FALL DETECTION SYSTEM FOR THE ELDERLY

A project report on fall detection using ESP32, MPU6050, and Telegram alerts.

GROUP MEMBERS

GAVI LEMUEL MAWUTOR – 1804622

MOHAMMED SHAMSUDEEN JAMILATU – 1805822

BAAH-OPOKU COMFORT DZIFA – 1808522

OSEI HANNAH ADOBEA – 1806722

KING LOUIS MACIAH – 1805522

MARIAM BENSON – 1802922

SAKYI MENSAH JOSEPHINE – 1807422

ANITA AKOSUA NUFUSIASIN – 1806222

HENRIQUES JAIME FERNANDES – 7085221

ABASS RASHEED – 1799422

Contents

1. Problem Statement

Elderly people are at risk of falling over and seriously injuring or killing themselves if help does not arrive early. And because many of them live alone, there is often nobody nearby to help them right after a fall.

Our Solution

We developed a Fall Detection System with an ESP32 microcontroller and an MPU6050 accelerometer. This system:

- Tracks real-time movement with sensor data
- It uses a sudden drop in acceleration to detect falls.
- When fall is detected, activates alert (via buzzer and LED).
- Sends a Telegram alert (in a real ESP32 with Wi-Fi this function works, but in our simulation the Wi-Fi connection is not made).

2. Methodology

Approach Taken

1. **Sensor Selection:**
    - The **MPU6050 accelerometer/gyroscope** is used because it can measure acceleration in the X, Y, and Z axes as well as angular velocity.
2. **Data Collection:**
    - The sensor constantly collects movement data in all directions.
3. **Fall Detection Algorithm:**
    - A fall is detected when the **total acceleration magnitude** drops below a preset threshold (FALL_THRESHOLD), indicating a moment of free-fall.
4. **Alert System:**
    - When fall is detected, the system triggers a buzzer and a LED to give a local alert.
    - In a real-world scenario, it would also send a **Telegram message** via Wi-Fi. However, in the simulation, the Wi-Fi connection is not available; instead, the Telegram alert is simulated.

Understanding the MPU6050 (X, Y, Z Axes in Real Life)

The MPU6050 sensor measures movement in three dimensions:

Axis	**Real-Life Representation**
X-Axis	Forward & backward movement (walking, leaning)
Y-Axis	Side-to-side movement (shifting weight, tilting sideways)
Z-Axis	Up & down movement (standing, falling, jumping)

How the Values Change During a Fall:

- **Standing Still:** The Z-axis normally reads about 9.8 m/s² (gravity) while X and Y are near 0.
- **During a Fall:** The Z-axis drops close to 0, indicating free-fall.
- **On Impact:** A sudden spike occurs in one or more axes as the body contacts the ground.

Code Explanation

1. Reading Sensor Data

The ESP32 reads acceleration from the MPU6050 using:

```
sensors_event_t a, g, temp;
mpu.getEvent(&a, &g, &temp);
```

- **a:** Contains acceleration values for X, Y, and Z.
- **g:** Contains gyroscope data (rotation).
- **temp:** Contains temperature data (not used here).

2. Detecting a Fall

The total acceleration magnitude is calculated:

```
float accelMagnitude = sqrt(sq(a.acceleration.x) + sq(a.acceleration.y) + sq(a.acceleration.z));
```

- **Detection Logic:** If accelMagnitude falls below FALL_THRESHOLD (set to 2.0), it indicates a fall.

3. Activating Buzzer & LED

When a fall is detected:

```
if (accelMagnitude < FALL_THRESHOLD) {
  Serial.println("⚠️ Fall Detected! Activating Buzzer & LED...");
```

```
  digitalWrite(BUZZER_PIN, HIGH);
  digitalWrite(LED_PIN, HIGH);
  delay(1000);
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN, LOW);
  // Telegram alert simulation follows...
}
```

- **Action:** The buzzer and LED are turned on for 1 second to alert nearby individuals.

