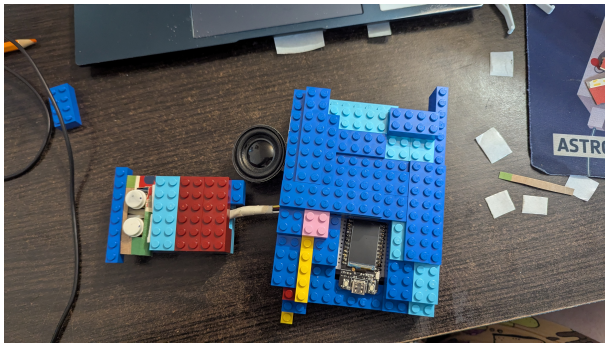
Mini Consola de jocuri prin Bluetooth



Introducere

BlueBomb Console este o miniconsola ce pune la dispozitie jocul de MineSweeper multiplayer prin Bluetooth. Cu ajutorul unei aplicatii Android, jucatorii se pot conecta la consola prin intermediul Bluetooth si se pot duela pentru titlul de cel mai bun <insert titulatura pozitiva>.

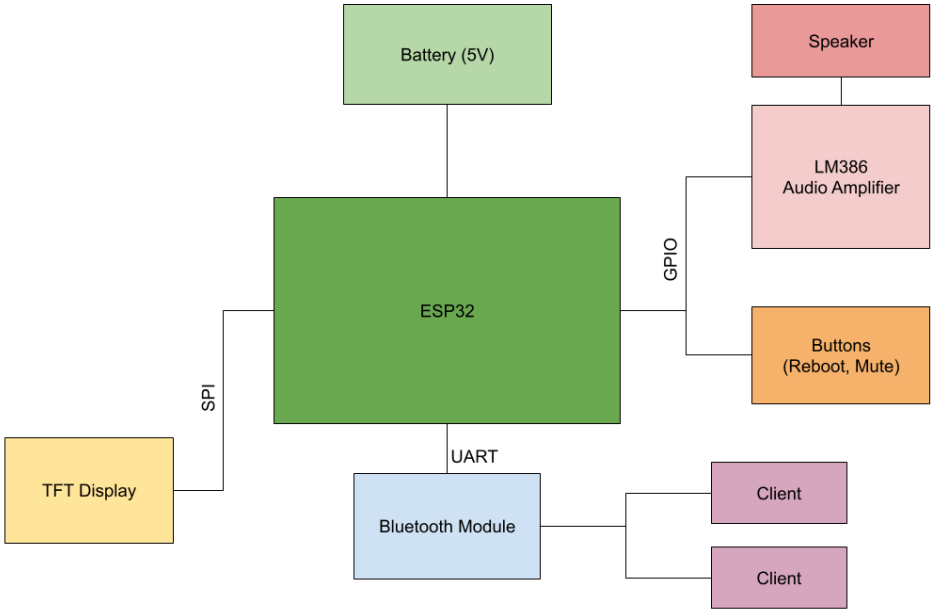
Scopul proiectului este de a crea un portable entertainment device, care sa nu depinda de conexiunea la reteaua de WiFi.

Ideea Generala a proiectului consta in proiectarea unui device ce va putea fi folosit atat pentru un joc single player de Minesweeper, cat si a unuia multiplayer. Device-ul este echipat cu un ecran LCD pe care va fi proiectat atat jocul cat si mesaje pentru utilizatori si un minispeaker pentru a crea o experienta unica de joc.

Descriere generală

O schemă bloc cu toate modulele proiectului vostru, atât software cât şi hardware însoţită de o descriere a acestora precum şi a modului în care interacţionează.

Exemplu de schemă bloc: http://www.robs-projects.com/mp3proj/newplayer.html



Componente folosite:

Nume Componenta	Descriere	Datasheet
TTGO T-Display	Placa de dezvoltare integrata	
ESPRESSIF-ESP32 240MHz Xtensa*	Microcontroler care gestioneaza interactiunea cu clientii si da comenzi catre celelalte componente	Datasheet
ST7789V IPS LCD*	Display	Datasheet
LM386	Audio Power Amplifier	Datasheet
Butoane		
Fire de legatura		

* - Componentele marcate sunt integrate in placa de dezvoltare TTGO T-Display. Datasheet-ul este util pentru urmarirea implementarii functionalitatilor pe aceste componente. }

Hardware Design

Software Design

Arhitectura software

Arhitectura software este împărțită în următoarele componente principale:

* **Logica de joc** - implementată în clasa `Minesweeper`

* **Comunicație Bluetooth** - gestionată prin BLE

* **Interfață grafică** - utilizând biblioteca TFT_eSPI

* **Gestionarea mesajelor** - coadă de mesaje pentru procesare asincronă

* **Handlere de evenimente** - pentru întreruperi hardware și evenimente Bluetooth

Structura codului

Fișierul main.cpp

Acest fișier conține programul principal și coordonează celelalte componente.

