Gamepad realizat cu ESP32

Autor:
Solomon Andrei-Eduard

1. Rezumat

1.1 Descrierea proiectului

Proiectul constă în crearea unei console de joc utilizând microcontrollerul ESP32. Acest gamepad include patru butoane digitale și două joystick-uri analogice. Procesul implică configurarea microcontrollerului ESP32 cu ajutorul Arduino IDE, folosind biblioteci precum BLE GamePad și NIM BLE pentru software. După asamblarea circuitului și încărcarea codului sursă, gamepadul este testat prin conectarea Bluetooth la un laptop, verificând funcționalitatea butoanelor și a joystickurilor. Proiectul demonstrează o metodă eficientă de a crea un controller personalizat pentru jocuri pe calculator.

1.2 Descrierea implementării

Implementarea proiectului include următorii pași principali:

- **Configurarea ESP32**: Utilizarea Arduino IDE pentru a configura ESP32. Instalarea bibliotecilor BLE GamePad și NIM BLE necesare pentru comunicarea Bluetooth.
- **Asamblarea Circuitului**: Conectarea celor patru butoane digitale, a LED-ului albastru de verificare a conexiunii Bluetooth, și a joystick-urilor analogice la ESP32, folosind un breadboard.
- **Programarea ESP32**: Încărcarea codului sursă pe ESP32 pentru a defini funcționalitățile butoanelor și joystick-urilor.
- **Testarea Conexiunii Bluetooth**: Conectarea gamepad-ului la laptop prin Bluetooth și verificarea conectivității.
- **Testarea Funcționalității**: Utilizarea unei aplicații de testare pentru a verifica funcționarea corectă.
- **Utilizarea în Jocuri**: Testarea practică a gamepad-ului într-un joc pe PC pentru a asigura o experiență de joc optimă.

1.3 Rezultatele Obținute

La finalul proiectului am obținut o consolă de joc, care poate fi utilizată cu ușurință. Proiectul astfel incorporeaza aspecte ce se regasesc și în controllerele de pe piața accesoriilor de gaming.

2. Introducere

2.1 Prezentarea temei

Tema principală a proiectului a fost dezvoltarea unei console interactive care poate fii folosită în interacțiunea cu diverse jocuri. Am ales să folosim plăcuța de dezvoltare ESP32 DevKitC-V4 datorită capabilităților sale avansate de procesare, a costului redus și a posibilităților de conectivitate wireless. Gamepad-ul nostru este echipat cu patru butoane digitale și două joystick-uri analogice, permițându-ne să oferim o experiență de joc captivantă și interactivă. De asemenea, am inclus un LED albastru ca indicator de stare, care se aprinde atunci când gamepad-ul este conectat prin Bluetooth și se stinge când nu este conectat, oferind un feedback vizual. Alimentarea gamepad-ului se face prin intermediul unui cablu USB conectat la PC, asigurând o sursă de alimentare stabilă de 3.3V.

2.2 Motivația practică pentru alegerea temei

Alegerea acestei teme a fost motivată de dorința de a explora și de a implementa cunoștințe practice în domeniul microcontrolerelor și al sistemelor integrate. Proiectul conține două periferice de intrare diferite, și anume buton digital și joystick analogic, și un periferic de ieșire, anume LED-ul indicator. Utilizarea tehnologiei Bluetooth demonstrează capacitatea ESP32 de a gestiona comunicații wireless. Scopul a fost de a aplica teoria într-un cadru practic, creând un dispozitiv funcțional și ușor de folosit în diverse jocuri, fie prin intermediul unei compatibilități directe, fie prin intermediul unui emulator precum "X360CE". Cu ajutorul acestuia, putând mapa comenzile pe un controller simulat de Xbox 360, crescând astfel numărul jocurilor compatibile cu acest proiect.