4. Setting the Telegram Recipient

The recipient of the Telegram alert is determined by two parameters:

- **telegramBotToken:** Generated using Telegram's BotFather when the bot is created.
- **chatID:** This unique identifier is obtained by sending a message to the bot and checking the response from the Telegram API (using getUpdates).
  In the code, these are defined as:

  ```
  const char* telegramBotToken = "7775742203:AAGPh5-BJaZahAW4FRlX8_xyqCfmsOaRKe0";
  const char* chatID = "6973280311";
  ```

this is the bottoken and chatID of the recipient we used

These values ensure that any Telegram alert sent by the ESP32 is delivered to the correct recipient's account.

5. Sending a Telegram Alert

On a real ESP32, the code attempts to send a Telegram message:

```
void sendTelegramMessage(String message) {
  if (WiFi.status() == WL_CONNECTED) {
```

```
  HTTPClient http;
  String url = "https://api.telegram.org/bot" + String(telegramBotToken) +
"/sendMessage?chat_id=" + String(chatID) + "&text=" + message;
  http.begin(url);
  int httpCode = http.GET();
  if (httpCode > 0) {
    Serial.println("✅ Telegram Alert Sent!");
  } else {
    Serial.println("❌ Telegram Failed!");
  }
  http.end();
 }
}
```