Includerea bibliotecilor necesare:

```
#include <Arduino.h>   // functii de baza folosite pentru simplitate

#include <BLEDevice.h>
#include <BLEServer.h> // Pentru folosirea modulului esp32 pe post de server
#include <BLE2902.h>

#include <TFT_eSPI.h>  // functionalitati display

#include "minesweeper.h"
#include "bt_commands.h" // Traduce comenzile de la aplicatia de pe
smartphone
```

Inițializarea obiectelor principale:

```
TFT_eSPI tft = TFT_eSPI(); // Obiect pentru controlul ecranului TFT

Minesweeper game;          // Obiect pentru logica jocului
```

Coada de mesaje pentru procesarea asincronă:

```
#define MAX_MESSAGES 10
#define MAX_MESSAGE_LENGTH 100

typedef struct
{
  uint16_t length;
  uint8_t data[MAX_MESSAGE_LENGTH];
} message_t;

message_t messageQueue[MAX_MESSAGES];
int messageQueueHead = 0;
```

```
int messageQueueTail = 0;

portMUX_TYPE messageQueueMux = portMUX_INITIALIZER_UNLOCKED;
```

Funcții pentru gestionarea cozii de mesaje:

Aceasta permite o experienta fluida in cadrul jocului, mesajele fiind procesate atunci cand microcontrollerul are timp pentru ele.

```
bool addMessageToQueue(uint8_t *data, size_t length)
{
  if (length > MAX_MESSAGE_LENGTH)
  {
    length = MAX_MESSAGE_LENGTH; // Truncate if too long
  }
  portENTER_CRITICAL(&messageQueueMux);

  int nextHead = (messageQueueHead + 1) % MAX_MESSAGES;

  if (nextHead == messageQueueTail)
  {
    // Queue is full
    portEXIT_CRITICAL(&messageQueueMux);
    return false;
  }

  messageQueue[messageQueueHead].length = length;
  memcpy(messageQueue[messageQueueHead].data, data, length);
  messageQueueHead = nextHead;
  portEXIT_CRITICAL(&messageQueueMux);
  return true;
}

bool getMessageFromQueue(message_t *message)
{
  portENTER_CRITICAL(&messageQueueMux);
  if (messageQueueHead == messageQueueTail)
  {
    // Queue is empty
    portEXIT_CRITICAL(&messageQueueMux);
    return false;
  }

  *message = messageQueue[messageQueueTail];
  messageQueueTail = (messageQueueTail + 1) % MAX_MESSAGES;
  portEXIT_CRITICAL(&messageQueueMux);
  return true;
}
```

Callback pentru evenimente Bluetooth:

```
// Server & Characteristic callbacks

class MyServerCallbacks : public BLEServerCallbacks
{
   // onConnect, onDisconnect
}

BLEServer *pServer = NULL;
BLECharacteristic *pCharacteristic = NULL;
class MyCallbacks : public BLECharacteristicCallbacks
{
  void onWrite(BLECharacteristic *pCharacteristic, esp_ble_gatts_cb_param_t
*param)
  {
    std::string value = pCharacteristic->getValue();
    if (value.length() > 0)
    {
      Serial.print("Received Value: ");
      for (int i = 0; i < value.length(); i++)
      {
        Serial.print(value[i]);
      }
      // Send back the received value
      Serial.printf(" From MAC: %02X:%02X:%02X:%02X:%02X:%02X\n",
                    param->write.bda[0], param->write.bda[1],
                    param->write.bda[2], param->write.bda[3],
                    param->write.bda[4], param->write.bda[5]);
      // Send back the received value
      // Add to message queue
      if (!addMessageToQueue(param->write.bda, (uint8_t *)value.c_str(),
value.length()))
      {
        Serial.println("Message queue is full, dropping message");
      }
      pCharacteristic->setValue(value);
      pCharacteristic->notify();
      Serial.println();
    }
  }
};
```

