IoT-Based Fall Detection System (using ESP-32 microcontroller)

Group 30:
Muskan Garg- 2023eeb1227
Parth Bansal- 2023eeb1231
Yatish Chaudhary- 2023eeb1428
Pragy Bagoriya - 2023eeb1234

This report presents an IoT-based fall detection system designed to monitor and alert caregivers when falls occur. Falls represent a significant health risk, particularly for elderly individuals, with the World Health Organisation reporting them as the second leading cause of unintentional injury deaths worldwide.

The system employs a three-tier architecture:

1. A device built on the ESP32 WROOM microcontroller with an integrated MPU6050 accelerometer
2. A Flask web server that processes and relays alerts
3. A Telegram bot interface for instant notifications to caregivers

1. Introduction

This IoT-based fall detection system addresses these limitations through an automated, sensor-based approach. The ESP32 WROOM's exceptional wireless capabilities and the MPU6050 accelerometer data enable threshold-based detection algorithms that can detect falls with high accuracy and immediately alert caregivers through multiple channels.

2. System Architecture Overview

Tier 1: ESP32 WROOM-based Device

- Hosts the MPU6050 accelerometer for motion detection
- Provides immediate local feedback via LED and buzzer
- Transmits alert data to the web server via Wi-Fi

Tier 2: Flask Web Server

- Receives and processes fall alerts from devices
- Forwards notifications to the Telegram messaging platform
- Implements database integration for long-term data storage

Tier 3: Telegram Bot Interface

- Provides instant notifications to caregivers
- Enables two-way communication for acknowledgement

Code Overview-

1. System Overview

- The system is designed to detect falls using an **MPU6050 accelerometer** and alert caregivers via a **Telegram bot** and a **web server**.
- It consists of three components:
    - Microcontroller with sensor and alert logic (ESP32 WROOM-based firmware).
    - Flask-based web server to receive alerts.
    - Telegram bot to forward alerts to caregivers.

2. WiFi & Device Configuration

- The ESP32 connects to WiFi using credentials defined.
- It retries the connection for up to 20 seconds before failing.
- It uniquely identifies the fall detector unit.

3. Fall Detection Logic

- The MPU6050 provides acceleration values along X, Y, Z axes.
- The code calculates the combined total acceleration using: $total\_accel = (accel\_x^2 + accel\_y^2 + accel\_z^2) \wedge 0.5$
- If total acceleration drops below a **threshold of 2.5g**, it is interpreted as a fall.

4. Local Alert System

- On fall detection:
    - LED and Buzzer are turned on for 2 seconds.
    - Then a 30-second cooldown prevents multiple alerts.
    - Alerts are printed via serial for debugging.

5. Sending Alerts to Server

- Containing:
    - Device ID
    - Timestamp
    - Acceleration values
    - Status ("FALL_DETECTED")
- Sent via HTTP POST request to a webhook URL.

6. Flask-Based Web Server

- Validates incoming data and logs alerts
- Converts the timestamp and acceleration values into a readable format.

