Faculty of Computer Applications

& Information Technology

iMCA PROGRAMME

Semester III

Project Report for

242301306 Introduction to Embedded Systems

Project Definition

ESP32 BLUETOOTH JOYSTICK FOR PC, ANDROID & PLAYSTATION

Submitted by:

Group No: 51

Harsh Parmar : 202402519010400

Rahil Pabari : 202402519010395

Arya Parmar : 202402519010398

1. ABSTRACT

This project presents the design and implementation of a Bluetooth joystick using ESP32 for multi-platform compatibility. The joystick works seamlessly with PC, Android, and PlayStation devices using BLE (Bluetooth Low Energy) protocol. The system integrates buttons, triggers, and analog joysticks mapped according to standard gamepad modes. The main objective is to provide a low-cost, wireless, and customizable alternative to commercial game controllers. The project demonstrates hardware interfacing, BLE configuration, and real-time data transfer for gaming applications. Expected outcomes include accurate control, reduced latency, and cross-platform support.

2. OBJECTIVE

The objective of this project is to design a universal Bluetooth game controller using ESP32 that can work with multiple platforms (PC, Android, and PlayStation). It aims to solve the problem of high-cost and platform-specific joysticks by offering a cost-effective, open-source, and student-friendly solution. The motivation behind the project is to enhance gaming experience, promote hardware design knowledge, and enable students to explore embedded system applications in real-time environments.

3. COMPONENTS

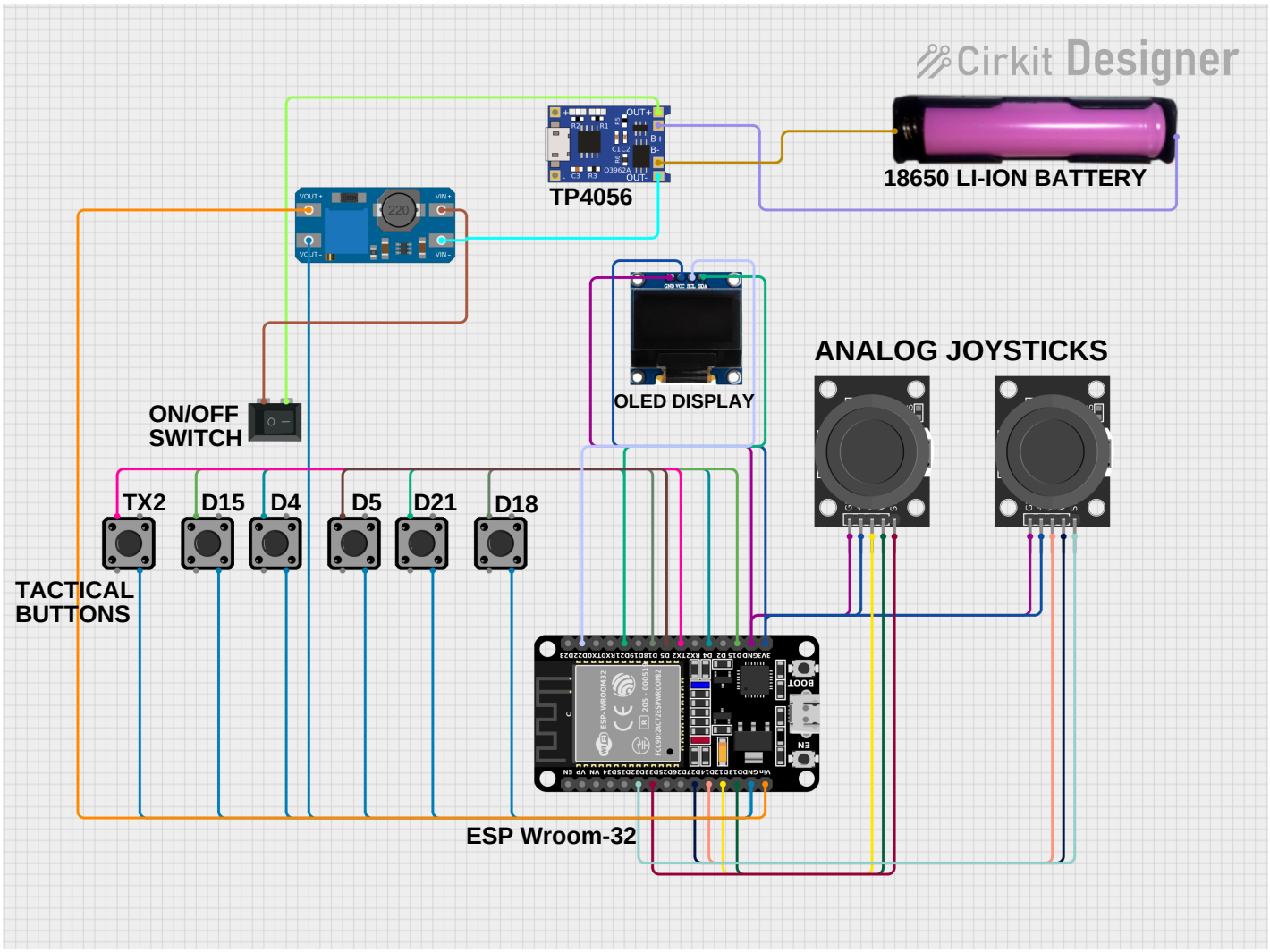
- **ESP32 – Main microcontroller with built-in Wi-Fi & Bluetooth for wireless communication.**
- **Push Buttons (13) – Used for gameplay input such as ABXY, triggers, start, select, PS, L3, and R3,(**in code we add 13 buttons but in gamepad we added 6 buttons**).**
- **Analog Joysticks (2) – Provides directional movement for left and right thumbsticks.**
- **Power Supply (5V/3.3V) – Provides operating voltage to ESP32 and components.**

