Wireless Glove Controller - Safronii Veaceslav

Introducere

Functionalitatea:

Wireless Glove Controller este un dispozitiv sub formă de mănușă care permite controlul calculatorului prin mișcări ale mâinii. Folosind gesturi simple, utilizatorul poate deplasa cursorul sau simula apăsarea unor taste, fără a atinge efectiv tastatura sau mouse-ul.

Scopul proiectului:

Crearea unei alternative la metodele clasice de interacțiune cu calculatorul, printr-un control intuitiv și hands-free. Este gândit să fie o soluție flexibilă, utilă în situații în care utilizatorul nu poate folosi periferice tradiționale sau în care se dorește o formă de interacțiune mai naturală și mai liberă.

Ideea de start:

Ideea a pornit din dorința de a combina funcționalitatea unui controller modern cu libertatea de mișcare oferită de un sistem wireless. M-au inspirat controlerele din realitatea virtuală și tehnologiile folosite în asistența persoanelor cu dizabilități.

Utilitatea:

Acest tip de control ar putea fi util în multe scenarii: de la gaming, la prezentări, până la aplicații educaționale sau medicale. Pentru mine, este o oportunitate de a aplica practic noțiunile studiate și de a învăța mai multe despre integrarea senzorilor, comunicație wireless și procesare de semnal într-un sistem real și interactiv.

Descriere generală

Hardware Design

Sistemul este alimentat de un **acumulator Li-Po 103450** de 3.7V, care este conectat la un **modul de încărcare TP4056**, responsabil pentru încărcarea sigură a bateriei printr-un port USB-C. Ieșirea

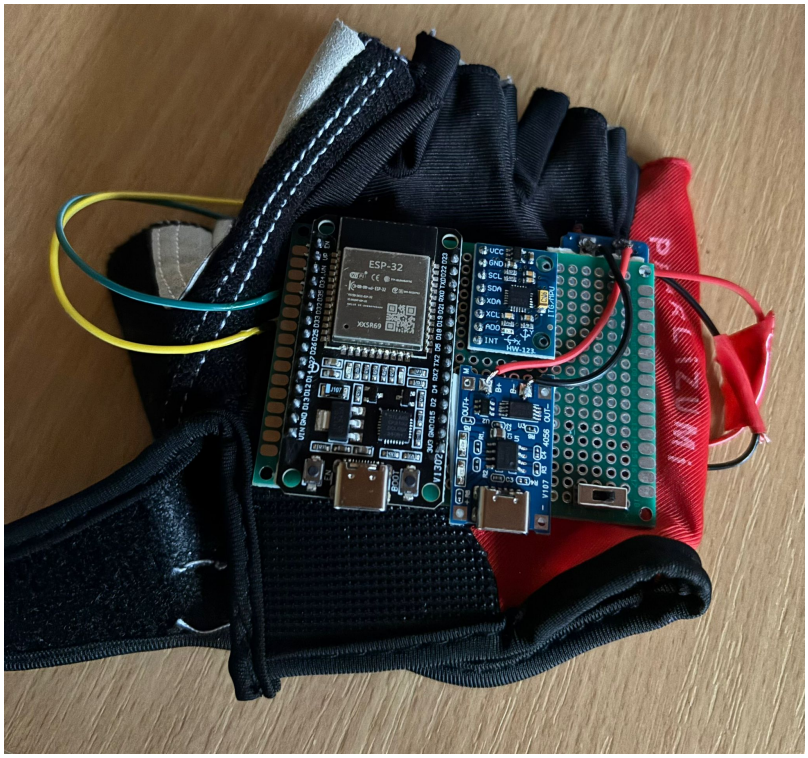
modulului TP4056 este conectată la un **ridicător de tensiune MT3608**, care converteste tensiunea de 3.7V la 5V, necesară pentru alimentarea **plăcii ESP32-WROOM-32** prin pinul VIN.

Senzorul de mișcare MPU6050, care integrează un accelerometru și un giroscop pe 3 axe, este conectat la ESP32 prin **interfața I²C** – folosind pinii SDA (GPIO 21) și SCL (GPIO 22). Acest senzor permite detectarea orientării și mișcării mâinii, informație esențială pentru aplicația glove-based.