Fișierul minesweeper.cpp

```
#define WIDTH 8
#define HEIGHT 16

#define NUM_BOMBS (WIDTH * HEIGHT / 10)
```

```
class Minesweeper
{
private:
    uint8_t bombs[NUM_BOMBS];
    uint8_t flag_is_revealed[(WIDTH * HEIGHT + 7) / 8]; // common for both
players
    uint8_t player_position[2];
    uint8_t marked_as_bomb[2][(WIDTH * HEIGHT + 7) / 8]; // For marking
positions as bombs

    int player_turn; // 0 or 1, which player is currently playing
    bool is_lost;
    void _reveal_until_neighbouring_bomb(uint8_t position);

public:
    Minesweeper();
    static inline uint8_t get_x_pos(uint8_t position);
    static inline uint8_t get_y_pos(uint8_t position);
    uint8_t is_bomb(uint8_t position)
    {
        {
            for (int i = 0; i < NUM_BOMBS; i++)
            {
                if (bombs[i] == position)
                {
                    return 1;
                }
            }
            return 0;
        }
    }
    void move_player(command_t command);
    uint8_t get_player_position()
    {
        return player_position[player_turn];
    }
    inline bool is_revealed(uint8_t position);
    inline void set_revealed(uint8_t position);

    bool is_marked_as_bomb(uint8_t position);
    void set_marked_as_bomb(uint8_t position);

    void builtin_button_pressed();

    bool shoot();
    uint8_t how_many_neighbouring_bombs(uint8_t position);

    inline bool is_game_over()
    {
        return is_lost;
    }
```

```
    void draw_map(TFT_eSPI &tft);

    bool won();

    inline void set_player_turn(int turn)
    {
        player_turn = turn;
    }

    bool displayed_final = false;

};
```

Optimizări și detalii de implementare

Structuri de date optimizate

Pentru a economisi memorie, starea hărții este stocată eficient folosind biți individuali:

```
bool Minesweeper::is_revealed(uint8_t position)

{
    int32_t x = get_x_pos(position);
    int32_t y = get_y_pos(position);
    return (flag_is_revealed[x] & (1 << y)) != 0;
}
```

Audio asincron in bucla loop

Pentru a asigura faptul ca microcontrollerul poate primi mesaje, executa operatii in timp ce canta, initializam un timer si in cod ne vom referi mereu la el:

* Cod pentru folosirea unui timer custom:

```
hw_timer_t *my_timer = NULL;
uint32_t timerCounter = 0; // this increments every milisecond
void IRAM_ATTR timerISR()
{
  timerCounter++;
}

// Initializare:
my_timer = timerBegin(0, 80, true);                 // Timer 0, prescaler 80 (1
```

```
tick = 1 us)
timerAttachInterrupt(my_timer, &timerISR, true); // Attach the interrupt
timerAlarmWrite(my_timer, 1000000 / 1000, true); // Set alarm for 1/1000
second
timerAlarmEnable(my_timer);                      // Enable the alarm
```

* Cod in bucla main pentru buzzer

```
if (play_song)
  {
    if (currentNote < note_size)
    {
      int noteDuration = 1000 / curr_durations[currentNote];
      if (!action_playing)
      {
        action_playing = true; // Set the action as playing
        tone(BUZZZER_PIN, curr_notes[currentNote], noteDuration);
      }
      if (timerCounter - lastTimerCheck >= noteDuration + noteDuration / 4)
// Check if enough time has passed
      {
        lastTimerCheck = timerCounter; // Update the last check time
        noTone(BUZZZER_PIN);
        currentNote++;
        Serial.printf("Current note: %d, Duration: %d ms\n",
                      currentNote, noteDuration);
        action_playing = false;
      }
    }
    else
    {
      play_song = false; // Stop playing when all notes are done
      currentNote = 0;   // Reset for next time
    }
  }
```

Implementare thread-safe

Coada de mesaje este implementată într-un mod thread-safe, folosind secțiuni critice pentru a preveni condițiile de cursă:

```
portENTER_CRITICAL(&messageQueueMux);

// Operațiuni pe coadă

portEXIT_CRITICAL(&messageQueueMux);
```

Rezultate Obţinute

Care au fost rezultatele obţinute în urma realizării proiectului vostru.

Concluzii

Concluzii:

- Comunicare wireless Bluetooth Low Energy (BLE)
- Joc simultan pentru doi jucători
- Sistem audio complet cu muzică și efecte sonore
- Ecran interactiv cu funcționalitate de marcare a bombelor
- Interfață de meniu principal și gestionarea stărilor jocului
- Detecția câștigătorului/învinsului și sisteme de feedback
- Workflow nonblocant in cadrul utilizarii
- Proiectul nu da crash

Download

O arhivă (sau mai multe dacă este cazul) cu fişierele obţinute în urma realizării proiectului: surse, scheme, etc. Un fişier README, un ChangeLog, un script de compilare şi copiere automată pe uC crează întotdeauna o impresie bună 😉.