Arduino IDE – Programming environment for ESP32.

BleGamepad Library – Enables Bluetooth HID Gamepad profile.

1.OLED DISPLAY - For Animation
2.TP4056
3.BOOSTER
4.LI-ION BATTERY
5.ON/OFF SWITCH

4. CONNECTION DIAGRAM



1. PROJECT CODE SNIPET WITH COMMENTS TO EXPLAIN CODE SECTIONS

```
#include <Arduino.h>
#include <BleGamepad.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// --- OLED SETTINGS ---
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// --- BITMAP: GAMEPAD ICON ---
const unsigned char PROGMEM gamepad_icon[] = {
  0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xff, 0xe0, 0x18, 0x00, 0x00, 0x18, 0x20, 0x00, 0x00, 0x04,
  0x40, 0x18, 0x18, 0x02, 0x40, 0x18, 0x18, 0x02, 0x8f, 0x18, 0x18, 0xf1, 0x89, 0x00, 0x00, 0x91,
  0x99, 0x00, 0x00, 0x99, 0x89, 0x00, 0x00, 0x91, 0x8f, 0xff, 0xff, 0xf1, 0x40, 0x00, 0x00, 0x02,
  0x40, 0x18, 0x18, 0x02, 0x20, 0x0f, 0xf0, 0x04, 0x18, 0x00, 0x00, 0x18, 0x07, 0xff, 0xff, 0xe0,
  0x00, 0x00, 0x00, 0x00
};

// --- PINS ---
#define X_BUTTON 15
#define CIRCLE_BUTTON 17
#define TRIANGLE_BUTTON 4
#define SQUARE_BUTTON 5

#define R1_BUTTON 18
#define R2_BUTTON 19

// NOTE: L1 is on RX (Pin 3). You must disconnect this wire while uploading code.
#define L1_BUTTON 3
#define L2_BUTTON 16

#define START_BUTTON 23
#define SELECT_BUTTON 26
#define PS_BUTTON 25
#define R3_BUTTON 33
#define L3_BUTTON 32

#define RIGHT_VRX_JOYSTICK 12
#define RIGHT_VRY_JOYSTICK 13
#define LEFT_VRX_JOYSTICK 14
#define LEFT_VRY_JOYSTICK 27

#define NUM_BUTTONS 13

int buttonsPins[NUM_BUTTONS] = {
  X_BUTTON, CIRCLE_BUTTON, TRIANGLE_BUTTON, SQUARE_BUTTON,
  R1_BUTTON, R2_BUTTON, L1_BUTTON, L2_BUTTON,
  START_BUTTON, SELECT_BUTTON, PS_BUTTON,
  L3_BUTTON, R3_BUTTON
};

// Mappings
int PCGamepadButtons[NUM_BUTTONS] = {1, 2, 4, 3, 6, 8, 5, 7, 9, 10, 13, 11, 12};

uint16_t leftVrxJoystickValue = 0;
uint16_t leftVryJoystickValue = 0;
uint16_t rightVrxJoystickValue = 0;
uint16_t rightVryJoystickValue = 0;

BleGamepad bleGamepad("ESP32_Pro_FX", "DIY");
BleGamepadConfiguration bleGamepadConfig;

// --- HELPER FUNCTIONS ---
int applyDeadzone(uint16_t value, int center = 16384, int deadzone = 2000) {
  if (value > center - deadzone && value < center + deadzone) return center;
  return value;
}
```

```
void playCoolBootAnim() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(20, 25);
  display.print("SYSTEM BOOT");

  // Quick expand line
  for(int w=0; w<64; w+=4) {
    display.drawFastHLine(64-w, 35, w*2, WHITE);
    display.display();
  }
  delay(100);

  display.clearDisplay();
  display.drawBitmap((128-32)/2, 10, gamepad_icon, 32, 17, WHITE);
  display.setCursor(45, 40);
  display.print("READY");
  display.display();
  delay(500);
}

void setup() {
  Serial.begin(115200);

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 failed"));
  } else {
    playCoolBootAnim();
