitalWrite(). L'uso di queste operazioni bitwise è più efficiente perché permette di manipolare direttamente i registri del microcontrollore, evitando l'overhead associato all'uso di funzioni più generiche. L'uso di queste operazioni bitwise evita il sovraccarico associato alla gestione più astratta che Arduino fa internamente con digitalWrite(). Questo è particolarmente utile nei sistemi che richiedono una gestione precisa degli interrupt, con il timer e il lampeggio del LED. Inoltre, l'accesso diretto ai registri è più veloce rispetto all'uso delle funzioni di alto livello di Arduino, riducendo il consumo di tempo macchina.

Macro letturaPin:

Questa macro consente di leggere lo stato di un pin (HIGH o LOW) in modo simile a digitalWrite(), ma eseguito direttamente sui registri del microcontrollore. La macro utilizza l'operatore bitwise AND (&) per controllare se il bit specificato è 1 (HIGH) o 0 (LOW). Analogamente alla macro di scrittura, la lettura diretta tramite i registri è molto più veloce rispetto all'uso di digitalWrite(), che comporta un sovraccarico aggiuntivo. Inoltre, l'uso di questa macro consente di leggere più pin contemporaneamente in un unico registro di input, migliorando l'efficienza.

Funzione setupPins()

La funzione setupPins() è stata implementata per configurare i pin come input o output. In questa funzione, i pin sono configurati manualmente tramite i registri di configurazione del microcontrollore, senza utilizzare le funzioni interne di Arduino come pinMode().

I registri DDRx, PORTx, e PINx sono direttamente associati alla gestione dei pin a livello hardware, e l'uso di queste operazioni bitwise consente una configurazione e un controllo dei pin molto più rapidi. Arduino fornisce funzioni più generiche (come pinMode() e digitalWrite()) che però aggiungono un overhead in termini di tempo di esecuzione e memoria.

Con la gestione diretta dei registri, è possibile configurare in modo più preciso i pin, come l'attivazione dei pull-up interni per i pin di ingresso, senza passare attraverso le funzioni di Arduino.

Nel caso dell'uso di timer e ISR, il controllo diretto dei pin è essenziale per garantire che non ci siano ritardi indesiderati causati da funzioni di livello superiore. La manipolazione diretta dei registri permette di ottimizzare il tempo di esecuzione delle operazioni.

Joystick :

Il joystick fornisce valori analogici tramite l'ADC dell'Arduino Nano, con una risoluzione di 10 bit (0–1023).

Lettura dei valori analogici:

I valori letti dal joystick vengono ottenuti con la funzione

```
uint16_t readAnalog(uint8_t pin) {  
    ADMUX = (1 << REFS0) | pin;  
    ADCSRA |= (1 << ADSC);  
    while (ADCSRA & (1 << ADSC));  
    return ADC;  
}
```

In questo caso, l'ADC legge il valore analogico dai pin `JOY_X_PIN` e `JOY_Y_PIN`, che corrispondono agli assi X e Y del joystick. I valori letti vengono utilizzati per determinare la direzione del movimento del cursore nella matrice.

Mappatura dei movimenti:

I valori analogici letti sono confrontati con delle soglie predefinite (`thresholdHigh` e `thresholdLow`) per determinare il movimento del cursore:

```
int dx = (joyX > thresholdHigh) - (joyX < thresholdLow);  
int dy = (joyY > thresholdHigh) - (joyY < thresholdLow);
```

I valori di dx e dy indicano il movimento della matita lungo gli assi X e Y.

Ad esempio, se `joyX` è maggiore di `thresholdHigh`, la matita si sposterà verso destra. Analogamente, se `joyY` è maggiore di `thresholdHigh`, la matita si sposterà verso l'alto.

Pulsante di Selezione :

Il pulsante è utilizzato per scrivere un pixel o per entrare in modalità configurazione del colore.

La pressione prolungata viene utilizzata per entrare in modalità setup, mentre una pressione breve registra un pixel. Il debouncing del pulsante potrebbe essere necessario per evitare letture errate.

Matrice di LED WS2812:

La matrice è composta da 32 LED (4 colonne e 8 righe), ognuno con 24 bit di controllo (8 bit per ciascun canale RGB). Il protocollo WS2812 richiede che i dati siano inviati tramite un segnale PWM con una precisione temporale elevata.

La trasmissione dei dati ai LED deve rispettare i cicli temporali del protocollo WS2812, con ogni bit che richiede circa 1.25µs per essere trasmesso. Questo impone restrizioni sul tempo che l'Arduino può dedicare ad altre operazioni, come la gestione del timer e il movimento della matita.

Limiti del protocollo WS2812

Il protocollo WS2812 è un protocollo di comunicazione seriale a basso livello, utilizzato per controllare i LED RGB. I dati per ciascun LED sono inviati in serie con un segnale PWM, e ogni LED richiede 24 bit per essere controllato (8 bit per ciascun canale: Rosso, Verde, Blu).

Durata di un ciclo di comunicazione: Ogni bit di dati richiede 1.25 microsecondi (μ s) per essere trasmesso, con una sequenza di 24 bit per LED:

$$24\text{bit} \times 1.25\mu\text{s} = 30\mu\text{s}$$

Per 32 LED, la durata totale della trasmissione dei dati è di circa 960 μ s.

Implicazioni sui cicli di macchina: La trasmissione dei dati ai LED WS2812 deve essere eseguita in modo preciso per rispettare i cicli temporali del protocollo.

Per garantire che i dati vengano inviati correttamente, l'Arduino deve operare in modalità di scrittura diretta tramite codice assembly inline:

```
asm volatile (
    "lwhile0:\n\t" // Ciclo esterno per caricamento byte
    "ld r24, %a[vp]+ \n\t" // Carico prossimo byte
    "ldi r25, 8 \n\t"      // Contatore bit
    ...
);
```

L'operazione di trasmissione dei dati ai LED WS2812 non può interferire con le altre operazioni in esecuzione nel sistema, in particolare con la gestione del timer. Per evitare conflitti, la trasmissione dei dati ai LED viene eseguita in una sezione di codice assembly che non blocca l'esecuzione dell'ISR.

Timer (Timer2):

Il timer viene utilizzato per gestire il lampeggio del LED e il movimento della matita. È configurato con un prescaler di 256, che genera un overflow circa ogni 16ms.

