X si 0

Nume: Vasile Vlad Raul

Grupa: 335CA

Introducere

Ce face?

Proiectul consta in realizarea unei table de joc X si 0 care ofera posibilitatea de a juca cu prietenii sau cu calculatorul. Aceasta ofera si un display care evalueaza miscarile fiecaruia si un buzzer care scoate sunete cand castigi sau pierzi.

Care este scopul lui?

Distractie singur sau cu prietenii!

Care a fost ideea de la care aţi pornit

Acesta este un joc pe care l-am jucat in continuu pe parcursul vietii, mai ales in scoala primara. Mi-ar placea sa am o versiune mai interactiva a acestui joc.

De ce credeţi că este util pentru alţii şi pentru voi?

In viata de la facultate, nu ai mereu un pix si o hartie la indemana. Proiectul meu rezolva aceasta problema.

Descriere generală

Creierul proiectului este un **ESP32 WEMOS LOLIN32** Lite. Acesta proceseaza mutarile jucate si calculeaza urmatoarea mutare optima.

Legatura intre jucator si tabla este facuta de o matrice 3×3 de **fotorezistori**, pe care daca se plaseaza piesa, placuta inregistreaza mutarea.

Pentru a-i comunica jucatorului unde vrea sa puna piesa, tabla lumineaza **ledul** din matricea 3×3 corespunzator patratelului unde doreste sa mute.

In plus, tabla ofera jucatorului si un **display** care ii arata iconite similare cu cele de pe chess.com care

evalueaza mutarile lui.

La finalul jocului, in functie de castig sau esec, un **buzzer** scoate un mic sunet.

Hardware Design

Componenta	Cantitate
ESP32 WEMOS Lolin32 Lite	1
Fotorezistori	9
Rezistente 10kΩ	9
Display OLED 0.96” Alb I2C cu 4 butoane	1
Banda LED WS2812	1 segment (9 leduri)
Buzzer pasiv	1
Rezistenta 100Ω	1

Placuta ESP32 WEMOS Lolin32 Lite are un cip ESP32-D0WDQ6.

Pentru fiecare fotorezistor am folosit o rezistenta de 10kΩ pentru a crea un divizor de tensiune.

Buzzer-ul este conectat la pin-ul placutei printr-o rezistenta de 100Ω care limiteaza curentul.

Software Design

Descrierea codului aplicaţiei (firmware):

- mediu de dezvoltare (if any) (e.g. AVR Studio, CodeVisionAVR)
- librării şi surse 3rd-party (e.g. Procyon AVRlib)
- algoritmi şi structuri pe care plănuiţi să le implementaţi
- (etapa 3) surse şi funcţii implementate

Mediu de dezvoltare: PlatformIO, Framework: espidf

Librarii

Header	Rol
`driver/spi_master.h`	Comunicare SPI cu banda LED WS2812
`driver/i2c_master.h`	Comunicare I2C cu display-ul OLED
`driver/ledc.h`	Generare PWM pentru buzzer pasiv

`esp_adc/adc_oneshot.h`	Citire valori de la fotorezistori prin ADC
`freertos/task.h`	Task-uri si delay-uri non-blocante
`driver/gpio.h`	Configurare si citire butoane

Implementari manuale

1. Driver WS2812 prin SPI

Emulez protocolul benzii de leduri (1 = HIGH pentru 800ns, 0 = HIGH pentru 400ns) prin SPI la 2.5 MHz. Fiecare bit al unui pixel este codificat ca 3 biti astfel:

- 1 logic → `0b110`
- 0 logic → `0b100`

Functii implementate:

- `init_native_ws2812()` - configureaza frecventa la 2.5 MHz si linia MOSI pe pinul bun
- `init_native_ws2812(idx, r, g, b)` - scrie in buffer datele pentru un led
- `clear_leds()` - seteaza toate ledurile pe 0
- `show_leds()` - trimite datele catre banda de leduri

2. Driver OLED SSD1306 prin I2C

Display-ul este controlat prin protocolul I2C. Memoria acestuia este reprezentata de un array de 128 * 8 octeti. Fiecare octet controleaza 8 pixeli.