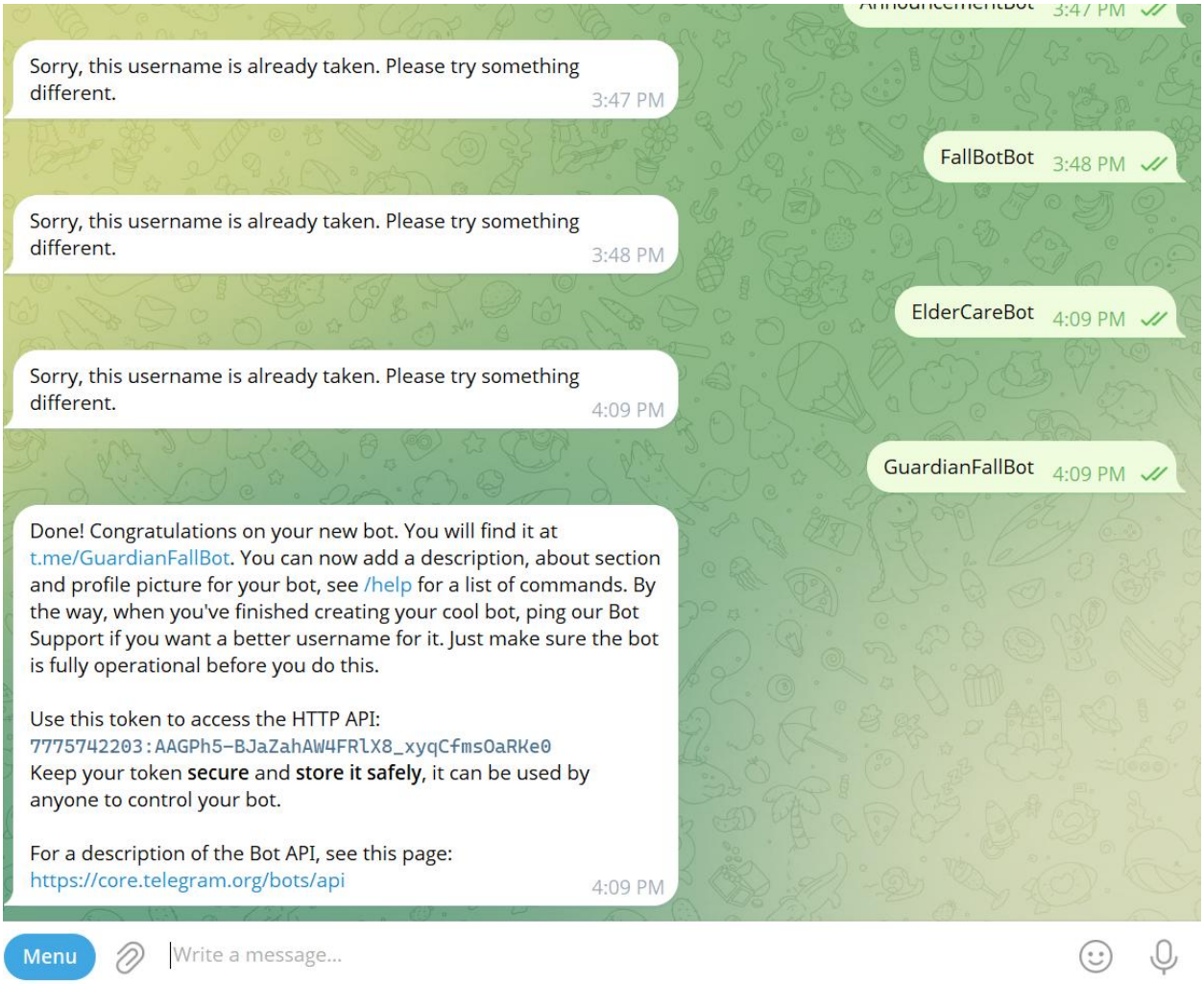

Figure 1 Bot Creation on Telegram

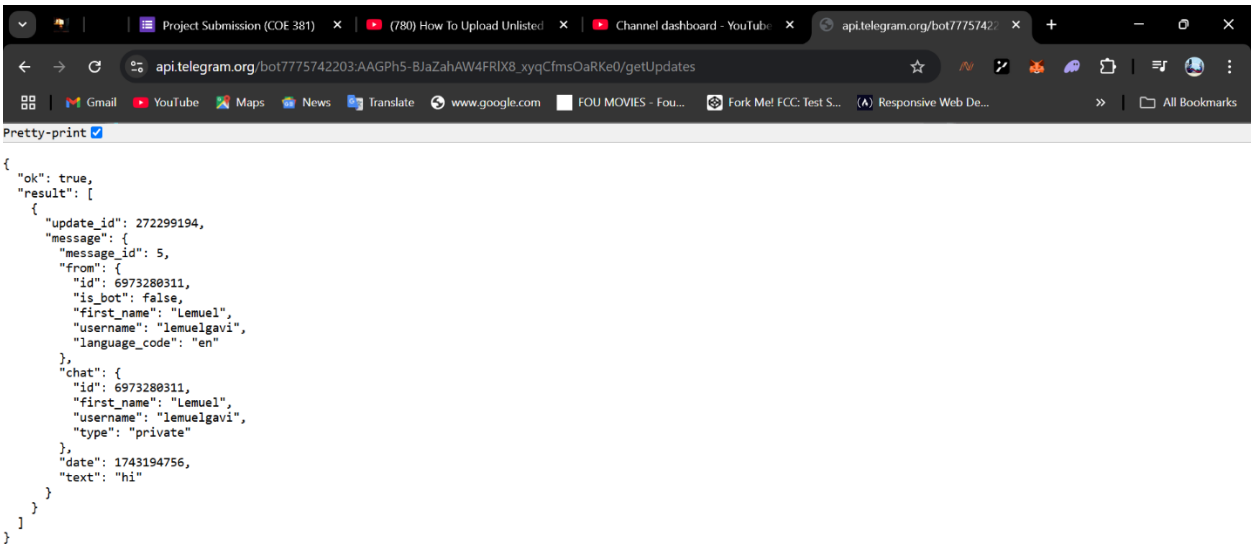

Figure 2 Telegram API Response

- **Note for Simulation:** Since Wokwi does not support Wi-Fi, the code is configured to simulate the Telegram alert by printing a message to the Serial Monitor instead of making any HTTP requests.