3. Prezentarea platformei hardware

3.1 Caracteristici tehnice ale procesorului

Microcontrollerul care stă la baza proiectului este ESP32 DevKitC V4 bazat pe chip-ul ESP32 de la Espressif Systems. Acesta oferă o combinație de putere de procesare, capacitate Wi-Fi și Bluetooth, fiind ideal pentru proiecte IoT (Internet of Things). ESP32 DevKitC V4 vine echipat cu un procesor dual-core, capacitate bună de memorie și o gamă largă de posibilități de conectare, inclusiv I/O digitale, I2C, SPI, UART, și I2S. De asemenea, este compatibil cu platforme de dezvoltare cum ar fi Arduino, NodeMCU și FreeRTOS, facilitând dezvoltarea rapidă și flexibilă a unei varietăți

largi de aplicații. Dimensiunea sa compactă, împreună cu un consum redus de energie, o face ideală pentru proiecte mobile, autonome și de dimensiuni reduse.

Caracteristici Tehnice:

- Microcontroller: ESP32 DevKitC V4
- Tensiune de funcționare: 3.3 sau 5 Volți
- Pini GPIO (General Purpose Input Output): 34
- Canale PWM: 28
- UART: 3
- I2C: 2
- SPI: 4
- I2S: 2
- Pini de intrare analogici: 15
- Curent continuu pentru pinul de 3,3 V: 130 mA
- Memorie Flash: 16 MB
- SRAM: 520KB
- Viteza ceasului: 240 MHz

3.2 Interfețe disponibile

3.2.1 UART

Unul dintre cele mai de bază protocoale de comunicații din electronică este protocolul serial Universal Asynchronous Receive Transmit (UART). Protocolul UART permite ca două dispozitive să comunice între ele. Protocolul necesită două fire între dispozitivele care comunică; una pentru fiecare direcție de comunicare. Fiecare dispozitiv are un modul de transmisie și recepție independent. Când un dispozitiv transmite, acesta trimite date ca o serie de impulsuri. Fiecare impuls reprezintă un bit de date, deci un octet (8 biți) de date este trimis ca opt impulsuri pe fir. Aceste impulsuri sunt trimise cu o anumită sincronizare predefinită numită o rată de transmisie (Baud Rate) care trebuie înțeleasă de ambele dispozitive.

3.2.2 SPI

Interfața seriala SPI (Serial Peripheral Interface) este o interfața sincronă standard de mare viteză, ce operează în mod full duplex. Ea e folosită ca sistem de magistrală serială sincronă pentru transmiterea de date, unde circuitele digitale pot să fie interconectate pe principiul master-slave. Aici modul master/slave înseamnă că dispozitivul (circuitul) digital master inițiază cuvântul de date.

Mai multe dispozitive (circuite) digitale slave sunt permise cu slave select individual, adică cu selectare individuală.

3.2.3 I2C

Protocolul Inter Integrated Circuit (I2C) este un protocol creat pentru a permite mai multor circuite integrate "slave" sa comunice cu unul sau mai mute cipuri "master". Acest tip de comunicare poate fi folosit doar pe distanțe mici de comunicare si asemenea protocolului UART are nevoie doar de 2 fire de semnal pentru a trimite/primii informații.