Frequenza di lampeggio e movimenti in Hz

La frequenza di lampeggio del LED simula il movimento della "matita" sulla matrice. Il LED lampeggia a 4 impulsi al secondo, ovvero 4Hz, e il movimento della matita è gestito tramite un interrupt di Timer2.

Calcolo della Frequenza

Per far lampeggiare il LED a 4 impulsi al secondo (4Hz), bisogna calcolare il tempo tra ciascun impulso e quindi configurare correttamente l'interrupt del timer per ottenere questa frequenza.

Frequenza richiesta: 4 impulsi al secondo (4Hz).

Il tempo per impulso è:

$$\text{Tempo per impulso} = \frac{1}{\text{Frequenza}} = \frac{1}{4} = 250 \text{ ms}$$

Il Timer2 dell'Arduino è configurato con un prescaler di 256, il che significa che ogni ciclo del timer ha una durata di circa 4.096 ms.

Prescaler: 256

Durata di un ciclo del timer: $256 \times 16 \mu\text{s} = 4096 \mu\text{s} = 4096 \text{ ms}$

Quindi, per ottenere un tempo di 250 ms tra ogni impulso, bisogna determinare quanti overflow del Timer2 sono necessari:

$$\frac{250 \text{ ms}}{4.096 \text{ ms}} \approx 61$$

Quindi, per ottenere un intervallo di 250 ms tra ogni impulso, dobbiamo aspettare 61 overflow del Timer2.

Nel codice, la frequenza di lampeggio viene gestita tramite un **interrupt di Timer2**:

```
ISR(TIMER2_OVF_vect) {
    static uint8_t blinkCount = 0;
    if (blinkCount >= 61) { // Ogni 61 overflow ≈ 250ms
        blinkState = !blinkState;
        updateDisplayBuffer();
        blinkCount = 0;
    }
    blinkCount++;
}
```

Ogni volta che il timer genera un overflow, la variabile blinkCount viene incrementata. Quando blinkCount raggiunge 61, il LED lampeggia e viene aggiornato il buffer del display.

La gestione degli interrupt di overflow del timer permette di controllare il lampeggio e il movimento della matita, senza bloccare il sistema. È fondamentale che le operazioni eseguite nell'ISR siano leggere e veloci per evitare di compromettere le prestazioni del sistema.

Ogni interrupt del Timer2 avviene ogni 4.096 ms (con prescaler di 256). All'interno dell'ISR, viene controllato il lampeggio del LED e il movimento della matita:

```
ISR(TIMER2_OVF_vect) {
    static uint8_t blinkCount = 0;
    static uint8_t moveCount = 0;
    static uint16_t buttonPressCount = 0;
    ...
}
```

L'ISR viene eseguito per 61 overflow per ottenere il lampeggio a 4Hz. Inoltre, la gestione del movimento della matita viene aggiornata ogni 125 ms (31 overflow).

Le operazioni nell'ISR sono limitate a compiti leggeri, come l'inversione del lampeggio del LED e l'incremento del contatore per il movimento. Tutte le operazioni più complesse, come il controllo del joystick e la scrittura dei pixel, sono eseguite nella funzione loop(), evitando di bloccare il sistema con calcoli pesanti nell'ISR.

Algoritmi Computazionali

Gli algoritmi principali sono descritti dai seguenti flowchart semplificati :

Movimento della Matita:

Controllo Modalità Setup (setupMode)

Se setupMode è attivo, salta il movimento della matita e passa alla modifica del colore.

Se setupMode non è attivo, continua con il movimento.

Lettura Joystick (Read Joystick)

Leggi i valori analogici dai pin JOY_X_PIN e JOY_Y_PIN tramite ADC.

Calcolo Direzione (Calculate Direction)

Calcola la direzione del joystick:

- o dx (spostamento orizzontale) se JOY_X_PIN è superiore o inferiore ai limiti.
- o dy (spostamento verticale) se JOY_Y_PIN è superiore o inferiore ai limiti.

Controllo Intervallo di Movimento (Check Move Interval)

Se sono passati almeno 500ms dall'ultimo movimento (lastMoveTime), aggiorna la posizione della matita:

- o Aggiungi dx a pencilX (posizione orizzontale).
- o Aggiungi dy a pencilY (posizione verticale).

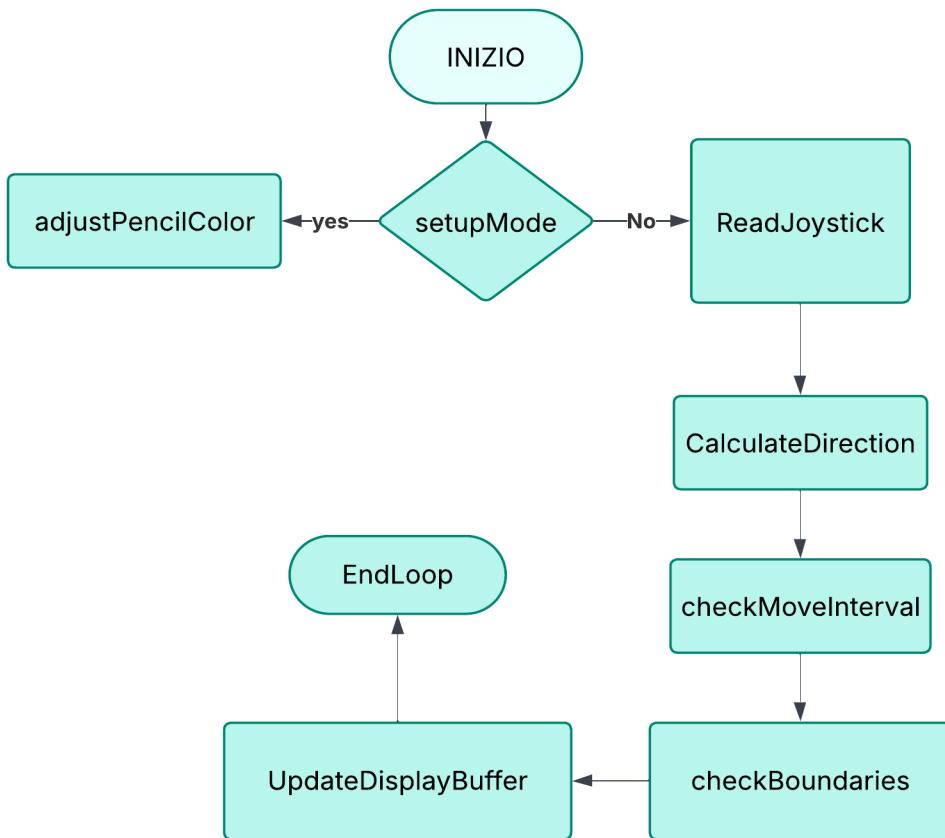
Aggiorna lastMoveTime con il tempo corrente.