Circuit Schematic

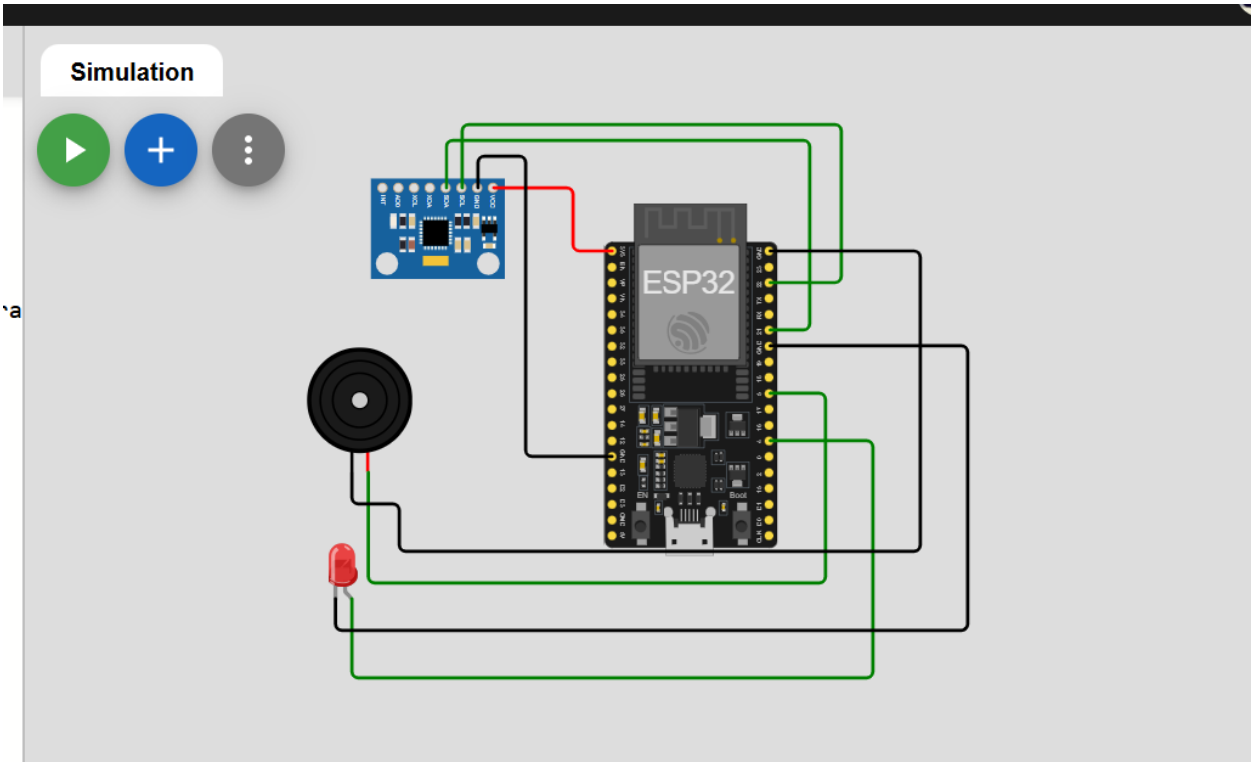

Figure 3 Wokwi diagram.json showing connections for ESP32, MPU6050, LED, and buzzer.

Component	**Connection on ESP32**
MPU6050 (I2C SDA)	esp 21
MPU6050 (I2C SCL)	esp 22
Buzzer (+)	esp 5
LED (+)	esp 4 (via resistor)

Demonstration & Simulation

Running the Simulation (Wokwi)

1. Open the Wokwi ESP32 Simulator.
2. Upload and run the provided code.
3. In the simulation, adjust the **Z-axis acceleration** (e.g., set it between -0.2g and 0.2g) to simulate a fall.

Expected Outputs

- **Before the Fall:** Serial Monitor shows normal acceleration.
- **During the Fall:**
    - When the acceleration drops below the threshold, the code notes that the fall has occurred.
    - The **buzzer and LED** are turned on.
    - Since Wokwi does not support Wi-Fi, a simulated Telegram alert is printed in the Serial Monitor Wokwi doesn't support Wi-Fi, so a simulated Telegram alert is printed in Serial Monitor:
    - 📩 [SIMULATED] Telegram Alert: 🚨 Fall Detected!

4. Conclusion

Findings

- The system accurately **detects falls** using acceleration data from the MPU6050.
- The **buzzer and LED** respond appropriately when a fall is detected.
- While the Telegram alert function cannot be executed in the simulation, it is designed to work on a real ESP32 with Wi-Fi.

Challenges

- **Simulation Limitations:**
    - Wokwi does not support real Wi-Fi connectivity, so Telegram alerts are only simulated.
- **Threshold Tuning:**
    - Fine-tuning the FALL_THRESHOLD is necessary to minimize false positives and ensure accurate fall detection.

Additional Note on Simulation

Since we are simulating the project with Wokwi, the ESP32 cannot establish a real Wi-Fi connection. However, the code contains a functionality to send a Telegram alert, this alert is simulated in the environment. On a real ESP32 with proper Wi-Fi connectivity, the Telegram message will be sent as intended.