Pentru interacțiunea utilizatorului, sistemul include două **butoane tactile**, conectate la **pinii GPIO** ai ESP32 (GPIO 34 și 35), folosind rezistențele de pull-up interne.

Piese:

Componentă	**Descriere / Observație**
ESP32-WROOM-32 Devkit v1	Placă de dezvoltare cu WiFi și Bluetooth
Sensor cu giroscop și accelerometru	MPU6050 – comunicare I²C
Modul de încărcare	TP4056 – pentru baterii Li-Ion/Li-Po
Ridicător de tensiune	MT3608 – boost la 5V pentru ESP32
Acumulator litiu-polimer	103450 – 3.7V, 2000mAh
Butoane tactile	Pentru input manual
Placa de prototipare cablaj	PCB



Software Design

Mediu de dezvoltare

1. **PlatformIO** - Mediu integrat de dezvoltare (IDE) bazat pe Visual Studio Code, specializat pentru dezvoltarea sistemelor embedded
2. **Framework-ul Arduino** pentru ESP32, care oferă un strat de abstractizare peste SDK-ul nativ

ESP-IDF

Librării și surse third-party

1. **Biblioteca BLE Arduino** - Implementarea oficială a stivei Bluetooth Low Energy pentru ESP32
2. **Biblioteca MPU6050** - Adaptată și optimizată pentru comunicarea directă I2C cu senzorul de mișcare
3. **Bleak** - Bibliotecă Python cross-platform pentru comunicarea Bluetooth Low Energy folosită în aplicația receptor

Algoritmi și structuri implementate

1. Structura modulară a firmware-ului

Codul firmware-ului a fost organizat în module funcționale pentru o mai bună întreținere și scalabilitate:

- **main.cpp** - Punct de intrare, inițializare și bucla principală
- **config.h** - Constante, definiri și configurație globală
- **ble_manager** - Gestionarea comunicației Bluetooth Low Energy
- **i2c_interface** - Implementarea protocolului I2C pentru comunicare cu senzori
- **mpu6050** - Comunicare și procesare date de la senzorul inerțial MPU6050
- **controller** - Logica de control și mapare a gesturilor la acțiuni

Inițializarea și configurarea ESP32

```
void setup() {
  Serial.begin(SERIAL_BAUD_RATE);
  Serial.println("Wireless Glove Controller initializing...");

  // Setup I2C pins
  pinMode(SDA_PIN, INPUT_PULLUP);
  pinMode(SCL_PIN, INPUT_PULLUP);

  // Configure button with interrupt
  pinMode(MODE_BUTTON_PIN, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(MODE_BUTTON_PIN), buttonISR,
FALLING);

  // Initialize systems
  i2c_init();
  mpu6050_init();
  initBLE();
  calibrate_sensors();

  // Configure timer for sensor sampling
  sampleTimer = timerBegin(0, 80, true);
  timerAttachInterrupt(sampleTimer, &onSampleTimer, true);
  timerAlarmWrite(sampleTimer, SAMPLE_RATE_MS * 1000, true);
  timerAlarmEnable(sampleTimer);
}
```

Handler de întrerupere pentru eșantionare

```
void IRAM_ATTR onSampleTimer() {
  portENTER_CRITICAL_ISR(&timerMux);
  readSensorFlag = true;
  portEXIT_CRITICAL_ISR(&timerMux);
}
```

Bucla principală

```
void loop() {
  // Process sensor data when ready
  if (readSensorFlag) {
    readSensorFlag = false;

    if (deviceConnected) {
      process_motion_data();
    }
  }

  // Handle button press
  if (buttonInterruptOccurred) {
    buttonInterruptOccurred = false;
    handle_button_interrupt();
  }

  // Manage BLE reconnections
  handleBLEEvents();

  delay(1);
}
```

Citirea datelor de la MPU6050

```
void mpu6050_read_accel(int16_t *accelX, int16_t *accelY, int16_t *accelZ) {
  uint8_t buffer[6];

  read_registers(MPU6050_ADDR, MPU6050_ACCEL_XOUT_H, buffer, 6);

  *accelX = (int16_t)((buffer[0] << 8) | buffer[1]);
  *accelY = (int16_t)((buffer[2] << 8) | buffer[3]);
  *accelZ = (int16_t)((buffer[4] << 8) | buffer[5]);
}
```