3.2.4 I2S

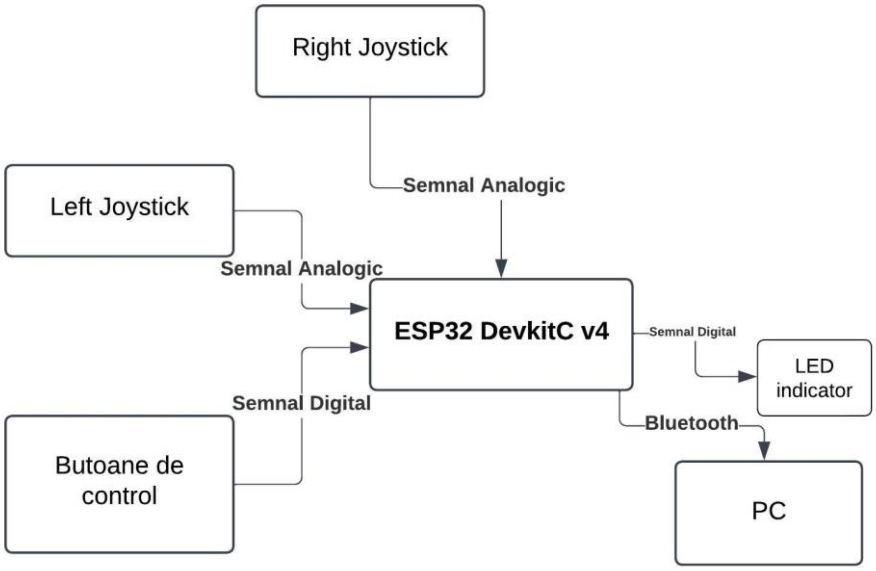
Interfața I2S (Inter-IC Sound) este un standard de comunicație serială pentru dispozitive audio digitale, dezvoltată inițial de Philips. Folosită pentru a conecta componente precum DAC-uri, ADC-uri și microfoane digitale, I2S transmite semnale audio digitale folosind o metodă eficientă care necesită puține conexiuni fizice. Această interfață separă canalele audio stâng și drept, oferind calitate înaltă a sunetului. Este o soluție simplă și eficientă, larg adoptată în industria electronică pentru diverse aplicații audio.

3.3 Modalitatea de alimentare a plăcii

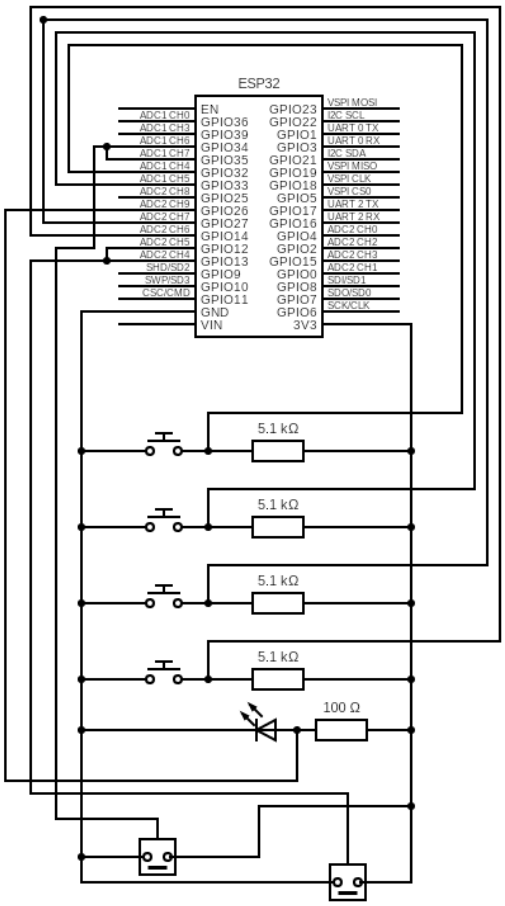
Întrucât piesele alese nu necesită alimentare externă și se alimentează direct de la 3,3 V, am reușit să le alimentăm direct din ESP32, astfel ținând tot sistemul la dimensiuni reduse.

Placa ESP32 se alimentează prin intermediul unui cablu Micro USB la calculator, prin care se pot transmite și date, astfel încât să se poată încărca și programul scris în C++, pe placă, cu care se poate folosi controller-ul.

4. Schema Bloc a Implementării



5. Schema Electrică



6. Costurile Realizării Practice

Nr. curent	Denumire componentă	Cantitate	Preț (ron)
1.	Plăcuța de dezvoltare ESP32 cu WiFi și Bluetooth	1	65
2.	Joystick Breakout Board	2	11
3.	Butoane de control (Black button with round cover)	4	8
4.	Breadboard	1	7
5.	Led albastru de 5 mm	1	0,3
6.	Rezistor 5,1kΩ	4	0,4
7.	Rezistor 100Ω	1	0,5
8.	Fire Colorate Tată-Tată (40p, 10 cm)	1	8
9.	Fire Colorate Mamă-Tată (10p, 20 cm)	1	4
TOTAL			104,2

