

RELAZIONE: Tetris Arduino

Enrico Ferraiolo 0001191698

Laurea Magistrale in Informatica

Corso: Laboratorio di Making
a.a. 2024-2025

Indice

1	Introduzione	3
2	Componenti Hardware	3
2.1	Microcontrollore	3
2.2	Display a matrice LED - Campo di Gioco	3
2.3	Display LCD - Informazioni di Gioco	3
2.4	Controlli	4
2.4.1	Controlli Infrarossi	4
2.4.2	Encoder Rotativo	4
3	Il Gioco	4
3.1	Tetramini	4
4	Ambienti di Sviluppo	5
5	Setup Hardware	5
5.1	Display a matrice LED (MAX7219)	6
5.2	Display LCD 16x2	6
5.3	Ricevitore Infrarossi (IR)	7
5.4	Encoder Rotativo	7
6	Build del Progetto	7
6.1	Struttura del Progetto	8
6.2	Simulatore	9
6.3	Hardware Reale	10
7	Screenshot del Gioco	10

1 Introduzione

Il gioco Tetris è uno dei puzzle game più celebri di sempre: l'utente deve ruotare e spostare pezzi geometrici ("tetramini") che cadono, completando linee orizzontali per ottenere punti.

L'obiettivo di questo progetto è realizzare una versione giocabile su Arduino di Tetris, utilizzando:

- un display a matrice LED 8x8 (MAX7219) per il campo di gioco
- un display LCD 16x2 per visualizzare punteggio e stato
- un telecomando IR e un encoder rotativo per i controlli

Lo scopo del progetto è quindi implementare una versione completamente funzionante del gioco Tetris su Arduino con diversi moduli di input e output.

2 Componenti Hardware

Di seguito vengono elencati e descritti i componenti hardware utilizzati per il progetto.

2.1 Microcontrollore

Il microcontrollore utilizzato è il **Elegoo UNO R3**, una scheda equivalente all'Arduino UNO che fornisce tutte le funzionalità necessarie per il progetto.

2.2 Display a matrice LED - Campo di Gioco

Il display a matrice LED è un modulo **MAX7219** con configurazione 8x8. Ogni LED della matrice può essere controllato individualmente, consentendo di:

- visualizzare il campo di gioco
- rappresentare i tetramini in movimento
- mostrare le celle occupate

Ogni LED rappresenta una singola cella del campo di gioco.

2.3 Display LCD - Informazioni di Gioco

Per visualizzare informazioni testuali viene utilizzato un display LCD 16x2 (**LCD 1602**). A schermo vengono mostrati:

- **Punteggio**: punteggio attuale del giocatore
- **Stato**: stato corrente del gioco (in corso, pausa, terminato)
- **Velocità**: velocità di caduta dei tetramini
- **Istruzioni ausiliarie**: informazioni utili per il giocatore

2.4 Controlli

Il progetto implementa due diversi sistemi di controllo per offrire un'interazione flessibile.

2.4.1 Controlli Infrarossi

Il telecomando IR consente di inviare istruzioni a distanza tramite segnali infrarossi, decodificati da un apposito ricevitore. I comandi principali sono:

Tabella 1: Funzioni dei tasti del telecomando IR

Tasto	Funzione
POWER	Accensione/spegnimento del gioco
FAST BACK	Movimento del tetramino a sinistra
FAST FORWARD	Movimento del tetramino a destra
PAUSE	Pausa/ripresa del gioco
VOL+	Rotazione del tetramino in senso orario
VOL-	Rotazione del tetramino in senso antiorario

2.4.2 Encoder Rotativo

L'encoder rotativo offre un controllo analogico della velocità di gioco:

- **Rotazione in senso orario:** aumento della velocità di caduta
- **Rotazione in senso antiorario:** diminuzione della velocità di caduta

3 Il Gioco










Tetris è un puzzle game in cui il giocatore deve manipolare dei tetramini che cadono dall'alto nel campo di gioco. L'obiettivo è:

- Ruotare e posizionare i tetramini per completare le righe orizzontali
- Quando una riga è completa, essa scompare e il giocatore guadagna punti
- Il gioco termina quando i tetramini impilati raggiungono la parte superiore del campo

3.1 Tetramini

I tetramini implementati nel gioco sono rappresentati nella [Tabella 2](#):