Verifica Limiti (Check Boundaries)

Assicuro che la matita non esca dai limiti della matrice ($0 \leq pencilX < MATRIX_COLS$ e $0 \leq pencilY < MATRIX_ROWS$).

Aggiorna Buffer Display (Update Display Buffer)

Se la matita è in movimento, aggiorna il displayBuffer con la nuova posizione e il colore della matita (se lampeggiante).



Scrittura del Pixel:

Controllo Modalità Setup (setupMode)

Se setupMode è attivo, salta la scrittura del pixel e passa alla modifica del colore.
Se setupMode non è attivo, continua con la scrittura del pixel.

Lettura del Pulsante (Button Press)

Leggi lo stato del pulsante di selezione.

Se il pulsante è premuto brevemente, continua con la scrittura del pixel.

Se il pulsante è premuto a lungo, entra in modalità setup per la selezione del colore.

Verifica Durata Pressione (Button Press Duration)

Se la durata della pressione è inferiore a 1000ms, considera la pressione come breve e procede alla scrittura del pixel.

Se la durata della pressione è maggiore di 1000ms, attiva la modalità di configurazione del colore.

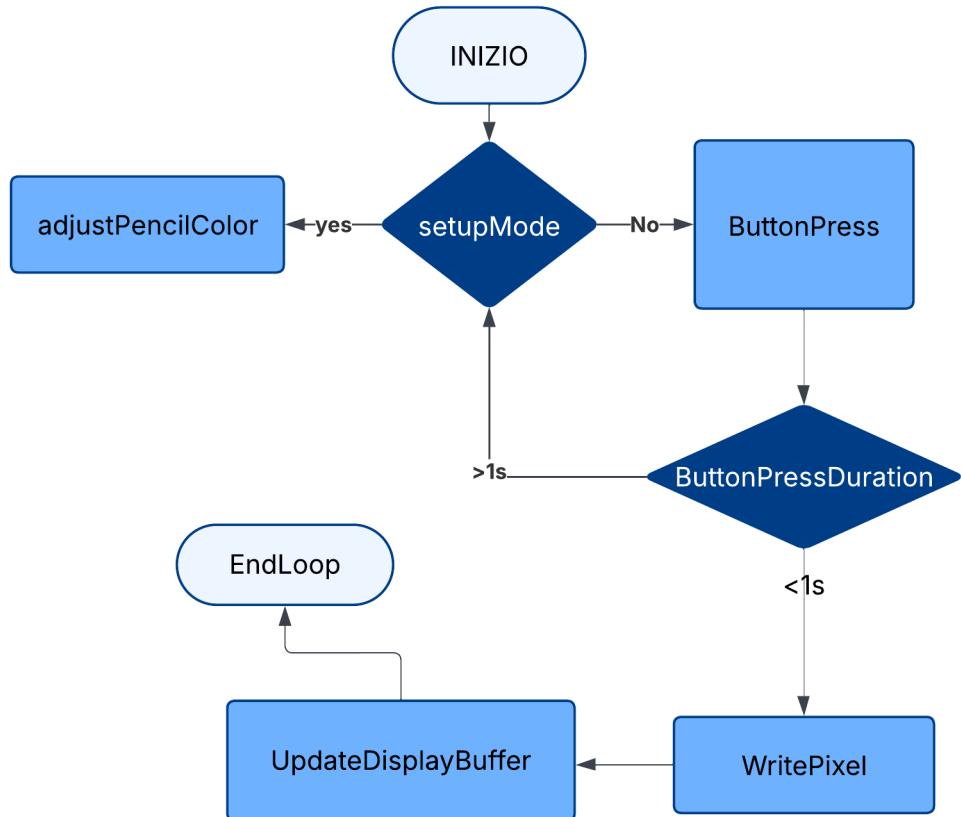
Scrittura del Pixel (Write Pixel)

Determina l'indice del pixel nella matrice di LED in base alle coordinate pencilX e pencilY.

Aggiorna il canvas (buffer della matrice) con il colore della matita (pencilR, pencilG, pencilB) nella posizione specificata.

Aggiorna il Buffer Display (Update Display Buffer)

Copia il canvas nel displayBuffer per prepararlo all'invio alla matrice di LED.



Configurazione del Colore:

Inizializzazione della modalità di configurazione del colore, che viene attivata quando il pulsante viene premuto a lungo.

Controllo Modalità Setup (setupMode)

Se setupMode è attivo, il sistema è in modalità di configurazione del colore.

Se setupMode non è attivo, il sistema esegue altre operazioni.

Lettura del Joystick (Joystick Input)

Leggi i valori analogici dei pin del joystick (JOY_X_PIN e JOY_Y_PIN).

Questi valori determinano come il colore della matita deve cambiare.

Aggiustamento del Colore (Adjust Pencil Color)

Componente Rossa :

Se il joystick si muove verso Nord-Ovest (valore basso su X e alto su Y), aumenta la componente rossa.

Se il joystick si muove verso Sud-Est (valore alto su X e basso su Y), diminuisce la componente rossa.

Componente Verde :

Se il joystick si muove verso Nord (valore alto su Y), aumenta la componente verde.

- Se il joystick si muove verso Sud (valore basso su Y), diminuisce la componente verde.