7. Prezentare software

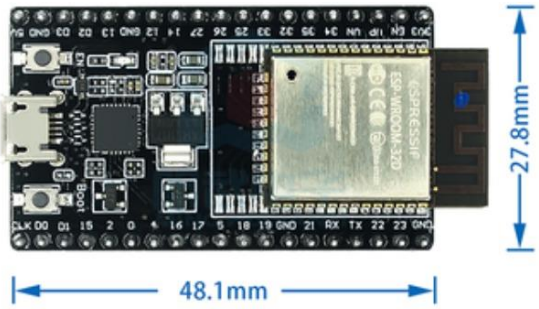
În proiectul de gamepad cu ESP32, am creat un cod în Arduino IDE care utilizează biblioteca BleGamepad pentru a gestiona comunicația Bluetooth între gamepad și un dispozitiv compatibil. Codul definește constante pentru butoanele principale ABXY și pentru două joystick-uri analogice. Am inclus și o definiție pentru un LED albastru, care indică starea conexiunii Bluetooth, aprinzându-se când gamepad-ul este conectat și stingându-se în absența unei conexiuni.

În bucla principală, verificăm starea conexiunii Bluetooth, citim valorile joystick-urilor și procesăm intrările de la butoane. Am implementat o funcție de "deadzone" pentru a evita fluctuațiile minore ale joystick-urilor, asigurând astfel o experiență de joc mai precisă.

Deși în codul actual nu sunt utilizate butoanele de tip trigger, meniu sau butoanele joystick-urilor (toate configurate cu pinul 0), structura codului permite o adaptare și o extindere ușoară. Dacă dorim să adăugăm mai multe componente și să creștem complexitatea proiectului, aceste butoane suplimentare pot fi ușor integrate și configurate prin modificări simple în codul C++. Această flexibilitate ne oferă posibilitatea de a dezvolta și îmbunătăți gamepad-ul, adaptându-l la nevoile specifice ale utilizatorilor sau la cerințele unor jocuri mai complexe.