Tabella 2: Rappresentazione dei tetramini

Pezzo	Codici binari	W×H	Forma
I	0b1111	4×1	
J	0b0111	3×2	
	0b0100		
L	0b0111	3×2	
	0b0001		
O	0b0011	2×2	
	0b0011		
T	0b0111	3×2	
	0b0010		
S	0b0110	3×2	
	0b0011		
Z	0b0011	3×2	
	0b0110		

4 Ambienti di Sviluppo

Il progetto è stato sviluppato per essere eseguibile in due configurazioni:

- **Hardware fisico:** con la scheda Elegoo UNO R3 (o equivalente) e tutti i moduli connessi
- **Simulatore:** per test e sviluppo senza hardware fisico

La configurazione dell'ambiente avviene tramite la costante `PRODUCTION` nel file sorgente principale:

- `PRODUCTION = true`: per l'utilizzo con hardware fisico
- `PRODUCTION = false`: per l'utilizzo con il simulatore

Questa differenziazione è necessaria principalmente per gestire i diversi codici infrarossi generati dal telecomando nei due ambienti.

5 Setup Hardware

Questa sezione descrive il collegamento dei vari componenti al microcontrollore.

Uno schema di collegamento è mostrato nella Figura 1.

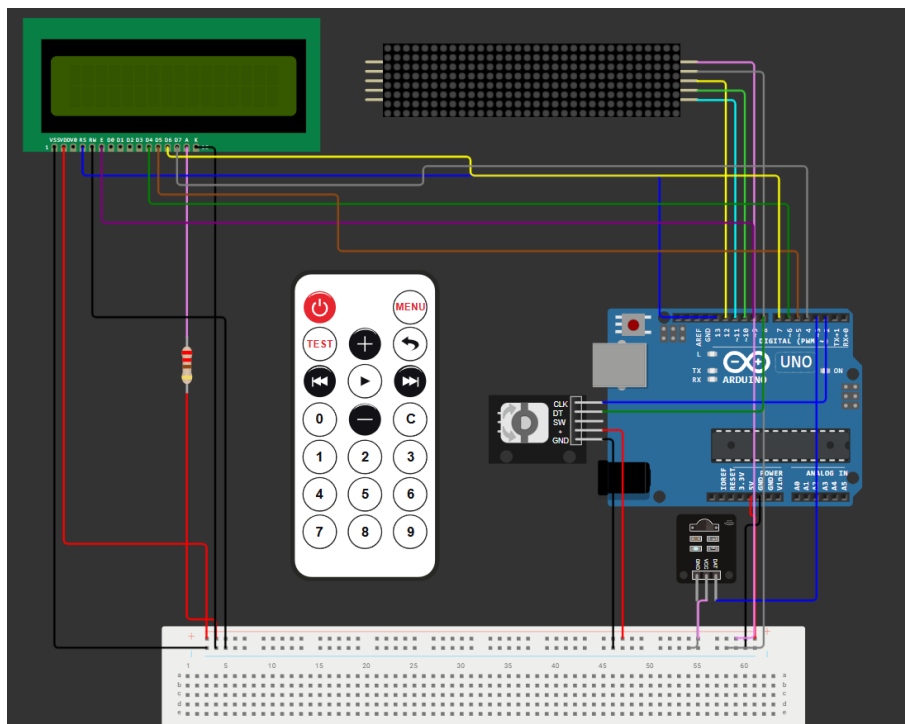


Figura 1: Schema di collegamento dei componenti hardware

5.1 Display a matrice LED (MAX7219)

Collegamenti tra il modulo MAX7219 e il microcontrollore:

Tabella 3: Collegamenti Matrix (MAX7219) - Microcontrollore

Matrix Pin	Microcontrollore Pin
VCC	5V
GND	GND
DIN	Pin 12
CS	Pin 10
CLK	Pin 11

5.2 Display LCD 16x2

Collegamenti tra il modulo LCD 1602 e il microcontrollore:

Tabella 4: Collegamenti LCD Display (16x2) - Microcontrollore

LCD Pin	Microcontrollore Pin
RS	Pin 13
E	Pin 9
D4	Pin 6
D5	Pin 5
D6	Pin 7
D7	Pin 4
VSS	GND
VDD	5V
RW	GND
A (Anodo)	5V (attraverso resistenza da 220 Ω)
K (Catodo)	GND