Componente Blu :

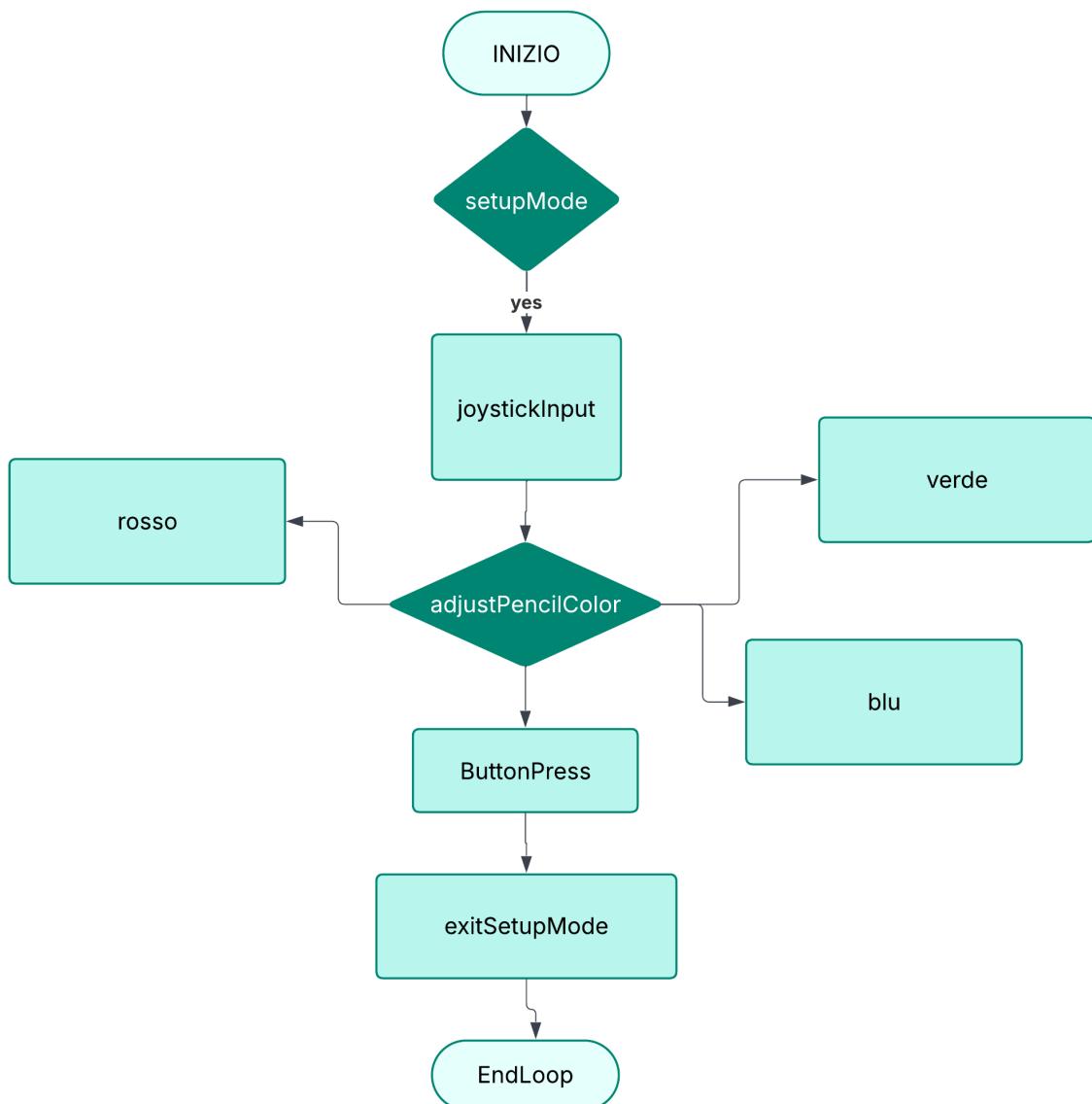
- Se il joystick si muove verso Nord-Est (valore alto su X e alto su Y), aumenta la componente blu.
- Se il joystick si muove verso Sud-Ovest (valore basso su X e basso su Y), diminuisce la componente blu.

Verifica Uscita dalla Modalità Setup (Exit Setup Mode)

Se il pulsante viene rilasciato (dopo una pressione lunga), esci dalla modalità di configurazione del colore e torna alla modalità normale.

Se il pulsante è premuto nuovamente, mantieni la modalità setup attiva.

La configurazione del colore è completa, e la modalità setup continua finché il pulsante è premuto. Se il pulsante viene rilasciato, il sistema torna alla modalità normale.



Gestione del Pulsante:

Lettura Stato Pulsante (Read Button State)

- Leggi lo stato del pulsante di selezione tramite il pin digitale associato.
- Verifica se il pulsante è premuto o rilasciato.

Controllo Pressione del Pulsante (Check Button Press)

- Se il pulsante è premuto, inizia a misurare il tempo di pressione.
- Se il pulsante è rilasciato, verifica la durata della pressione.

Verifica Durata della Pressione (Check Press Duration)