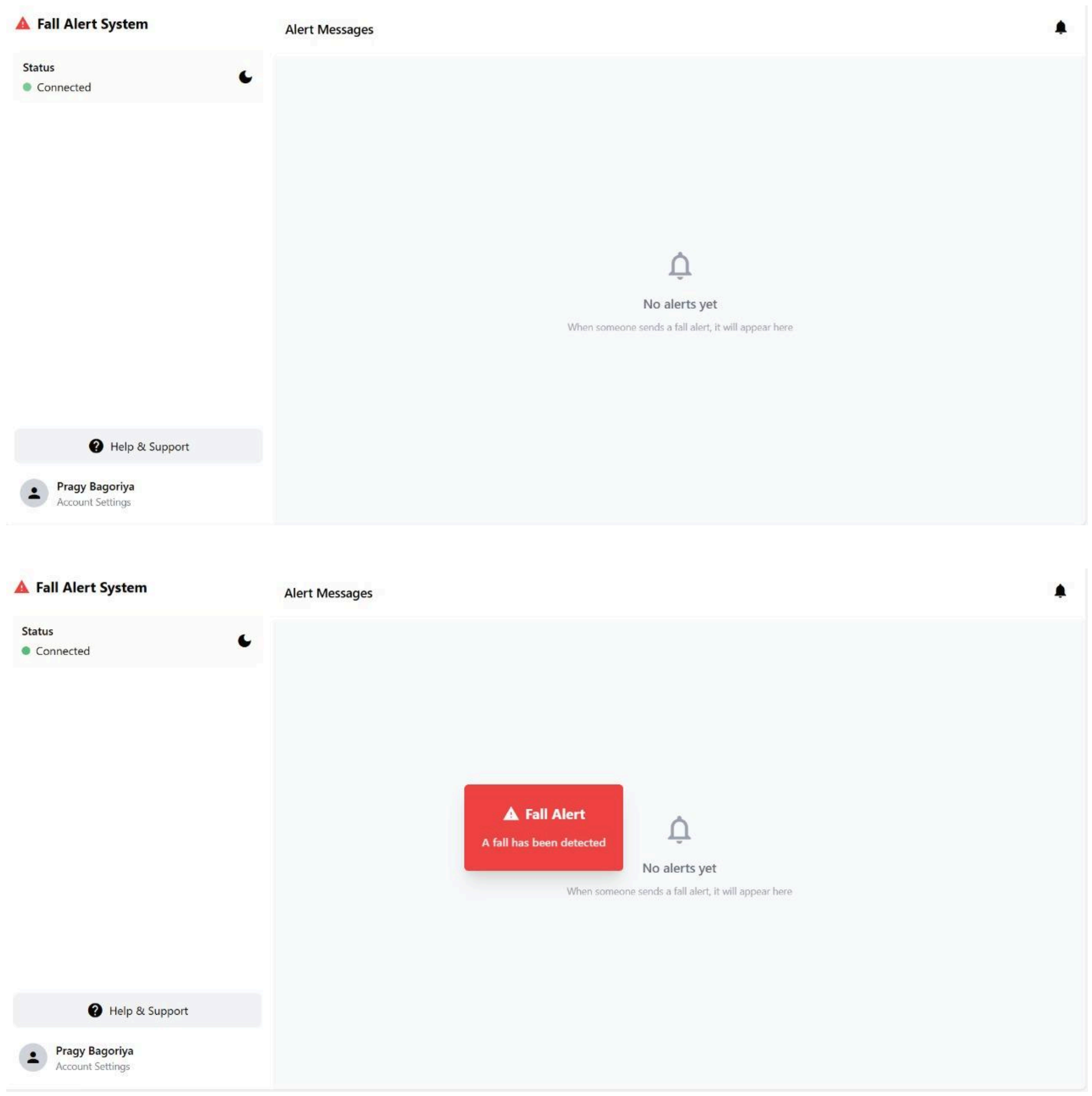
- Forwards the alert to a Telegram Bot via Telegram API.

7. Telegram Alert Bot

- Sends formatted alert messages to a specified Telegram ID with:
  - Device ID
  - Timestamp

8. Logging & Error Handling

- All alerts are logged with timestamps and device ID.
- Fallback messages are shown on Telegram



No alerts yet

When someone sends a fall alert, it will appear here

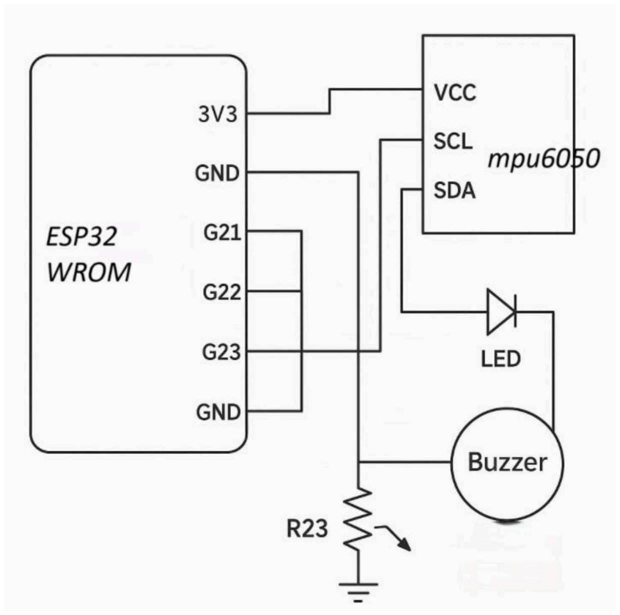
Fall Alert

A fall has been detected

No alerts yet

When someone sends a fall alert, it will appear here

fig- Screenshot of Webpage with Alert.