Procesare mișcare mouse

```
if (currentMode == MOUSE_MODE) {
  // Apply dead zone filter
  if (abs(deltaGyroX) < GYRO_DEADZONE) deltaGyroX = 0;
  if (abs(deltaGyroY) < GYRO_DEADZONE) deltaGyroY = 0;
```

```
  // Non-linear scaling for better precision
  float scaledX = (float)deltaGyroY / GYRO_SCALE_MAX;
  float scaledY = (float)deltaGyroX / GYRO_SCALE_MAX;

  int16_t mouseX = -scaledX * mouseAcceleration * MOUSE_SPEED_MAX;
  int16_t mouseY = scaledY * mouseAcceleration * MOUSE_SPEED_MAX;

  send_mouse_command(mouseX, mouseY);
}
```

Trimitere comandă BLE

```
void send_mouse_command(int16_t x, int16_t y) {
  if (!deviceConnected) return;

  // Limită pentru valorile ±127
  x = constrain(x, -127, 127);
  y = constrain(y, -127, 127);

  // Construcție pachet
  uint8_t buffer[4];
  buffer[0] = MOUSE_MOVE;
  buffer[1] = (uint8_t)(x & 0xFF);
  buffer[2] = (uint8_t)(y & 0xFF);
  buffer[3] = 0;  // Scroll

  // Transmisie BLE
  pTxCharacteristic->setValue(buffer, 4);
  pTxCharacteristic->notify();
}
```

Calibrare senzori

```
const int numSamples = 50;
for (int i = 0; i < numSamples; i++) {
  mpu6050_read_accel(&accelX, &accelY, &accelZ);
  mpu6050_read_gyro(&gyroX, &gyroY, &gyroZ);

  sumAccelX += accelX;
  sumAccelY += accelY;
  sumAccelZ += accelZ;

  delay(10);
}

// Calculare valori de bază
baseAccelX = sumAccelX / numSamples;
baseAccelY = sumAccelY / numSamples;
baseAccelZ = sumAccelZ / numSamples;
```

2. Aplicație Python (Receptor)

1. Comunicare Bluetooth:
   - **notification_handler()** - Procesarea notificărilor BLE primite
   - **_connect()** - Stabilirea conexiunii cu dispozitivul controller
2. Procesare comenzi:
   - **process_mouse_move()** - Interpretarea și aplicarea mișcărilor mouse-ului
   - **process_key_press()** / **process_key_release()** - Gestionarea apăsărilor de taste
3. Interfață utilizator
   - **create_gui()** - Construirea interfeței grafice
   - **log()** - Sistemul de înregistrare a evenimentelor pentru debugging

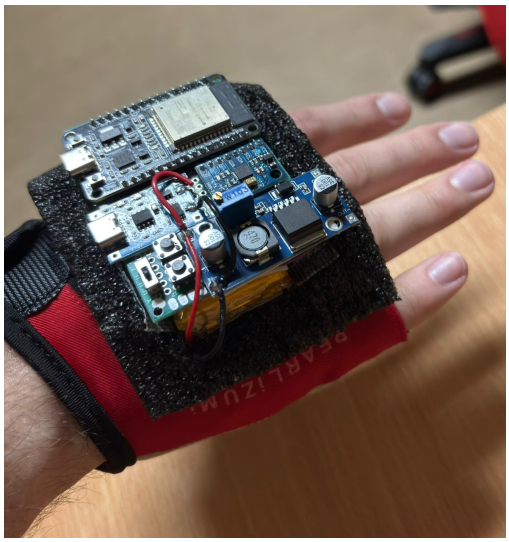
Handler notificări BLE (Python)

```
def notification_handler(self, sender, data):
    # Adaugă datele la buffer
    self.receive_buffer.extend(data)

    # Extrage comanda
    cmd = self.receive_buffer[0]

    if cmd == MOUSE_MOVE and len(self.receive_buffer) >= 4:
        x = struct.unpack('b', bytes([self.receive_buffer[1]]))[0]
        y = struct.unpack('b', bytes([self.receive_buffer[2]]))[0]
        self.process_mouse_move(x, y)
        self.receive_buffer = self.receive_buffer[4:]
```

Rezultate Obţinute

Demo:

Download

Codul sursa:

- src/ - Codul pentru ESP
- receptor/ - Programul in Python de receptie a comenzilor

wirelesscontroller.zip