5.3 Ricevitore Infrarossi (IR)

Collegamenti tra il modulo ricevitore IR e il microcontrollore:

Tabella 5: Collegamenti IR Receiver Module - Microcontrollore

IR Pin	Microcontrollore Pin
VCC	5V
GND	GND
OUT/Data	Pin 3

5.4 Encoder Rotativo

Collegamenti tra l'encoder rotativo e il microcontrollore:

Tabella 6: Collegamenti Rotary Encoder Module - Microcontrollore

Encoder Pin	Microcontrollore Pin
CLK	Pin 2
DT	Pin 8
SW (Switch)	Non utilizzato
VCC	5V
GND	GND

6 Build del Progetto

Il progetto può essere eseguito in due modalità: **simulatore** e **hardware fisico reale**.

Per eseguire il progetto in modalità simulatore il file da eseguire sarà `src/main.cpp`, mentre per eseguire il progetto su hardware reale il file da eseguire sarà `src/main/main.ino`. Questo perché l'IDE di Arduino vuole che il file da iniettare sulla scheda sia contenuto in una cartella con lo stesso nome del file.

I file sono identici se non per la riga di codice che definisce se il progetto è in produzione o meno.

- **Simulatore:** `PRODUCTION = false`
- **Hardware reale:** `PRODUCTION = true`

La differenziazione data da questa riga di codice è necessaria per gestire i codici infrarossi generati dal telecomando nei due ambienti.

Infatti, per il simulatore Wokwi, i codici infrarossi sono diversi rispetto a quelli inviati dal telecomando reale.

Se si volesse cambiare il telecomando, basterebbe rimappare i codici di production (`PRODUCTION_CODES`) nel file `lib/utils/utils.h`.

6.1 Struttura del Progetto

Il progetto è organizzato seguendo una struttura modulare che facilita lo sviluppo e la manutenzione del codice.

Di seguito è riportata la struttura delle directory e dei file principali:


```

TETRIS_ARDUINO
├── lib
│   └── utils
│       └── utils.h
│           └── (Libreria utility)
├── report
│   ├── (Report directory)
│   ├── media
│   │   └── (Media directory)
│   ├── report.tex
│   │   └── (Report in Latex)
│   ├── report.pdf
│   │   └── (Report prodotto)
├── src
│   ├── (Source code directory)
│   ├── main
│   │   ├── (Report directory)
│   │   ├── main.ino
│   │   │   └── (Arduino IDE entry point)
│   └── main.cpp
│       └── (PlatformIO entry point)
├── diagram.json
│   └── (File di configurazione Wokwi - circuiteria)
├── platformio.ini
│   └── (File di configurazione PlatformIO)
├── README.md
│   └── (Documentazione)
└── wokwi.toml
    └── (File di configurazione Wokwi - simulatore)

```

6.2 Simulatore

Di seguito sono riportati i passaggi per eseguire il progetto in modalità simulatore.

1. Aprire VSCode

2. Installare l'estensione [Wokwi](#)
3. Installare l'estensione [PlatformIO](#)
4. Aprire la cartella `tetris-arduino` come progetto PlatformIO all'interno di VSCode
5. In `src/main.cpp`, verificare che sia presente:

```
#define PRODUCTION false
```
6. Dal **Command Palette** (F1) eseguire PlatformIO: Build. Questo genera i file di firmware in `.pio/build/uno/`.
7. Controllare il file `wokwi.toml`, deve essere del tipo:

```
[wokwi]
version = 1
firmware = "percorso del file firmware.hex"
elf       = "percorso del file firmware.elf"
```

8. Avviare il simulatore con il comando Wokwi: **Start Simulator** dal **Command Palette** di VSCode.
9. Wokwi caricherà automaticamente lo schema contenuto in `diagram.json` insieme al firmware appena compilato e una nuova finestra interattiva del simulatore verrà aperta.

6.3 Hardware Reale

1. In `src/main/main.ino`, assicurarsi che sia presente:

```
#define PRODUCTION true
```
2. Collegare il microcontrollore al PC
3. Aprire Arduino IDE
4. Selezionare la scheda **Arduino UNO** e la porta COM corretta
5. Iniettare il codice sulla scheda

7 Screenshot del Gioco

Di seguito sono riportati alcuni screenshot del gioco in esecuzione.

Durante il gioco il monitor seriale del microcontrollore mostra informazioni utili per il debug, come i codici IR ricevuti e le azioni eseguite.