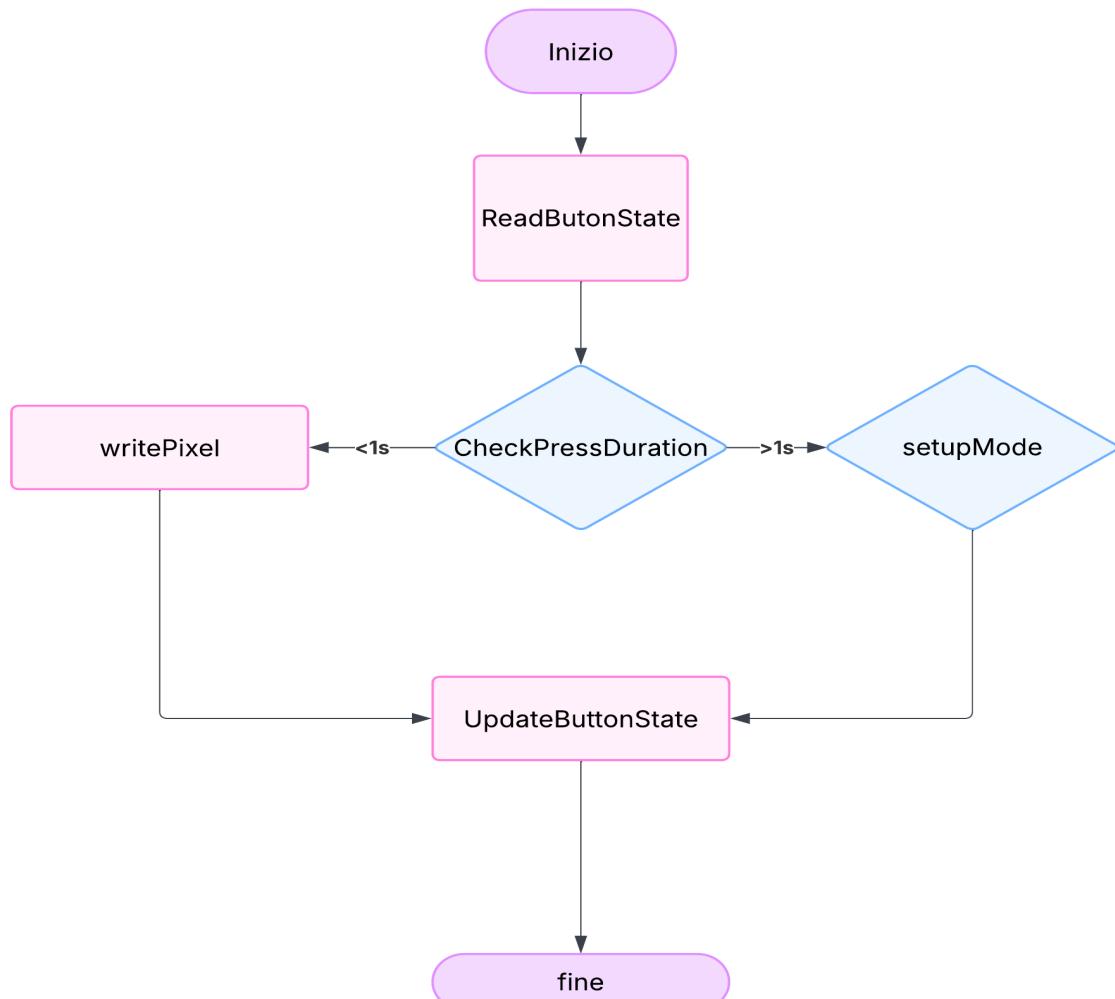
- Se la durata della pressione è inferiore a 1000 ms , Pressione breve: Esegui l'azione di scrivere un pixel nella posizione corrente della matita.
- Se la durata della pressione è maggiore di 1000 ms, Pressione lunga: Entra in modalità setup per modificare il colore della matita.

Esecuzione delle Azioni

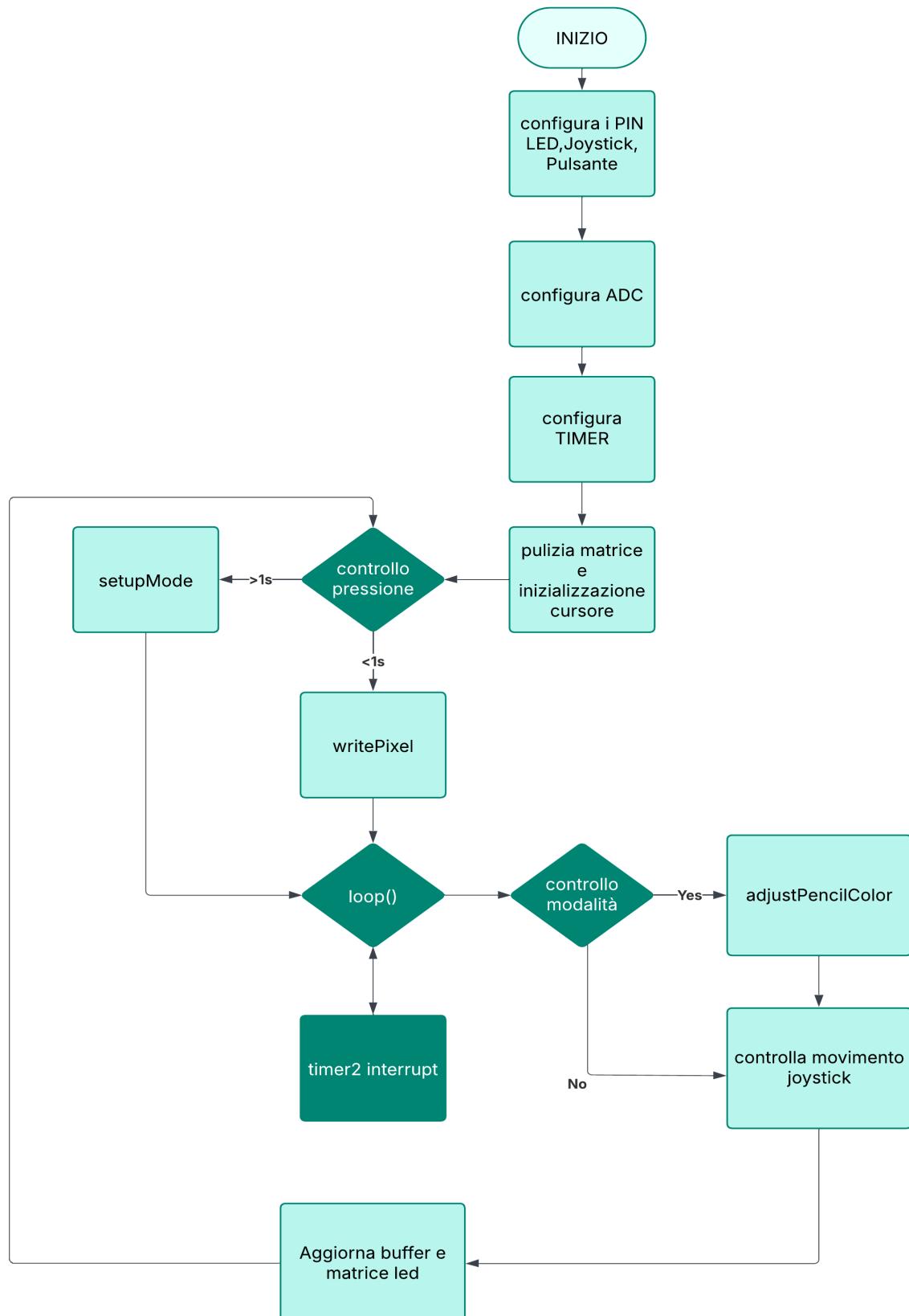
- Pressione breve: Scrivi il pixel alla posizione corrente della matita (utilizzando la funzione writePixel()).
- Pressione lunga: Cambia lo stato di setupMode (attiva o disattiva la modalità setup).