  }

  for (int i = 0; i < NUM_BUTTONS; i++) {
    pinMode(buttonsPins[i], INPUT_PULLUP);
  }

  bleGamepadConfig.setAutoReport(false);
  bleGamepadConfig.setControllerType(CONTROLLER_TYPE_GAMEPAD);
  bleGamepad.begin(&bleGamepadConfig);
}

void loop() {
  // 1. Read Inputs
  uint16_t lx = analogRead(LEFT_VRX_JOYSTICK);
  uint16_t ly = analogRead(LEFT_VRY_JOYSTICK);
  uint16_t rx = analogRead(RIGHT_VRX_JOYSTICK);
  uint16_t ry = analogRead(RIGHT_VRY_JOYSTICK);

  // 2. Process Joystick
  leftVrxJoystickValue = applyDeadzone(map(ly, 0, 4095, 0, 32767));
  leftVryJoystickValue = applyDeadzone(map(lx, 0, 4095, 32767, 0));
  rightVrxJoystickValue = applyDeadzone(map(ry, 0, 4095, 0, 32767));
  rightVryJoystickValue = applyDeadzone(map(rx, 0, 4095, 32767, 0));

  // --- 3. ANIMATED HUD ---
  display.clearDisplay();

  // --- A. CENTER RADAR ---
  int centerX = 64;
  int centerY = 38;

  // Radar Box with corner accents
  display.drawRect(centerX-20, centerY-20, 40, 40, WHITE);
  display.fillRect(centerX-20, centerY-20, 3, 3, WHITE); // Corner
  display.fillRect(centerX+17, centerY-20, 3, 3, WHITE); // Corner
  display.fillRect(centerX-20, centerY+17, 3, 3, WHITE); // Corner
  display.fillRect(centerX+17, centerY+17, 3, 3, WHITE); // Corner

  int dotX = map(leftVrxJoystickValue, 0, 32767, centerX-18, centerX+18);
  int dotY = map(leftVryJoystickValue, 0, 32767, centerY-18, centerY+18);
  display.fillCircle(dotX, dotY, 2, WHITE);

  // --- B. LEFT SIDE (Square & Triangle) ---

  // SQUARE (Animation: Shield Effect)
  if(digitalRead(SQUARE_BUTTON) == LOW) {
    display.fillRect(5, 18, 10, 10, WHITE);
    display.drawRect(3, 16, 14, 14, WHITE); // Outer Ring Glow
  } else {
    display.drawRect(5, 18, 10, 10, WHITE);
  }

  // TRIANGLE (Animation: Energy Core)
  if(digitalRead(TRIANGLE_BUTTON) == LOW) {
    display.fillTriangle(10, 42, 5, 52, 15, 52, WHITE);
    display.fillCircle(10, 48, 1, BLACK); // Center hole (Core look)
    display.drawTriangle(10, 40, 3, 54, 17, 54, WHITE); // Outer Aura
  } else {
    display.drawTriangle(10, 42, 5, 52, 15, 52, WHITE);
  }
```

```
// --- C. RIGHT SIDE (X & Circle) ---

  // CROSS/X (Animation: Target Lock)
  if(digitalRead(X_BUTTON) == LOW) {
    // Bold X
    display.drawLine(113, 18, 123, 28, WHITE); display.drawLine(114, 18, 124, 28, WHITE);
    display.drawLine(123, 18, 113, 28, WHITE); display.drawLine(124, 18, 114, 28, WHITE);
    // Circle around X (Target Locked)
    display.drawCircle(118, 23, 8, WHITE);
  } else {
    display.drawLine(113, 18, 123, 28, WHITE); display.drawLine(123, 18, 113, 28, WHITE);
  }

  // CIRCLE (Animation: Sun Burst)
  if(digitalRead(CIRCLE_BUTTON) == LOW) {
    display.fillCircle(118, 48, 5, WHITE);
    // Rays coming out
    display.drawPixel(118, 40, WHITE); // Top
    display.drawPixel(118, 56, WHITE); // Bottom