8. Prezentare montaj

- **Componentele principale:**
  - ESP32 DevKitC V4

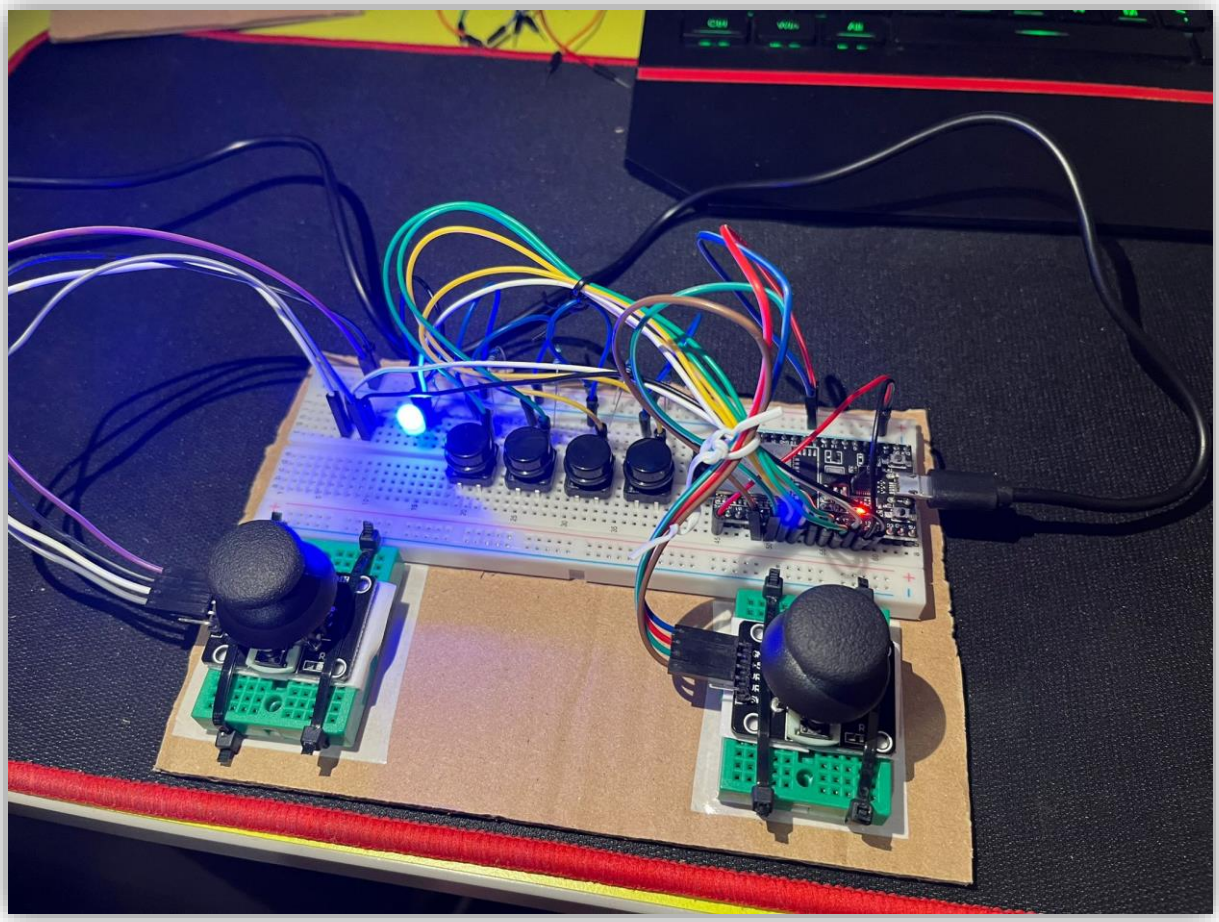


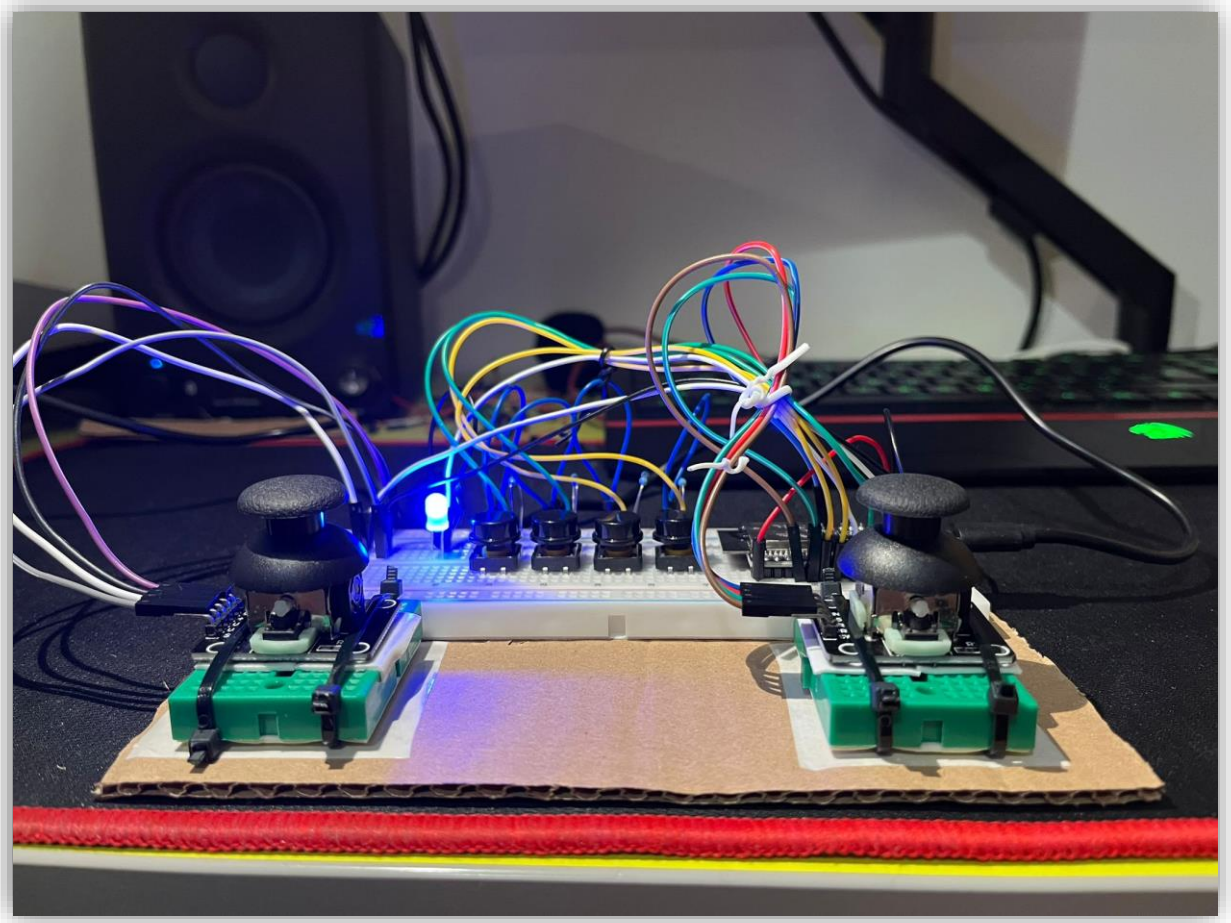
  - Joystick Analogic

  - Buton cu capac rotund

  - LED

- Montaj:





9. Concluzie

La finalizarea proiectului, am realizat un dispozitiv personalizat, eficient și cu costuri reduse. Lista componentelor necesare a inclus o placă de dezvoltare ESP32 cu Wi-Fi și Bluetooth, două joystick-uri, patru butoane de control, un breadboard, un LED albastru, rezistoare și fire colorate pentru conexiuni. Costul total al componentelor s-a ridicat la 104,2 RON, ceea ce demonstrează accesibilitatea financiară a construirii unui gamepad personalizat. Prin această inițiativă, mi-am îmbogățit cunoștințele practice în domeniul electronic și software, punând în aplicare principiile învățate. Acest proiect nu numai că încurajează învățarea practică și aplicarea teoretică, dar oferă și o bază solidă pentru extensii și îmbunătățiri viitoare.