Aggiornamento Stato Pulsante (Update Button State)

- Dopo aver eseguito l'azione in base alla durata della pressione, resetta il conteggio della durata della pressione e aggiorna lo stato del pulsante.



FLOWCHART SEMPLIFICATO DEL CODICE



1.4 Analisi delle soluzioni alternative

Utilizzo di millis() e gestione temporale software

Una delle soluzioni alternative che sarebbe stata possibile implementare, sarebbe stata quella di gestire i ritardi e gli eventi temporali utilizzando esclusivamente millis(), senza l'uso di Timer2,

Funzionamento della soluzione alternativa:

Lampeggio del cursore: Utilizzare millis() per controllare il lampeggio del cursore ogni 125 ms. In pratica, questo approccio impiega il tempo di sistema (millis()) per calcolare il ritardo, evitando l'uso di timer hardware.

Movimento del cursore: Utilizzare millis() per verificare che siano passati almeno 500 ms dal movimento precedente, gestendo così il movimento del cursore con un ritardo minimo.

Controllo del pulsante: La durata della pressione del pulsante viene misurata tramite millis() per distinguere tra una pressione breve e lunga.

Questa soluzione elimina la necessità di configurare Timer2 o utilizzare interrupt, utilizzando esclusivamente il **timer software** gestito da millis().

In particolare:

Trasmissione alla Matrice LED

```
void showStrip() {
    cli();
    for (uint16_t i = 0; i < NUM_LEDS * 3; i++) {
        uint8_t byte = displayBuffer[i];
        for (uint8_t bit = 0x80; bit; bit >>= 1) {
            if (byte & bit) {
                PORTD |= (1 << 6);
                asm volatile("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
                PORTD &= ~(1 << 6);
                asm volatile("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
            } else {
                PORTD |= (1 << 6);
                asm volatile("nop\n\t""nop\n\t");
                PORTD &= ~(1 << 6);
                asm volatile("nop\n\t""nop\n\t""nop\n\t""nop\n\t");
            }
        }
    }
    sei();
    delayMicroseconds(60);
}
```

- Invia i dati alla matrice LED in modalità bit-banging.
- Il codice è ottimizzato per il timing dei LED, evitando librerie e minimizzando overhead.

Motivazioni per lo scarto:

Imprecisione Temporale: Sebbene millis() sia utile per la gestione di eventi temporali, ha una bassa precisione rispetto all'uso di Timer2. L'accuratezza di millis() dipende dal ciclo del sistema e può essere soggetta a lievi imprecisioni, in particolare quando il carico di lavoro aumenta o se ci sono altre operazioni in esecuzione nel loop. Questo potrebbe portare a un leggero slittamento degli intervalli temporali, influenzando negativamente il comportamento del lampeggio del LED o del movimento del cursore.

Mancanza di Indipendenza Temporale: Con millis(), tutte le operazioni temporali vengono gestite nel loop principale, il che significa che la gestione del tempo è dipendente dall'esecuzione del codice in quel momento. Se il sistema ha bisogno di gestire più eventi temporali (come il movimento e il lampeggio), il ciclo del loop potrebbe diventare più complicato, con potenziale riduzione della reattività. Ad esempio, se il sistema è occupato nell'eseguire operazioni più lunghe (come il calcolo dei pixel o la gestione della matrice LED), l'aggiornamento temporale potrebbe non avvenire con la stessa precisione.

Sovraccarico di Funzioni Software: Utilizzare millis() comporta la necessità di verificare continuamente i tempi nel loop principale e di gestire multiple variabili di stato per eventi temporali diversi (come il lampeggio, il movimento del cursore, il pulsante). Questo approccio introduce un sovraccarico software in quanto richiede frequenti verifiche e aggiornamenti del tempo, che avrebbero potuto essere gestiti in modo più efficiente utilizzando Timer2 e interrupt hardware.

Prestazioni Minori: L'alternativa richiede più interazioni software per la gestione temporale, aumentando il carico di lavoro della CPU. Rispetto all'approccio con Timer2, che permette una gestione asincrona del tempo tramite interrupt, l'uso di millis() impone più verifiche nel loop e comporta una gestione più complessa dei ritardi temporali. Di conseguenza, la reattività del sistema potrebbe diminuire con l'aumento del numero di eventi temporali da gestire.

2. Sintesi

Microcontrollore Arduino Nano

Il cuore del sistema è un **Arduino Nano**, scelto per la sua compattezza, il basso consumo energetico e le capacità sufficienti per gestire la matrice di LED e il joystick.

Caratteristiche:

- **Processore:** ATmega328P
- **Memoria:** 32KB di memoria flash, 2KB di SRAM, 1KB di EEPROM.
- **I/O:** 14 pin digitali di input/output (di cui 6 PWM), 8 pin analogici di input.
- **Comunicazione:** Gestisce la comunicazione con il joystick, il pulsante di selezione e la matrice di LED.

Joystick Analogico

Il **joystick analogico** è utilizzato per determinare la direzione del movimento della "matita" (LED lampeggiante) sulla matrice. Questo dispositivo fornisce due segnali analogici, uno per ogni asse (X e Y), che vengono letti dal microcontrollore tramite l'ADC (convertitore analogico-digitale).

- **Potenziometri:** Il joystick è composto da due potenziometri (uno per ciascun asse) che regolano i valori tra 0 e 1023 (risoluzione a 10 bit) per ciascun asse.
- **Comportamento:** La posizione del joystick determina il movimento del cursore sulla matrice: valori più alti o più bassi sui rispettivi assi indicano il movimento verso destra/sinistra e verso l'alto/basso.
- **Pin di collegamento:** Ogni asse (X e Y) del joystick è collegato a un pin analogico dell'Arduino per la lettura.

Pulsante di Selezione

Il **pulsante di selezione** è utilizzato per due operazioni principali:

Scrivere un pixel: Una pressione breve del pulsante di selezione scrive un pixel alla posizione corrente della matita sulla matrice.

Attivare la modalità setup del colore: Una pressione lunga del pulsante attiva la modalità in cui è possibile cambiare il colore della "matita".

- **Collegamento:** Collegato a un pin digitale dell'Arduino, con l'uso di un pull-up interno per una gestione semplice del segnale.