Il simulatore offre anch'esso la possibilità di visualizzare il monitor seriale tramite il terminale.

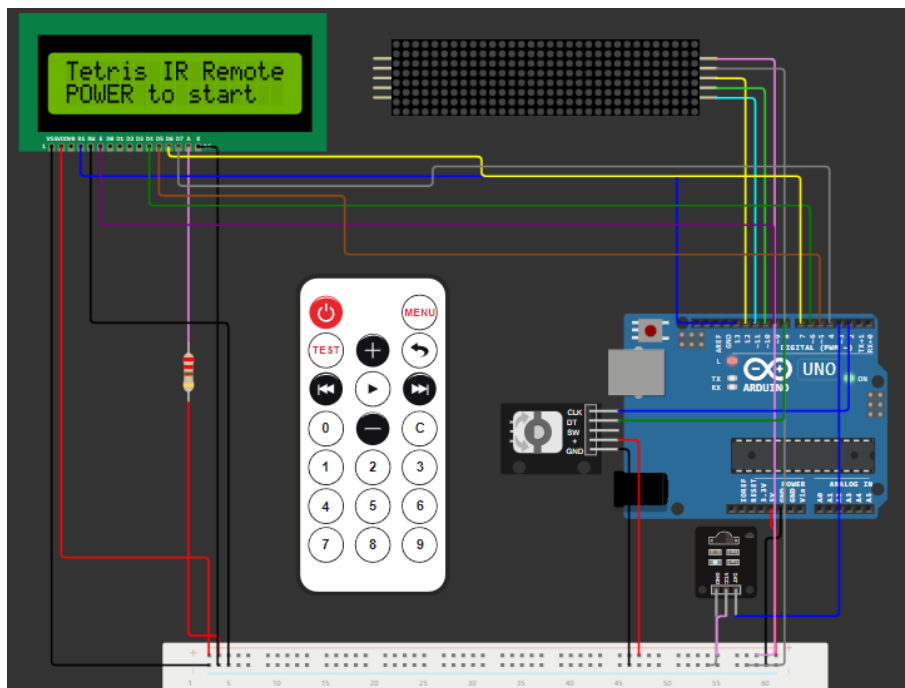


Figura 2: Screenshot del gioco appena acceso

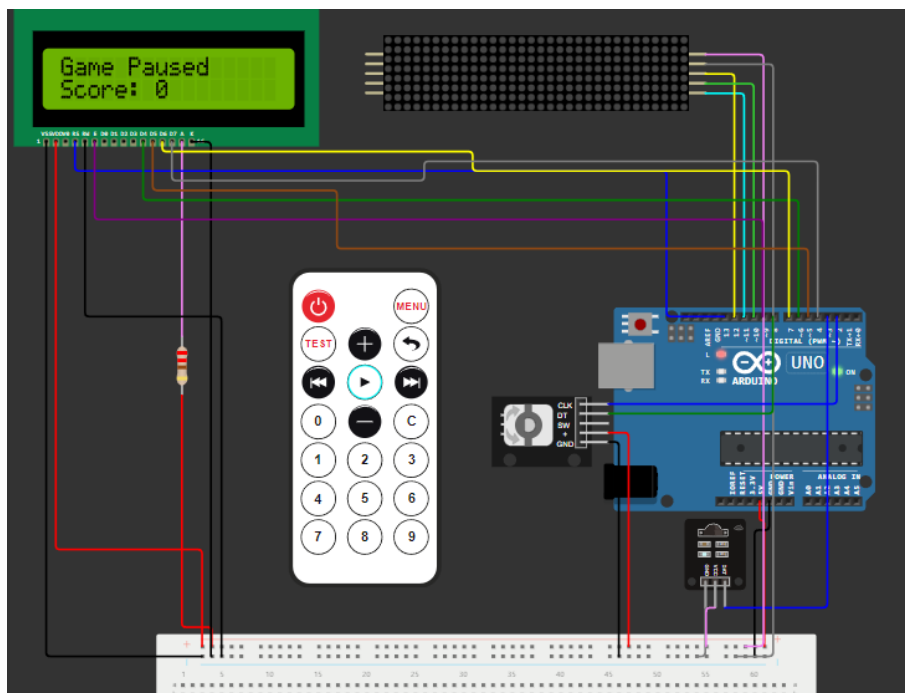


