Pong Game

Ciupitu Andi Mihai

331CD

Introducere

Prezentarea pe scurt a proiectului vostru:

- Joc clasic Pong: poate fi jucat de 2 persoane fiecare avand cate un joystick pentru manevrarea paletei sus si jos, toata actiunea este afisata pe display iar buzzer ul va scoate sunete in functie de coliziunea dintre minge si paleta sau perete(punct pentru adversar).

Descriere generală

!POTENTIOMETRELE din schema reprezinta MODULE JOYSTICK! Pini folositi:

Pinul GND este folosit pentru a conecta masa la breadboard pentru a completa circuitul.

Pinul 5V furnizează 5V către breadboard asigurând alimentarea necesară.

Pinul 12 al placii Arduino fiind un pin PWM(Pulse Width Modulation) permite controlul volumului conectat la pinul + buzzerului.

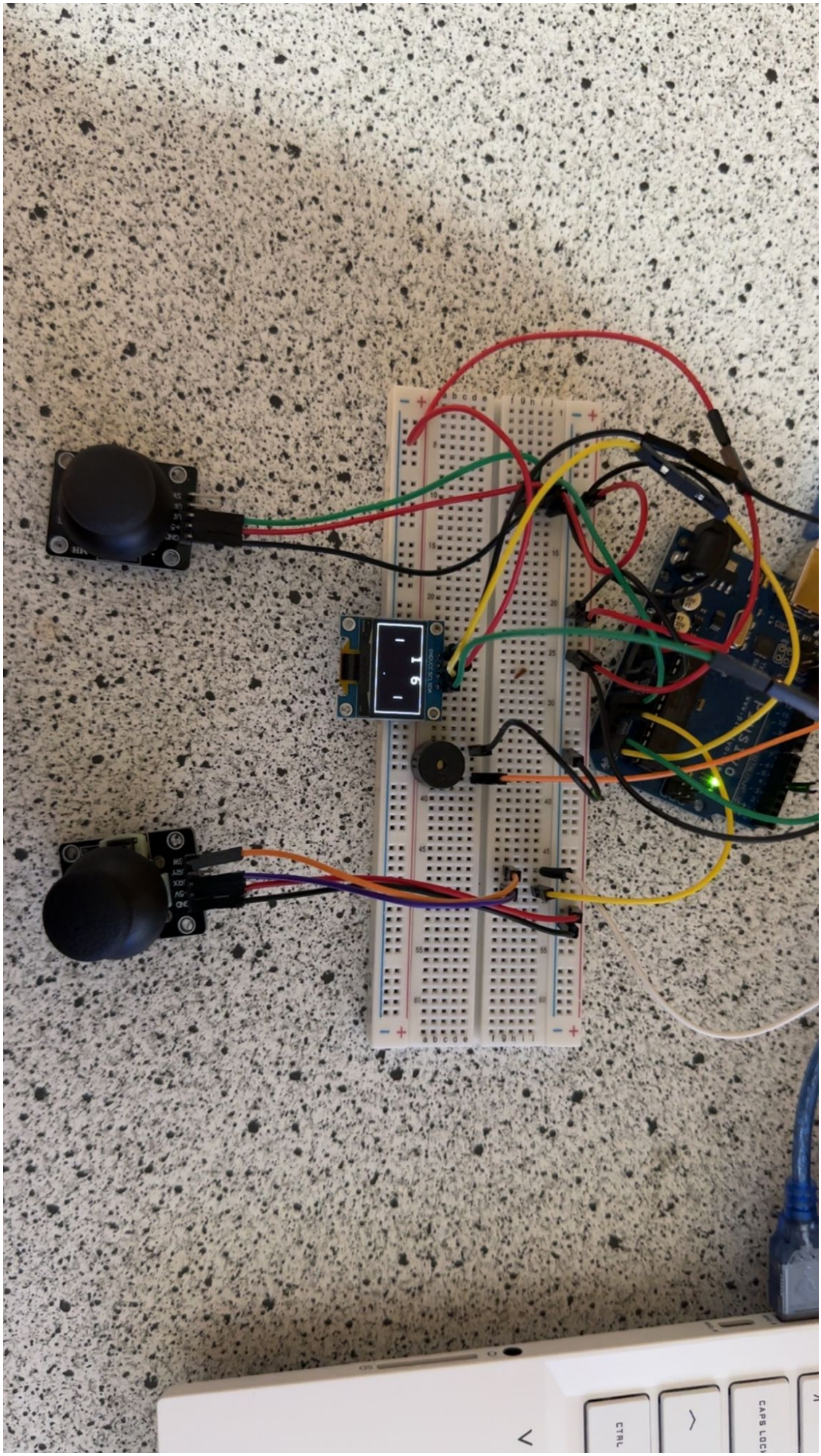
Conectarea pinilor A4 și A5 la SDA și SCL permite transferul eficient de date între placa Arduino și dispozitivul I2C.