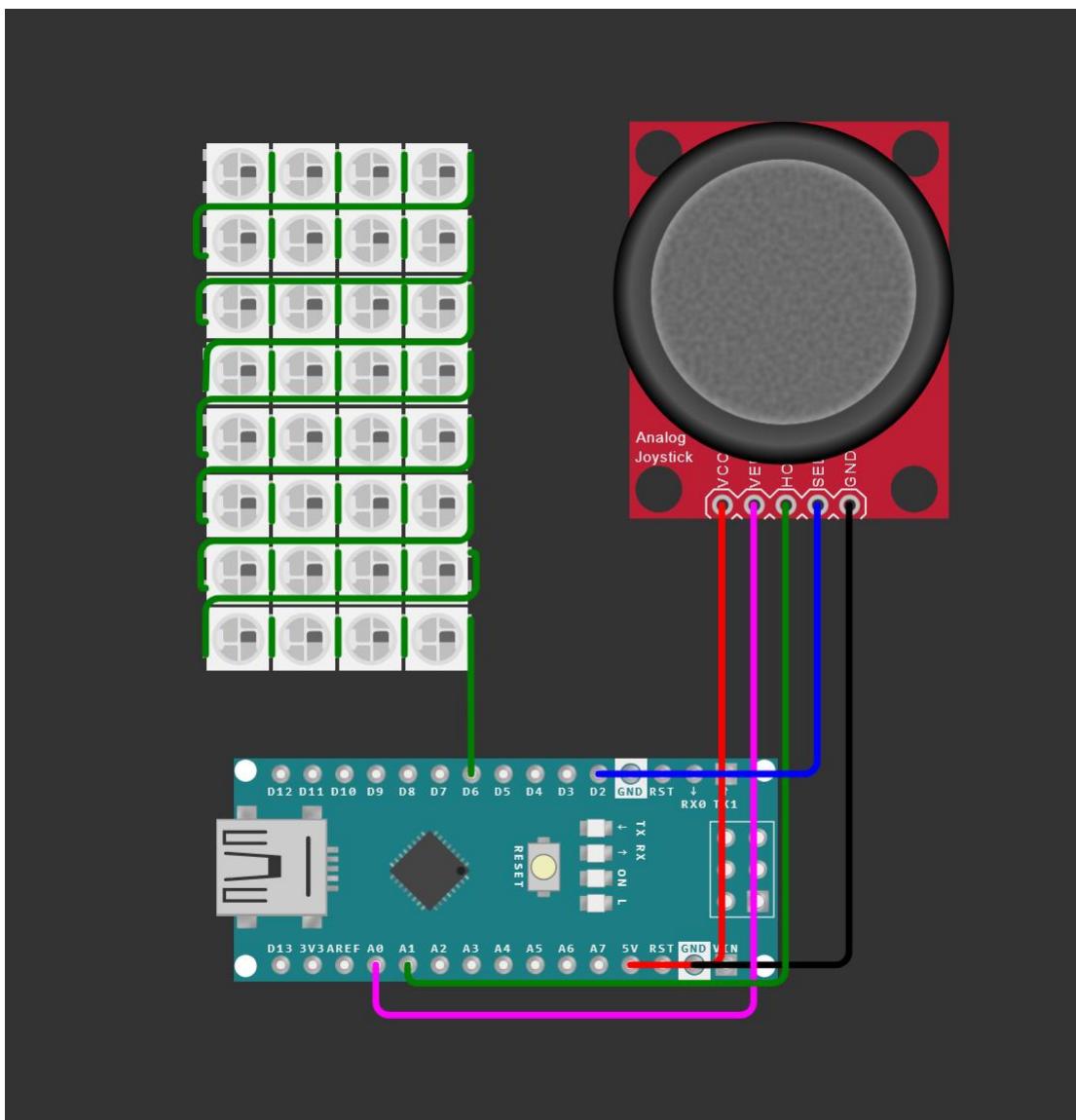
Matrice LED WS2812

La **matrice LED WS2812** è utilizzata per visualizzare il disegno creato dall'utente. Ogni LED è controllato individualmente e può essere programmato per visualizzare un colore RGB specifico. I LED WS2812 utilizzano un protocollo di comunicazione seriale a singolo filo che richiede un timing preciso per la corretta trasmissione dei dati.

Caratteristiche:

- **Numero di LED:** La matrice è composta da 32 LED (4 colonne per 8 righe).
- **Protocollo:** I dati vengono trasmessi utilizzando il protocollo WS2812, che richiede la trasmissione di 24 bit per LED (8 bit per ciascun canale RGB).
- **Controllo:** La comunicazione con i LED avviene tramite **bit-banging** in modalità seriale, con ogni byte dei dati che viene inviato bit per bit.

Cablaggio e Connessioni



Il cablaggio prevede collegamenti tra i pin di Arduino e le rispettive connessioni al joystick, e alla matrice di LED.

Connessioni principali:

- **Joystick:** il VCC è collegato alla scheda a 5V.
I Pin analogici VERT e HORIZ ai pin A0 e A1 di Arduino.
Il pin del pulsante di selezione SEL è collegato al pin nano2.
Il pin di terra GND al GND della scheda.
- **Matrice di LED:** Il pin di controllo della matrice WS2812 è collegato al pin nano6 di Arduino, mentre i dati vengono trasmessi in serie per ciascun LED.

Sintesi delle Componenti Hardware:

Arduino Nano:

Il microcontrollore centrale che gestisce tutte le operazioni del sistema.

L'Arduino Nano è una delle schede di sviluppo più popolari nel mondo dell'elettronica e della prototipazione rapida. È una versione compatta della classica scheda Arduino, pensata per progetti che richiedono una dimensione ridotta, ma che mantengono comunque le potenzialità e la versatilità di un microcontrollore. È basata su un chip **ATmega328P** (come l'Arduino Uno), ma con una disposizione più piccola .

Caratteristiche principali:

Microcontrollore: ATmega328P

Frequenza di clock: **16 MHz**

Memoria Flash: **32 KB** (di cui 2 KB utilizzati dal bootloader)

SRAM: 2 KB

EEPROM: **1 KB

Pin di I/O:

14 pin digitali di input/output (di cui 6 possono essere usati come **uscite PWM**)

8 pin analogici di ingresso (risoluzione 10 bit)

Capacità di alimentazione: 5V o 3.3V (tramite il pin **RAW**)

Alimentazione:

Tensione di alimentazione: 5V o 3.3V

Può essere alimentato tramite:

Porta USB (5V)

Pin RAW per alimentazione esterna (7-12V)

Connessioni di Programmazione:

La scheda Arduino Nano può essere programmata via USB tramite il modulo **FTDI** integrato (convertitore da USB a seriale).