    display.drawPixel(110, 48, WHITE); // Left
    display.drawPixel(126, 48, WHITE); // Right
  } else {
    display.drawCircle(118, 48, 5, WHITE);
  }

  // --- D. STATUS BAR (Clean) ---
  display.setTextSize(1);
  display.setCursor(35, 0);
  if(bleGamepad.isConnected()) {
    display.print("CONNECTED");
    display.fillRect(0, 10, 128, 2, WHITE); // Underline
  } else {
    if ((millis()/400)%2==0) {
      display.print("SEARCHING");
      display.drawFastHLine(35, 8, 54, WHITE);
    }
  }

  // --- E. TRIGGERS (Segmented Bars - Cool Tech Look) ---
  // Left Side (L1/L2)
  if(digitalRead(L1_BUTTON)==LOW) {
    // 3 Blocks for L1
    display.fillRect(0, 15, 4, 8, WHITE);
    display.fillRect(0, 25, 4, 8, WHITE);
    display.fillRect(0, 35, 4, 8, WHITE);
  }
  if(digitalRead(L2_BUTTON)==LOW) {
    // 2 Blocks for L2 (Bottom)
    display.fillRect(0, 45, 4, 8, WHITE);
    display.fillRect(0, 55, 4, 8, WHITE);
  }

  // Right Side (R1/R2)
  if(digitalRead(R1_BUTTON)==LOW) {
    display.fillRect(124, 15, 4, 8, WHITE);
    display.fillRect(124, 25, 4, 8, WHITE);
    display.fillRect(124, 35, 4, 8, WHITE);
  }
  if(digitalRead(R2_BUTTON)==LOW) {
    display.fillRect(124, 45, 4, 8, WHITE);
    display.fillRect(124, 55, 4, 8, WHITE);
  }

  display.display();

  // --- 4. BLE SEND ---
  if (bleGamepad.isConnected()) {
      for (int i = 0; i < NUM_BUTTONS; i++) {
        if (!digitalRead(buttonsPins[i])) bleGamepad.press(PCGamepadButtons[i]);
        else bleGamepad.release(PCGamepadButtons[i]);
      }
      bleGamepad.setX(leftVrxJoystickValue);
      bleGamepad.setY(leftVryJoystickValue);
      bleGamepad.setZ(rightVrxJoystickValue);
      bleGamepad.setRX(rightVryJoystickValue);

      bleGamepad.sendReport();
  }
  delay(10);
}
```

6. WORKING OF SYSTEM

The system works by reading button presses and joystick analog values from hardware inputs connected to ESP32. These inputs are mapped to Android, PS, or PC Gamepad button standards using the BleGamepad library. The ESP32 transmits the data wirelessly over Bluetooth, allowing the host device (PC/Android/PlayStation) to recognize it as a standard game controller. Deadzone logic is applied to joysticks for smooth movement. For Charging we added TP4056 with safety. For Animations we added OLED display and also added ON/OFF Switch.For converting battery's 3.7V to 5V we added Booster.

7. RESULT ANALYSIS

- The joystick successfully connects to PC, Android, and PlayStation without additional drivers.
- Button mappings work accurately as per selected mode (Android, PS, or PC).
- Joystick axes respond smoothly, and deadzone correction improves stability.
- The project demonstrates a low-cost, DIY game controller suitable for students and hobbyists.
- Limitations: Range depends on Bluetooth signal strength (10m)

SIGNATURE

_____