Hardware Design

Lista de piese necesara:

- Ecran LCD (afiseaza dinamica jocului)
- 2 joysticks (permite controlul paletelor pentru lovirea mingii din joc)
- fire (conectivitate)
- Breadboard (placa de circuit pentru conectivitatea pieselor intre ele)
- Arduino UNO (realizeaza si executa logica proiectului fiind un microcontroler programabil)
- Buzzer (produce sunet pentru a oferi feedback utilizatorului)

Stadiul actual al implementarii hardware:



Software Design

Librării şi surse 3rd-party:

#include <Wire.h> este utilizată pentru a configura și gestiona comunicația I2C pe Arduino.

#include <Adafruit_GFX.h> este utilizată pentru a simplifica desenarea și manipularea graficelor pe afișajul OLED.

#include <Adafruit_SSD1306.h> este utilizată pentru a gestiona direct afișajul OLED SSD1306. Aceasta se bazează pe Adafruit_GFX.h pentru a oferi funcții grafice, dar adaugă funcționalități specifice pentru controlul afișajului OLED.

Laboratoare utilizate

1. Laboratorul 3: Timere, Pulse Width Modulation (PWM) Functia tone() utilizează PWM pentru a genera un semnal audio pe pinul specificat (BEEPER). Utilizarea funcției millis() este o metodă comună pentru temporizare.
2. Laborator 4: Analog Digital Convertor (ADC) Am folosit analogRead() pentru monitorizarea pozitiilor paletelor prin joystickuri, aceasta functie utilizeaza ADC pentru a converti tensiunea analogică de pe pinii analogici în valori digitale.
3. Laboratorul 6: I2C (Inter-Integrated Circuit) protocolul I2C (Inter-Integrated Circuit) a fost folosit pentru a comunica cu ecranul OLED (Adafruit_SSD1306) conectat.

Algoritmi si functii implementate

setup() este apelată o singură dată, la începutul programului, pentru a inițializa componentele hardware și a pregăti mediul de execuție. {

```
// Initialize the display
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
display.display();
unsigned long start = millis();

// Initialize pins
pinMode(BEEPER, OUTPUT);
pinMode(SW_pin, INPUT);
pinMode(RESET_BUTTON, INPUT_PULLUP);
digitalWrite(SW_pin, HIGH);

// Clear display and draw initial court and score
```

```
display.clearDisplay();
drawCourt();
drawScore();

// Display initial screen for 2 seconds
while (millis() - start < 2000);
display.display();
```

```
// Set initial update times
ball_update = millis();
paddle_update = ball_update;
```

}

loop() rulează continuu și conține logica principală a jocului. { bool update = false;

```
unsigned long time = millis();
```

```
// Check if ball needs to be reset
if (resetBall) {
  if (player1Score == maxScore || player2Score == maxScore) {
    gameOver();
  } else {
    resetGame();
  }
}
```

```
// Update ball position
if (time > ball_update) {
  updateBallPosition();
  update = true;
}
```

```
// Update paddle positions
if (time > paddle_update) {
  updatePaddlePositions();
  update = true;
}
```

```
// Update display if necessary
if (update) {
  drawScore();
  display.display();
}
```

}

resetGame() resetează poziția și direcția mingii și re-desenază terenul și scorul. {

```
display.fillScreen(BLACK);
drawScore();
```

```
drawCourt();
ball_x = random(45, 50);
ball_y = random(23, 33);
do {
  ball_dir_x = random(-1, 2);
} while (ball_dir_x == 0);
```

```
do {
  ball_dir_y = random(-1, 2);
} while (ball_dir_y == 0);
```

```
resetBall = false;
```

```
}
```

updateBallPosition() actualizează poziția mingii și gestionează coliziunile. {

```
uint8_t new_x = ball_x + ball_dir_x;
uint8_t new_y = ball_y + ball_dir_y;
```

```
// Check for vertical wall collisions
if (new_x == 0 || new_x == 127) {
  handleVerticalWallCollision(new_x);
}
```

```
// Check for horizontal wall collisions
if (new_y == 0 || new_y == 63) {
  soundBounce();
  ball_dir_y = -ball_dir_y;
  new_y += ball_dir_y + ball_dir_y;
}
```

```
// Check for paddle collisions
if ((new_x == PLAYER2_X && new_y >= player2_y && new_y <= player2_y +
PADDLE_HEIGHT) ||
    (new_x == PLAYER_X && new_y >= player1_y && new_y <= player1_y +
PADDLE_HEIGHT)) {
  soundBounce();
  ball_dir_x = -ball_dir_x;
  new_x += ball_dir_x + ball_dir_x;
}
```

```
// Draw ball in new position
display.drawPixel(ball_x, ball_y, BLACK);
display.drawPixel(new_x, new_y, WHITE);
ball_x = new_x;
ball_y = new_y;
```

```
ball_update += BALL_RATE;
```