Può anche essere programmata tramite il **pin ISP** (In-System Programming) per la programmazione diretta del microcontrollore, ma solitamente viene usata la programmazione via USB.

Comunicazione:

1 porta UART (comunicazione seriale)

1 interfaccia I2C (SCL e SDA)

1 interfaccia SPI (per la comunicazione con dispositivi come sensori, display e altri dispositivi di memoria)

Programmazione:

Utilizza l'IDE Arduino, che fornisce una piattaforma semplice e potente per scrivere e caricare il codice nel microcontrollore.

Compatibile con **linguaggio C++**, ma con l'uso di librerie predefinite che semplificano molte operazioni comuni come la gestione di LED, sensori, motori e altre periferiche.

Pin di alimentazione:

5V: alimentazione a 5V, viene fornito tramite USB o tramite il pin **RAW** se alimentato esternamente.

3.3V: alimentazione a 3.3V, disponibile su uno dei pin.

GND: pin di terra comune per i collegamenti di circuito.

RAW: pin per alimentazione esterna (quando si usa un adattatore o batteria esterna).

Joystick Analogico:

Utilizzato per determinare il movimento del cursore e la selezione del colore. È composto da due potenziometri (uno per l'asse orizzontale e uno per l'asse verticale), e un pulsante di selezione integrato che può essere utilizzato per eseguire azioni specifiche.

Caratteristiche principali:

Potenziometri per la Direzione (X, Y):

- **Potenziometro X (Orizzontale):** Regola il movimento lungo l'asse orizzontale (destra/sinistra).
- **Potenziometro Y (Verticale):** Regola il movimento lungo l'asse verticale (su/giù).
- Ogni potenziometro fornisce un segnale analogico che varia in base alla posizione del joystick. Il valore analogico (letto tramite un **convertitore analogico-digitale** (ADC)) varia generalmente da 0 a 1023 (su 10 bit), dove:
 - **Valore medio (circa 512)** corrisponde alla posizione centrale del joystick.
 - **Valori estremi (0 o 1023)** indicano la posizione completamente a sinistra/destra (per l'asse X) o su/giù (per l'asse Y).

Pulsante di Selezione (SEL):

- Il joystick è dotato di un pulsante integrato che può essere premuto per eseguire azioni, come selezionare o attivare una modalità. Il pin **SEL** invia un segnale **LOW** quando il pulsante è premuto e **HIGH** quando non lo è.
- Il pulsante di selezione è utile in molte applicazioni dove è necessario fare una scelta o attivare una funzione con il joystick.

Connessioni:

- **3 Pin di Output:**
 - **X:** Pin analogico per l'asse orizzontale.
 - **Y:** Pin analogico per l'asse verticale.
 - **SEL:** Pin digitale per il pulsante di selezione.
- **GND:** Pin di terra, che deve essere collegato al pin GND del microcontrollore per il corretto funzionamento del joystick.
- **VCC:** Pin di alimentazione, generalmente collegato a 5V per i modelli comuni.

Matrice LED WS2812: Visualizza il disegno creato dall'utente, con un controllo preciso dei colori.

Controllo Individuato per Ogni LED:

- Ogni LED nella matrice può essere controllato singolarmente, permettendo di impostare il colore e l'intensità di ciascun LED in modo indipendente.
- Questo è possibile grazie al protocollo **WS2812**, che utilizza un singolo filo di dati seriali per inviare informazioni a ciascun LED in modo sequenziale.

Struttura del LED:

- Ogni singolo LED è composto da un chip **integrato** che gestisce i 3 canali di colore (RGB), ed è composto da 3 diodi LED (Rosso, Verde, Blu).
- Ogni LED ha una capacità di visualizzare **256 livelli di intensità** per ciascun colore (Rosso, Verde e Blu), il che consente la combinazione di 16.7 milioni di colori diversi (256 x 256 x 256).

Protocollo di Comunicazione:

- I LED WS2812 utilizzano un **protocollo di comunicazione seriale a singolo filo**, dove ogni LED riceve i dati in una sequenza seriale.
- Ogni LED riceve 24 bit (8 bit per ciascun colore: Rosso, Verde, Blu).
- **Dati seriali:** Ogni LED riceve il dato che definisce i suoi colori, e poi trasmette i dati successivi al LED successivo nella catena.

Alimentazione:

- I LED WS2812 operano con **tensione di alimentazione 5V**.
- Ogni LED consuma circa **60mA** a piena luminosità (tutti i colori al massimo).
- È possibile alimentare catene di LED WS2812, ma è importante considerare la potenza totale richiesta, specialmente per grandi matrice di LED.

Comunicazione:

- La comunicazione tra i microcontrollori (come Arduino) e la matrice LED avviene tramite un protocollo one-wire (un filo) che invia dati seriali.
- Per il controllo della matrice di LED, viene inviata una sequenza di bit che definiscono il colore di ciascun LED.

Frequenza di Aggiornamento:

- I LED WS2812 possono aggiornare i colori a una velocità elevata (fino a 800 Kbps o più), il che li rende adatti per applicazioni dinamiche come la visualizzazione di effetti luminosi.
- Il **tempo di latenza** tra l'invio dei dati e la visualizzazione effettiva del colore dipende dalla lunghezza della stringa di LED.

Colore:

- Ogni LED può essere impostato su una combinazione di colori tra **Rosso, Verde e Blu**. Ogni colore può avere 256 intensità, quindi sono possibili **16.7 milioni di colori** ($256 * 256 * 256$).

Velocità di Trasmissione:

- La velocità di trasmissione dei dati è **800 Kbps**, che consente di aggiornare rapidamente i colori anche su matrici di grandi dimensioni.
- Ogni LED riceve i 24 bit di colore (8 bit per ogni canale RGB) in un ciclo di trasmissione.

3. Sistema complessivo

LINK WOKWI : <https://wokwi.com/projects/397302239952129025>