Fişierele se încarcă pe wiki folosind facilitatea **Add Images or other files**. Namespace-ul în care se încarcă fişierele este de tipul **:pm:prj20??:c?** sau **:pm:prj20??:c?:nume_student** (dacă este cazul). **Exemplu:** Dumitru Alin, 331CC → **:pm:prj2009:cc:dumitru_alin**.

Link catre codul ESP32: https://github.com/HoriaMercan/PM_project

Link catre aplicatia self-made pentru comunicatie: https://github.com/HoriaMercan/android_joystick_for_esp32_bluetooth

Jurnal

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

Laboratoare folosite:

- GPIO

- Intreruperi

- Timere

- Bonus: BLE communication

30 aprilie 2025 - Configurare Inițială Commit:
`7dfe52f2b2996f25b5107adbca77ea00c26fe5f4`

- **Commit inițial, Bluetooth funcționează**
- A fost configurată structura fundamentală a proiectului PlatformIO
- A fost implementată funcționalitatea de bază Bluetooth

5 mai 2025 - Integrarea Componentelor Principale Commit:
`7a76a08a1dfcdb8888a6efa2abc4b1a0aec38723`

- **Bluetooth și ecranul funcționează**
- Funcționalitatea ecranului a fost integrată cu succes cu Bluetooth
- A fost stabilită comunicarea între componentele hardware

11 mai 2025 - Mecanicile Jocului și Comunicarea Commit:
`e679104dedf167d8ba7de339504adad6fdf984ae`

- **Comunicare configurată. Toate funcțiile jocului implementate. Jucătorul se poate mișca**
- Au fost implementate controalele de mișcare ale jucătorului
- A fost stabilit protocolul de comunicare complet
- Mecanicile de bază ale jocului Minesweeper sunt funcționale

12 mai 2025 - Optimizare Cod și Migrare BLE Commit:
`c557d4cb5135d9c70ee7b7e118a7e770c31b7958`

- **Jocul funcționează bine. Încercare BLE**
- Funcționalitatea jocului a fost rafinată
- A început tranziția de la Bluetooth clasic la Bluetooth Low Energy (BLE)

Commit: `310101410c4b5ee8bb2b291ea0f39cd7c4ef2ba9`

- **Actualizare gitignore**
- Mentenanță și curățare proiect

18 mai 2025 - Punct de Referință Stabilitate Commit:
`a5f01f2c6d7bfe9e641f310086ed531b068262ae`

- **Funcționează**
- A fost atinsă o operare stabilă a jocului
- Toate funcțiile de bază funcționează corect

22 mai 2025 - Experiență de Utilizare Îmbunătățită Commit:
`9f0fe7772a90b76d46d15b94bf353dacd6cc0f4c`

- **S-au adăugat muzică, marcarea bombelor pe hartă și interfață câștigător/învins**
- Au fost adăugate funcții audio și muzică de fundal
- A fost implementată funcționalitatea de marcare a bombelor
- Au fost create interfețe pentru stările de joc câștigător/învins

23 mai 2025 - Succes BLE și Sistem de Meniu Commit:
`5919cd7aad8ce22be359a6508cbd5c36bf833d39`

- **Funcționează cu BLE**
- Implementarea BLE a fost finalizată cu succes
- A fost stabilită o comunicare wireless stabilă

Commit: `71f9cb224292d314461519b63771af14ee53feb2`

- **Meniu principal creat. TODO: actualizare jucător curent**
- A fost dezvoltată interfața meniului principal
- A fost identificată nevoia de actualizare a indicatorului jucătorului curent

24 mai 2025 - Multiplayer și Îmbunătățiri Audio Commit:
`36b101f67488fbbf002290adc2a992c876b03d19`

- **Două conexiuni și gameplay pentru doi jucători realizate. Adăugare melodii diferite**
- Au fost implementate conexiuni Bluetooth duale
- Funcționalitatea multiplayer pentru doi jucători a fost finalizată
- Sistemul audio a fost îmbunătățit cu melodii multiple

Commit: `cb3467eaed00221d5fe8b6c4c2e69cd391d1cfcb`

- **Melodia de Game Over inclusă**
- A fost adăugat feedback audio pentru game over
- Experiența audio a fost completată cu muzică de final

Bibliografie/Resurse

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

Export to PDF