Functii implementate:

- `oled_send_cmd(cmd)` - trimite comenzi display-ului
- `init_i2c_and_oled()` - initializeaza display-ul
- `oled_update()` - repozitioneaza cursorul si trimite bufferul spre afisare
- `oled_clear()` - stinge toti pixelii
- `oled_draw_pixel(x, y)` — seteaza un bit în framebuffer
- `oled_draw_line(x0, y0, x1, y1)` — algoritmul **Bresenham** pentru linii
- `oled_draw_O(x, y, r)` — deseneaza un cerc
- `oled_draw_X(cx, cy, size)` — doua linii diagonale
- `oled_draw_char` / `oled_draw_string` — randare text font 5×7
- `oled_draw_char_2x` / `oled_draw_string_2x` — scalare 2x a fontului

3. Buzzer prin PWM

Buzzer-ul necesita un semnal PWM la frecventa dorita. Am folosit perifericul LEDC pentru a genera acest semnal.

Functii implementate:

- `init_buzzer()` - initialieaza buzzer-ul cu duty cycle pe 13 biti
- `play_tone(freq, duration)` - seteaza frecventa si porneste sunetul
- `play_win_sound()` - 4 note care reprezinta sunetul de castig
- `play_lose_sound()` - 4 note care sunt sunetul de esec

Exista si sunet de remiza, care este reprezentat printr-o singura nota, asa ca nu am facut o functie separata pentru el. In cazul in care sunt 2 jucatori, la finalul jocului se pune mereu suntetul de castig sau de remiza.

Implementarea

Logica jocului este implementata ca o masina de 6 stari, care ruleaza intr-un task FreeRTOS.

STATE_CALIBRATION

Aceasta stare ruleaza o singura data la pornire. Se citeste valoarea de la fiecare senzor de 10 ori si se face media pentru a calcula pragul de detectie.

STATE_MENU

In aceasta stare se selecteaza modul de joc. Se poate alege intre jucatul cu calculatorul sau cu alt jucator (butonul K1) si cine incepe (butonul K2), in cazul jucatului cu calculatorul. Butonul K3 porneste jocul.

STATE_PLAYER_TURN

La fiecare 50ms, placuta citeste valorile de la senzori si verifica daca vreuna depaseste pragul de detectie. La prima detectie, inregistreaza mutarea o evalueaza pentru a afisa iconita pe display si verifica daca jocul s-a terminat.

STATE_COMPUTER_TURN

Se ruleaza algoritmul Minimax si returneaza mutarea optima. Apoi trece in starea STATE_WAIT_FOR_PIECE.

STATE_WAIT_FOR_PIECE

Calculatorul comunica cu jucatorul pozitia unde vrea sa ii fie plasata piesa prin aprinderea ledului corespunzator. Apoi, acesta asteapta sa i se puna piesa unde trebuie. Nu inregistreaza mutari invalide.

STATE_GAME_OVER

In aceasta stare, buzzer-ul scoate un sunet in functie de rezultatul jocului. Banda de leduri aprinde ledurile de sub formatia castigatoare, daca este cazul. Daca s-a facut remiza, se aprind toate ledurile.

