UNIVERSITÀ DEGLI STUDI DI MILANO

Emulatore CHIP-8 su STM32

Presentazione del progetto di PROS,
corso tenuto da Danilo Bruschi

federico.bruzzone@studenti.unimi.it Federico Bruzzone
lorenzo.ferrante1@studenti.unimi.it Lorenzo Ferrante
andrea.longoni3@studenti.unimi.it Andrea Longoni

Scopo del progetto

- Realizzare un emulatore CHIP-8 e S-CHIP in grado di funzionare su un microcontrollore STM32

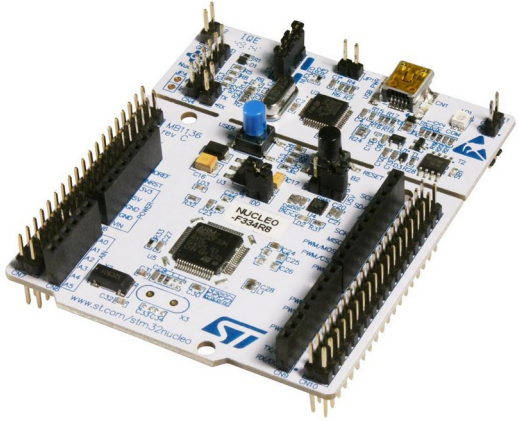
CHIP-8 (1)

- Linguaggio di programmazione
  - Sviluppo di videogiochi per microcomputer
- Macchina virtuale
  - Estensioni (S-CHIP e XO-CHIP)
- Videogiochi storici riscritti in CHIP-8
  - Pong, Space Invaders, Tetris

CHIP-8 (2)

- Stato dell'arte
  - Octojam – una gamejam annuale
  - Port su un elevato numero di piattaforme
    - Calcolatrici grafiche serie HP 48
    - Emacs (il famoso editor di testo)

Hardware (1)



Il microcontrollore *STM32F334R8T6*

- Processore ARM Cortex M4 da 72 MHz
- 16 Kb di SRAM
- 64 Kb di memoria flash

Hardware (2)

Lo schermo *ILI9341*

- Display TFT LCD a colori retroilluminato
- Dimensioni: 2.4 pollici
- Risoluzione: 320x240 px
- Lettore di schede microSD integrato

Hardware (3)