}

void handleVerticalWallCollision(uint8_t new_x) gestionează coliziunile mingii cu marginile verticale ale terenului. {

```
if (new_x == 0) {
  player1Score += 1;
  display.fillScreen(BLACK);
  soundPoint();
  resetBall = true;
} else if (new_x == 127) {
  player2Score += 1;
  display.fillScreen(BLACK);
  soundPoint();
  resetBall = true;
}
ball_dir_x = -ball_dir_x;
new_x += ball_dir_x + ball_dir_x;
```

}

void updatePaddlePositions() actualizează pozițiile paletelor în funcție de intrările analogice. {

```
paddle_update += PADDLE_RATE;
```

```
// Update Player 2 paddle
display.drawFastVLine(PLAYER2_X, player2_y, PADDLE_HEIGHT, BLACK);
if (analogRead(player2) < 475) {
  player2_y -= 2;
}
if (analogRead(player2) > 550) {
  player2_y += 2;
}
constrainPaddlePosition(player2_y);
display.drawFastVLine(PLAYER2_X, player2_y, PADDLE_HEIGHT, WHITE);
```

```
// Update Player 1 paddle
display.drawFastVLine(PLAYER_X, player1_y, PADDLE_HEIGHT, BLACK);
if (analogRead(player1) < 475) {
  player1_y -= 2;
}
if (analogRead(player1) > 550) {
  player1_y += 2;
}
constrainPaddlePosition(player1_y);
display.drawFastVLine(PLAYER_X, player1_y, PADDLE_HEIGHT, WHITE);
```

}

void constrainPaddlePosition(uint8_t &paddle_y) constrânge poziția paletei pentru a nu ieși în afara terenului de joc. {

```
if (paddle_y < 1) paddle_y = 1;
if (paddle_y + PADDLE_HEIGHT > 63) paddle_y = 63 - PADDLE_HEIGHT;
```

}

void drawCourt() desenează un dreptunghi care reprezintă marginea terenului de joc. {

```
display.drawRect(0, 0, 128, 64, WHITE);
```

}

void drawScore() afiseaza scorul jucătorilor pe ecran. {

```
display.setTextSize(2);
display.setTextColor(WHITE);
display.setCursor(45, 0);
display.println(player2Score);
display.setCursor(75, 0);
display.println(player1Score);
```

}

void gameOver() afișează mesajul de sfârșit de joc și resetează scorurile. {

```
display.fillScreen(BLACK);
display.setCursor(20, 15);
display.setTextColor(WHITE);
display.setTextSize(2);
if (player1Score > player2Score) {
  display.print("Player 1");
} else {
  display.print("Player 2");
}
display.setCursor(40, 35);
display.print("won");

delay(100);
display.display();
delay(2000);
player2Score = player1Score = 0;
```

```
unsigned long start = millis();
while (millis() - start < 2000);
ball_update = millis();
paddle_update = ball_update;
resetBall = true;
```

}

void soundBounce() emite un sunet scurt atunci când mingea lovește ceva. {

```
tone(BEEPER, 500, 50);
```

}

void soundPoint() emite un sunet scurt atunci când se marchează un punct. {

```
tone(BEEPER, 100, 50);
```

}

Video:https://youtu.be/PE-iA88xzM8

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

1. **Funcționalitate completă a jocului**: Proiectul finalizează implementarea unui joc de ping-pong pe un afișaj OLED controlat de un microcontroller Arduino. Jucătorii pot controla paletele, mingea se mișcă și colisionează corect cu marginile și paletele, iar scorul este actualizat și afișat corect.
2. **Interacțiune cu utilizatorul**: Joystickurile permit controlul efectiv al paletelor de către jucători. Sunetele emise la coliziuni și puncte adaugă un nivel de feedback auditiv.
3. **Afișaj clar și precis**: Folosirea bibliotecilor Adafruit pentru grafica OLED asigură un afișaj clar și precis al jocului, inclusiv grafică simplă pentru teren, palete, minge și scor.

Concluzii

Proiectul implică conectarea și utilizarea diferitelor componente hardware, inclusiv afișaj OLED, joysticks și un buzzer. Aceasta oferă o înțelegere practică a modulului de funcționare și interfațare a acestor componente.

Proiectul a rezultat joc simplu dar captivant, care poate fi jucat de două persoane, oferind ore de distracție.

Download

ciupitu_andi_mihai_331cd.zip

Jurnal

Puteți avea și o secțiune de jurnal în care să poată urmări asistentul de proiect progresul proiectului.

Bibliografie/Resurse

Export to PDF

From:
http://ocw.cs.pub.ro/courses/ - **CS Open CourseWare**

Permanent link:
http://ocw.cs.pub.ro/courses/pm/prj2024/avaduva/andi_mihai.ciupitu

Last update: **2024/05/27 02:21**