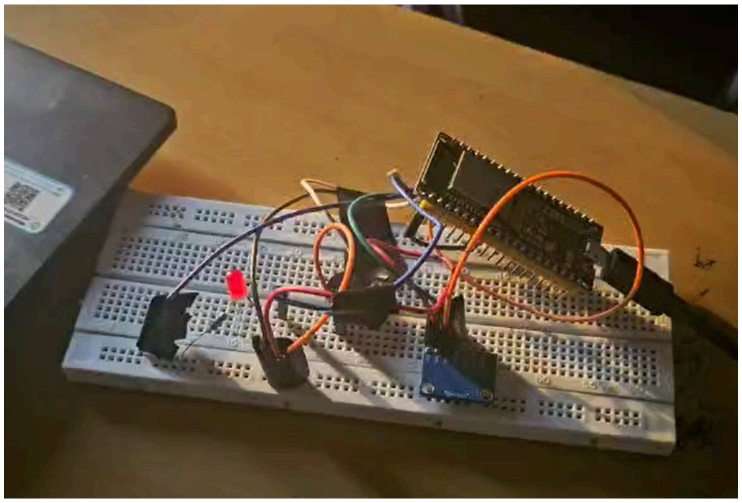


fig- Circuit Diagram of the connections

3. Hardware Components

3.1 ESP32 WROOM Microcontroller

3.2 MPU6050 Accelerometer

Implementation:

- Configured to measure acceleration in the 3 axes
- Mounted for maximum sensitivity to fall-related movements

3.3 Local Alert Mechanisms

- LED Indicator
- Piezoelectric Buzzer

4. Software Implementation

4.1 ESP32 WROM Firmware

4.2 Enhanced Fall Detection Algorithm- Primary trigger: Total acceleration magnitude below 2.5g threshold

4.3 Web Server Implementation

- Multiple HTTP endpoints for data reception and configuration
- Data validation, formatting, and storage
- Telegram integration for notifications

4.4 Telegram Bot Integration

5. Communication Protocols

- Wi-Fi Connectivity with ESP32 WROOM
- HTTP for Web Server Communication
- Telegram API Integration

6. Applications and Use Cases

- Elderly Care
- Medical Patient Monitoring
- Industrial Safety
- Sports and Adventure Activities

7. WROM Architecture Advantages

- **Dedicated Wireless Processing**: Offloads protocol handling from CPU cores
- **Advanced Power Management**: Multiple power modes for different scenarios
- **Enhanced Wireless Performance**: Extended range and robust operation
- **Security Enhancements**: Hardware-accelerated encryption and secure storage

8. Future Improvements

Enhanced Detection Algorithms- Sophisticated machine learning models, personalised detection thresholds based on user profiles

Hardware Enhancements- Alternative wireless protocols (BLE, LoRa, NB-IoT), Additional sensors (barometric pressure, magnetometer)

In conclusion, our IoT-based fall detection system offers an innovative solution to a critical healthcare challenge. Our implementation of Telegram notifications and web dashboard interfaces provides caregivers with immediate alerts and comprehensive monitoring capabilities. The system faced challenges including optimising fall detection accuracy, ensuring reliable connectivity and hardware working. From elderly care and hospital settings to industrial safety and sports monitoring, this project has multiple uses. As our population ages, technology like this becomes increasingly vital.

This project demonstrates how IoT innovation can directly impact quality of life, providing peace of mind for users and their caregivers while potentially saving lives through rapid response to emergencies. The future of healthcare monitoring is here, and it's both wireless.

9. Challenges Faced and Solutions

Throughout the development of our IoT-based fall detection system, we encountered several technical challenges that required innovative solutions:

Hardware Integration Challenges: The calibration of the MPU6050 accelerometer initially produced false positives, with normal movements triggering fall alerts. After extensive testing, we determined an optimal threshold value of 2.5g and implemented a 30-second cooldown period to prevent multiple alerts from a single event.

System Integration Difficulties: Synchronising the three-tier architecture (ESP32 device, Flask server, and Telegram bot) presented significant complexity. Ensuring reliable data flow between these components required developing robust error handling protocols and acknowledgement systems to verify message delivery across all platforms.

Notification System Pivot: Our original design incorporated Gmail for sending alerts to caregivers. However, Gmail's stringent security protocols and authentication requirements proved challenging for real-time IoT integration. We ultimately pivoted to Telegram's API, which offered simpler authentication, more reliable delivery, and additional features, including two-way communication capabilities.

Connectivity Reliability: Maintaining consistent wireless connectivity between the ESP32 and our web server across various network conditions proved difficult. We implemented retry mechanisms and connection status monitoring to ensure alert delivery even in suboptimal network environments.

These challenges provided valuable learning experiences that ultimately strengthened our system design, resulting in a more robust and reliable fall detection solution that effectively bridges hardware sensing with cloud-based notification systems.