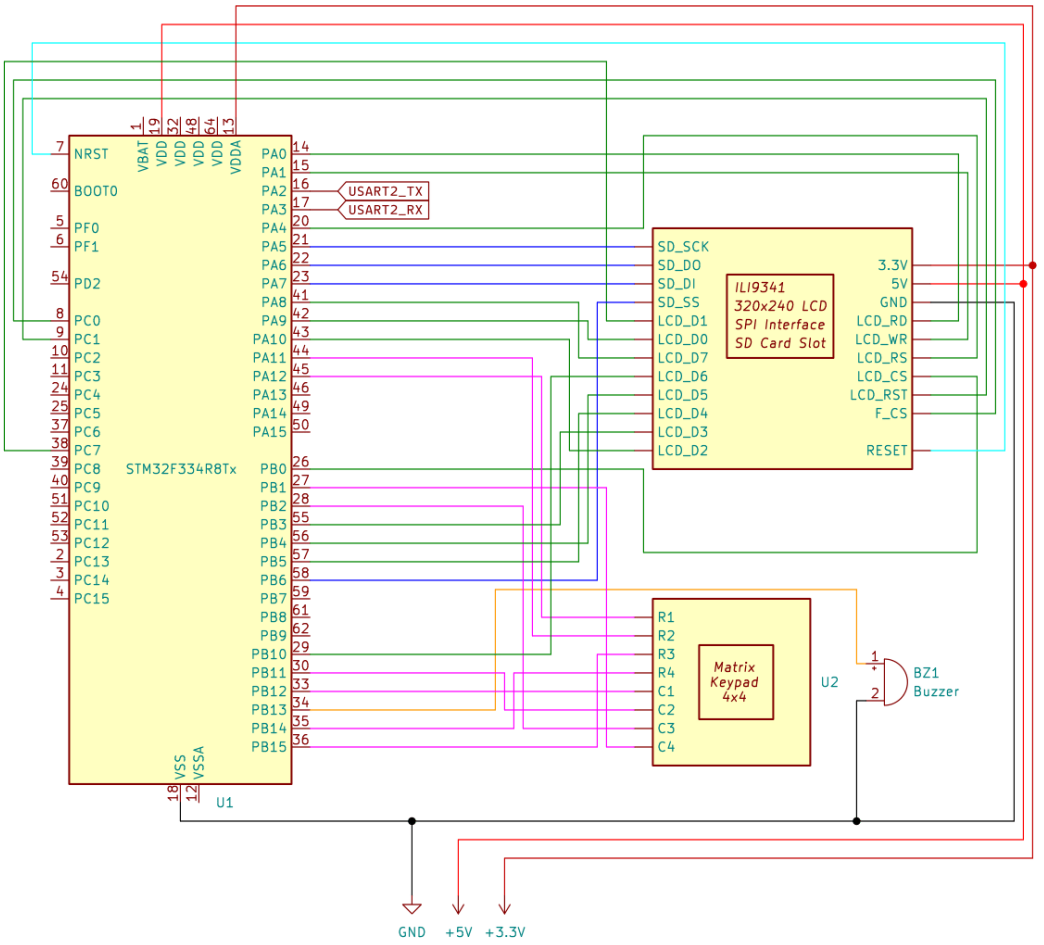
Il tastierino matriciale 4×4

Il beeper passivo monotono

Costi di realizzazione

Descrizione	**Modello**	**Costo unitario**	**Unità**	**Costo**
Microcontrollore	STM32 F334R8T6	14.99	1	14.99
Schermo	ILI9341 2.4"	6.50	1	6.50
Tastierino	Matrix keypad 4×4	3.99	1	3.99
Beeper		0.99	1	0.99
Cablaggio		4.99	1	4.99
Interruttore e altri materiali		4.99	1	4.99
Scocca	GW42002	9.99	1	9.99
			Totale	46.50€

Schema di collegamento (1)



Schema di collegamento (2)

- Display e lettore microSD
  - Pinout degli Shields di Arduino
- Tastierino matriciale 4x4
  - 8 pin GPIO
- Beeper
  - Pin GPIO e GND

Interprete CHIP-8 (1)

- Architettura basata su registri
  - 4 Kb di memoria
  - 16 registri general purpose
  - Un registro per gli indirizzi di memoria
  - Uno stack per le chiamate a subroutine
  - Uno stack pointer
  - Un program counter

Interprete CHIP-8 (2)

- L’interprete effettua il fetch dell’istruzione puntata dal PC in memoria, la decodifica e la esegue proprio come un processore reale
- Sono supportate 45 istruzioni diverse, ciascuna delle quali è rappresentata da un opcode contenente eventuali parametri

Interprete CHIP-8 (3)

- Implementazione altamente portabile
  - Scritta interamente in C99
  - Senza I/O
  - Freestanding (non dipende da libc)
- Testing
  - Port su desktop utilizzando SDL2
  - Test suite per verificare il comportamento corretto di ogni opcode

Gestione del timing

- Limitare la “velocità” dell’interprete bloccando temporaneamente la sua esecuzione
- Singola istruzione per ciclo di esecuzione
  - Ritardo variabile (dipende dalla “velocità” dell’interprete)
  - **Problema**: chiamata a funzione simil-sleep per un periodo troppo breve
- Multiple istruzioni per ciclo di esecuzione
  - Ritardo costante a ~16 ms → 60 FPS
  - N° di istruzioni eseguite dipende dalla “velocità” dell’interprete

Ottimizzazioni

- Lo schermo può essere visto come una matrice di 128x64 px monocromi
- **Problema**: questa rappresentazione occupa 8192 byte e abbiamo a disposizione solo 16 Kb di SRAM
- **Soluzione**: rappresentare lo schermo come un array unidimensionale di 1024 byte, dove ciascun pixel viene rappresentato da un singolo bit
- Otteniamo così un risparmio di spazio pari a ben l'87.5%
- **Livello di indirezione**: una coordinata sulla matrice deve essere mappata ad una coordinata in "memoria"

Comportamenti ambigui

- Gli emulatori CHIP-8 hanno sviluppato comportamenti ambigui nel corso degli anni
- Questi “quirk” variano in base alle piattaforme per cui è stato sviluppato l’interprete
- Per evitare la frammentazione bisogna supportare le piattaforme principali e i loro quirk
- Il nostro interprete supporta: CHIP-8, CHIP-48, S-CHIP 1.0, S-CHIP 1.1

Porting su STM32

- Interfaccia con lo schermo
- Interfaccia con la scheda microSD
- Gestore per il tastierino
- Gestore per il beeper