10. Listing Cod

```
#include <Arduino.h>
#include <BleGamepad.h>

#define DEADZONE 2000  // Prag pentru ignorarea micilor schimbări
#define LED_PIN 26

// ABXY BUTTONS
#define X_BUTTON 32          // A
#define CIRCLE_BUTTON 33     // B
#define TRIANGLE_BUTTON 27   // Y
#define SQUARE_BUTTON 14     // X

// TRIGGERS
#define R1_BUTTON 0
#define R2_BUTTON 0
#define L1_BUTTON 0
#define L2_BUTTON 0

// MENU BUTTONS
#define START_BUTTON 0
#define SELECT_BUTTON 0
#define PS_BUTTON 0

// JOYSTICKS BUTTONS
#define R3_BUTTON 0
#define L3_BUTTON 0

// JOYSTICKS
#define LEFT_VRX_JOYSTICK 12
#define LEFT_VRY_JOYSTICK 13
#define RIGHT_VRX_JOYSTICK 35
#define RIGHT_VRY_JOYSTICK 34

#define NUM_BUTTONS 13


BleGamepad bleGamepad("ESP32 Gamepad", "ESP32");

BleGamepadConfiguration bleGamepadConfig;

uint16_t processJoystickValue(uint16_t currentValue, uint16_t
centerValue) {
    if (abs(currentValue - centerValue) < DEADZONE) {
        return centerValue; // Returnează valoarea de centru dacă
schimbarea este sub prag
    }
```

```
    return currentValue; // Altfel returnează valoarea curentă
}

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,
                                R3_BUTTON, L3_BUTTON};

int androidGamepadButtons[NUM_BUTTONS] = {1, 2, 3, 4, 8, 10, 7, 9,
12, 11, 13, 15, 14};
int PCGamepadButtons[NUM_BUTTONS] = {1, 2, 4, 3, 6, 8, 5, 7, 10, 9,
0, 12, 11};

uint16_t leftVrxJoystickLecture = 0;
uint16_t leftVryJoystickLecture = 0;
uint16_t rightVrxJoystickLecture = 0;
uint16_t rightVryJoystickLecture = 0;

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

typedef enum {ANDROID, PC} GamepadModes;
GamepadModes gamepadMode = PC;

void setup() {
  delay(1000);
  Serial.begin(115200);

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

  pinMode(LED_PIN, OUTPUT);  // Setează pinul LED ca ieșire

  bleGamepadConfig.setAutoReport(false);
  bleGamepadConfig.setControllerType(CONTROLLER_TYPE_GAMEPAD);
  bleGamepadConfig.setVid(0xe502);
  bleGamepadConfig.setPid(0xabcd);
  bleGamepadConfig.setHatSwitchCount(4);
  bleGamepad.begin(&bleGamepadConfig);
}

void loop() {
  if(bleGamepad.isConnected()) {
    digitalWrite(LED_PIN, HIGH); // Aprinde LED-ul
    // Joysticks lecture
    leftVrxJoystickLecture = analogRead(LEFT_VRX_JOYSTICK);
```

```
    leftVryJoystickLecture = analogRead(LEFT_VRY_JOYSTICK);
    rightVrxJoystickLecture = analogRead(RIGHT_VRX_JOYSTICK);
    rightVryJoystickLecture = analogRead(RIGHT_VRY_JOYSTICK);

    // Compute joysticks value
    leftVrxJoystickValue =
processJoystickValue(map(leftVrxJoystickLecture, 0, 4095, 0, 32737),
16368);
    leftVryJoystickValue =
processJoystickValue(map(leftVryJoystickLecture, 0, 4095, 0, 32737),
16368);
    rightVrxJoystickValue =
processJoystickValue(map(rightVrxJoystickLecture, 0, 4095, 0,
32737), 16368);
    rightVryJoystickValue =
processJoystickValue(map(rightVryJoystickLecture, 0, 4095, 0,
32737), 16368);

    switch(gamepadMode) {
      case ANDROID:
        for(int i = 0; i < NUM_BUTTONS; i++) {
          if(!digitalRead(buttonsPins[i])) {
            bleGamepad.press(androidGamepadButtons[i]);
          } else {
            bleGamepad.release(androidGamepadButtons[i]);
          }
          joysticksHandlerForMobile(leftVrxJoystickValue,
leftVryJoystickValue, rightVrxJoystickValue, rightVryJoystickValue);
        }
        break;

      case PC:
        for(int i = 0; i < NUM_BUTTONS; i++) {
          if(!digitalRead(buttonsPins[i])) {
            bleGamepad.press(PCGamepadButtons[i]);
          } else {
            bleGamepad.release(PCGamepadButtons[i]);
          }
          joysticksHandlerForPC(leftVrxJoystickValue,
leftVryJoystickValue, rightVrxJoystickValue, rightVryJoystickValue);
        }
        break;
    }

    bleGamepad.sendReport();
  }
  else {
    digitalWrite(LED_PIN, 0); // Stinge LED-ul
  }
}
```

```
void joysticksHandlerForMobile(uint16_t leftVrx, uint16_t leftVry,
uint16_t rightVrx, uint16_t rightVry){
  bleGamepad.setLeftThumb(leftVrx, leftVryJoystickValue);
  bleGamepad.setRightThumb(rightVrxJoystickValue,
rightVryJoystickValue);
}

void joysticksHandlerForPC(uint16_t leftVrx, uint16_t leftVry,
uint16_t rightVrx, uint16_t rightVry){
  bleGamepad.setX(leftVrxJoystickValue);
  bleGamepad.setY(leftVryJoystickValue);
  bleGamepad.setZ(rightVrxJoystickValue);
  bleGamepad.setRX(rightVryJoystickValue);
}
```