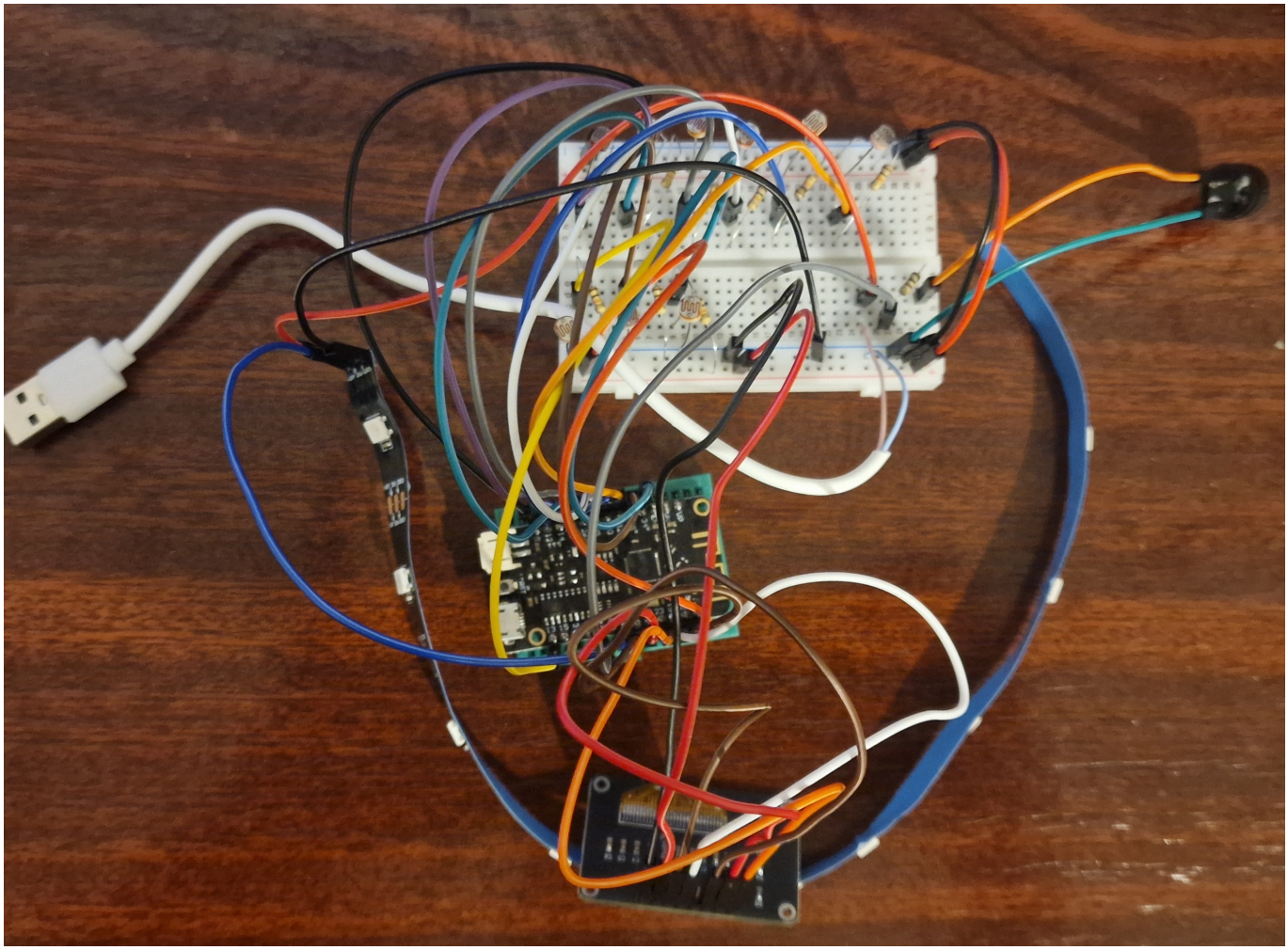
Algoritmul Minimax

Calculatorul foloseste algoritmul Minimax complet, care exploreaza tot arborele de stari al jocului. Acesta este rulat la fiecare mutare si ia in calcul adancimea arborelui ca sa prioritizeze castiguri rapide.

Iconitele de evaluare

Iconita	Semnificatie
! !	Mutare castigatoare
!	Mutare care forteaza castig
	Mutare care a blocat castigul adversarului
??	Blunder — adversarul castigă imediat
?	Mutare slaba — adversarul poate forta castig
	Mutare neutra

Rezultate Obţinute



Download

GitHub

Bibliografie/Resurse

Listă cu documente, datasheet-uri, resurse Internet folosite, eventual grupate pe **Resurse Software** şi **Resurse Hardware**.

ESP32 Datasheet - Placuta

WEMOS Lolin32 Lite Pinout

Informatii display

SSD1306 Datasheet - Display

WS2812 Datasheet - Banda de leduri