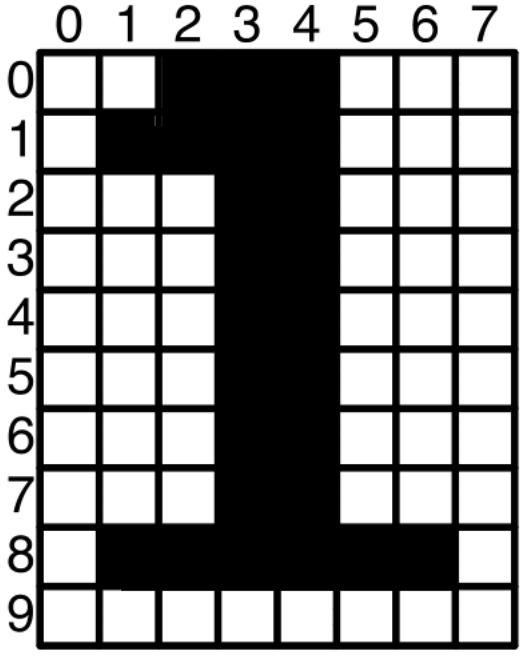
Interfaccia con la scheda microSD

- Scheda microSD
  - Memoria di massa dell'emulatore contente i giochi CHIP-8
  - Formattata con FAT32 e gestita utilizzando FatFS
- Comunicazione tramite *Serial Peripheral Interface* (SPI)
  - Microcontrollore → master
  - Scheda microSD → slave

Menù di selezione

- Lista su più pagine i giochi presenti sulla scheda microSD
- Viene generato dinamicamente
- Permette di impostare velocità e modalità di esecuzione

Rendering del font



- Font che rappresenta ogni carattere come una bitmap (matrice 8x10 px)
- **Esempio**: La prima riga è rappresentata dal numero 56, che in binario è 00111000. Ogni riga è un intero a 8 bit, dove i bit a 1 corrispondono ai pixel neri

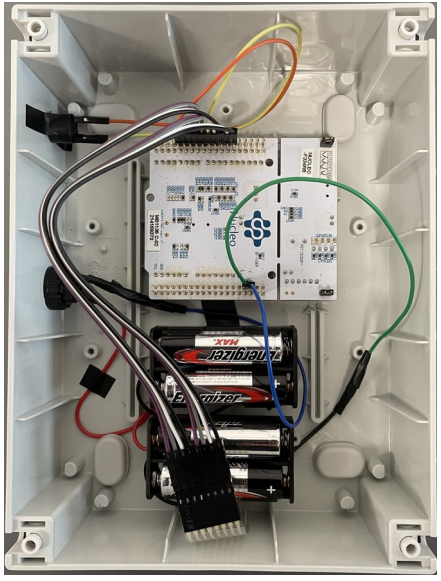
Funzionamento del keypad

- Gestione tramite polling
  - Richiede molte risorse
  - Non garantisce un frame rate stabile
- Gestione tramite interrupt
  - Pin GPIO come sorgente di interrupt
  - Funzione per la gestione di un interrupt
  - Pin di riga → *interrupt rising edge*
  - Pin di colonna → *push-pull output*

Interfaccia con lo schermo

- Il controller *ILI9341* supporta due interfacce:
  - SPI → richiede meno pin ma è più lenta
  - Parallela ad 8 bit → più veloce ma richiede più pin
- Ottimizzazione della funzione che stampa a schermo
  - **Problema**: l'HAL genera un overhead che rallenta l'emulatore
  - **Soluzione**: scrivere la funzione in *bare metal*

Assemblaggio



- Guscio protettivo (base di supporto polifunzionale)
- Fori sulla scocca per il beeper e l’interruttore
- Adesivi per il tastierino
- I collegamenti sono rimasti invariati

Consumo energetico

- Alimentazione USB vs Alimentazione esterna
  - Soluzione finale: 4 batterie AA da 1.5 v ciascuna
- Stima dell'autonomia dell'emulatore tramite STM32CubeIDE
  - Scheda: 30 mAh
  - Schermo: 90 mAh
  - $Autonomia = \frac{capacità}{consumo} = \frac{2500}{120} = 20$

Considerazioni finali

- Siamo riusciti a realizzare un emulatore CHIP-8 in grado di funzionare su un microcontrollore STM32
- Alcuni videogiochi non hanno un gameplay fluido a causa della potenza limitata della scheda
- **Ulteriore ottimizzazione**: interazione diretta tra l'interprete e il display

Riepilogo

- Componenti hardware
- Interprete CHIP-8
- Porting su STM32
- Assemblaggio
- Analisi del consumo energetico