Figura 3: Screenshot del gioco in pausa

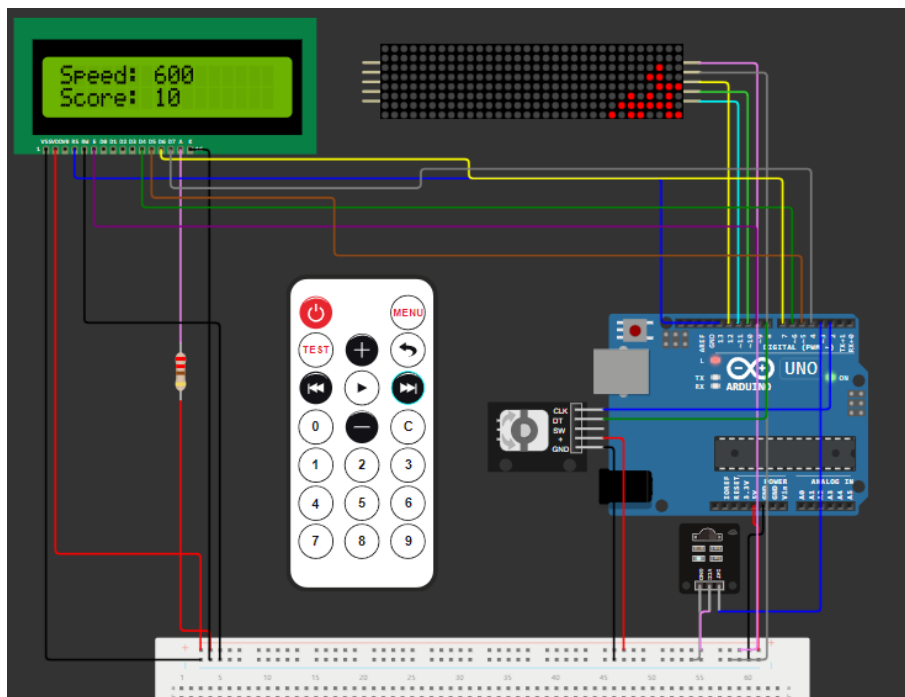


Figura 4: Screenshot del gioco in corso

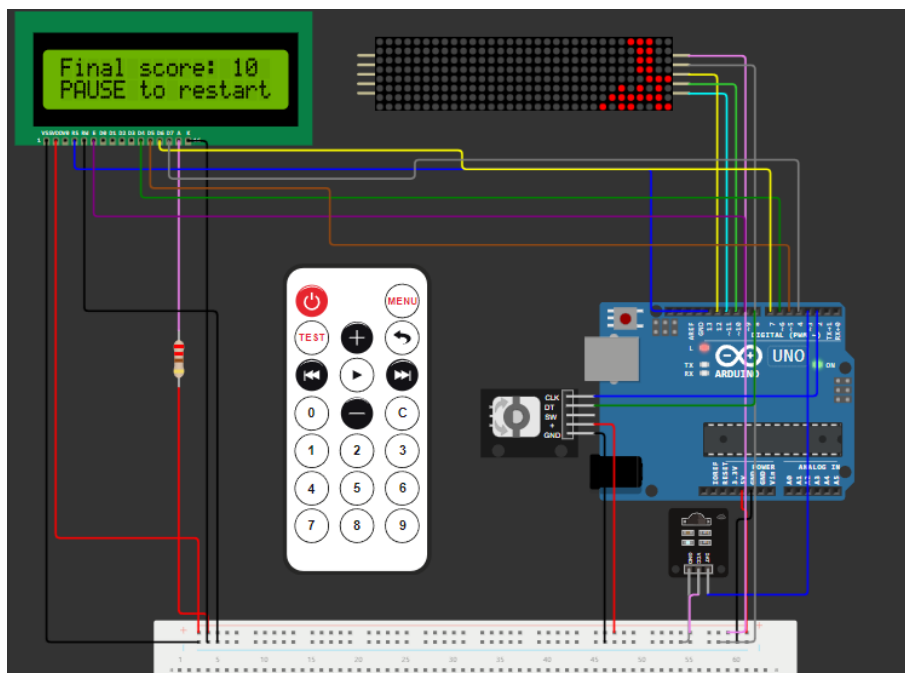


Figura 5: Screenshot del gioco terminato