Simulator de Dezamorsare

Introducere

Inspirat de jocul “Keep Talking and Nobody Explodes”, proiectul propus este un joc de simulare a dezamorsării unei bombe. Scopul acestui proiect este de a oferi o experiență captivantă și educativă utilizatorilor, permițându-le să simuleze procesul de dezamorsare într-un mod sigur și interactiv.

Acest proiect este conceput pentru a aduce o contribuție în domeniul divertismentului digital.

Descriere generală

Pentru simularea dezamorsării, e nevoie de un microcontroler și mai multe componente hardware pentru a crea o experiență interactivă.

Componente hardware:

1. **Microcontroller ESP32:** Responsabil pentru controlul și gestionarea tuturor componentelor și interacțiunilor din cadrul jocului.

2. **Display:** Folosit pentru afișarea informațiilor importante, cum ar fi numărul de secunde rămase până la detonare și diverse instrucțiuni pentru jocurile de dezamorsare.

3. **Buzzer:** Este utilizat pentru a emite semnale sonore, cum ar fi bipuri, pentru a indica jucătorului cât timp mai are disponibil și pentru a crea o atmosferă tensionată.

4. **Senzor de temperatură:** Acest senzor monitorizează temperatura ambientală și poate fi integrat într-unul dintre jocurile de dezamorsare pentru a adăuga o componentă de detectare a temperaturii.

5. **Senzor giroscopic:** Acest senzor măsoară orientarea și accelerația și poate fi utilizat pentru unul dintre jocurile de dezamorsare care implică mișcări specifice ale dispozitivului.

6. **Butoane:** Acestea sunt folosite pentru a interacționa cu jocurile de dezamorsare și pentru a introduce răspunsuri sau comenzi în sistem.

Hardware Design

Lista de piese:

- Adafruit HUZZAH32 cu ESP32
- Senzor de temperatura ASAIR AM2302
- LCD 1602 cu Interfata I2C
- Modul Accelerometru și Giroscop cu 3 Axe MPU6050
- Buzzer Pasiv 3.3V
- Modul accelerometru si giroscop GY521
- Modul Display OLED SPI SSD1306
- Butoane
- LED-uri
- breadboard

Software Design

Proiectul include un joc de labirint și un joc cu LED-uri, ambele controlate prin butoane, precum și un timer de bombă cu funcționalitate de alertă sonoră. Codul folosește diverse biblioteci pentru a facilita interfațarea cu hardware-ul și implementează algoritmi de debounce pentru citirea butoanelor, amestecarea secvențelor de butoane și LED-uri și desenarea grafică a labirintului pe afișajul OLED. Aplicația gestionează starea jocului și a bombei, oferind feedback utilizatorului prin intermediul afișajelor și al sunetelor buzzer-ului.

- Mediu de dezvoltare folosit: Arduino IDE
- Librării şi surse 3rd-party:

```
#include <Wire.h> // Biblioteca pentru comunicare I2C.
```

```
#include <MPU6050.h> // Biblioteca pentru interfațarea cu senzorul MPU6050.
```

```
#include <Adafruit_GFX.h> // Biblioteca grafică pentru afișaje.
```

```
#include <Adafruit_SSD1306.h> // Biblioteca pentru afișajul OLED SSD1306.
```

```
#include <LiquidCrystal_I2C.h> // Biblioteca pentru afișajul LCD 1602 cu
modul I2C.
```

- Funcții implementate

```
void setup() // Inițializează sistemul, afișajele și senzorii
```

```
void loop() // Rulează în mod continuu pentru a verifica starea butoanelor,
actualizează poziția jucătorului, gestionează starea bombei și labirintului
```

```
void displaySetup() // Configurează afișajul OLED și inițializează pinurile
butoanelor
```

```
void updatePlayerPosition() // Actualizează poziția jucătorului pe baza
intrărilor de la butoane
```

```
void drawMaze(int mazeIndex) // Desenează labirintul pe afișajul OLED
```

```
void buzzerSetup() // Configurează pinul pentru buzzer
```

```
void bombTick() // Gestionează sunetul buzzer-ului pentru tick-ul bombei
```

```
void explosionSound() // Redă sunetul exploziei
```

```
void playGameWinningSound() // Redă sunetul de câștig
```

```
void timeTick() // Actualizează timpul rămas și afișează pe LCD
```

```
void gyroSetup() // Inițializează senzorul MPU6050 și afișajul LCD
```

```
void gyroReadings() // Citește valorile de la senzorul MPU6050 și detectează mișcările bruște
```

```
void ledGameSetup() // Configurează jocul LED
```

```
void ledGameTick() // Gestionează starea jocului LED
```

```
void lose() // Funcția apelată când explozia are loc
```

```
void displayOkSign() // Desenează un semn "OK" pe afișajul OLED
```

```
void win() // Funcția apelată când jocul este câștigat
```

Rezultate Obţinute

Cu toate că proiectul final a avut o formă diferită față de cea inițial imaginată (unele piese fiind înlocuite sau eliminate complet, iar altele noi fiind adăugate pe parcurs), am reușit să obțin un joc destul de interesant de care sunt mulțumit.

Concluzii

În concluzie, mi-a plăcut să lucrez la acest proiect. Am întâmpinat multe probleme neașteptate, dar acestea m-au ajutat să învăț foarte multe, atât pe partea de software, cât și pe cea de hardware.

Download

ioanpopescu_proiect_pm.zip

Jurnal

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

Bibliografie/Resurse

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

Resurse Hardware:

- https://docs.sunfounder.com/projects/esp32-starter-kit/en/latest/_images/esp32_pinout.jpg
- http://ucapps.de/midibox_ng_manual_lcd.html

Resurse Software:

- https://edukits.co/lesson/noise-bomb/
- https://www.electronicshub.org/esp32-oled-display/
- https://lastminuteengineers.com/esp32-i2c-lcd-tutorial/?utm_content=cmp-true

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

Permanent link:
https://ocw.cs.pub.ro/courses/pm/prj2024/sseverin/ioan.popescu1910

Last update: **2024/05/27 03:23**