A Low-Cost, Real-Time Elderly Fall Detection and Alert System Using an ESP32 Wristband and Companion Android Application

Ayush Andure[1], Arjun Ghadge[2], Arya Patil[3], Aryan Jagtap[4]
[1,2,3,4]*Vishwakarma Institute of Technology*

Abstract—**Falls among the elderly are a major health concern, often leading to serious injury when assistance is delayed. Commercial alert systems can be expensive and lack customization. This paper presents the design, implementation, and refinement of a low-cost, open-source fall detection system consisting of an ESP32-based wristband and a companion Android application. The wristband uses an MPU6050 accelerometer to detect fall events based on a dual-threshold algorithm (free-fall and impact). Upon detecting a fall, it sends an alert via Bluetooth to a paired smartphone. The Android application, built in Kotlin, receives this alert, provides a window for the user to cancel a false alarm, and if uncancelled, automatically sends an SMS notification to pre-registered emergency contacts. The SMS includes the user's last known GPS coordinates and a Google Maps link to expedite assistance. The iterative development process, including debugging of contact management, SMS delivery, and build environment issues, is also discussed. The resulting prototype demonstrates a functional and affordable solution for enhancing the safety of elderly individuals living independently.**

I. INTRODUCTION

As the global population ages, ensuring the safety and well-being of the elderly has become a paramount societal challenge. Falls are a leading cause of fatal and non-fatal injuries among older adults. The critical factor in the outcome of a fall is often the speed at which medical assistance is rendered. Delays can exacerbate injuries and lead to severe complications. While various commercial personal emergency response systems (PERS) exist, they often come with high subscription costs, proprietary hardware, and limited functionality, making them inaccessible to many.

This project addresses the need for an affordable, customizable, and effective solution. We introduce a two-part system: a wearable, ESP32-powered wristband that actively monitors for falls, and a connected Android application that serves as the communication hub for emergency alerts. The system is designed with cost-effectiveness and accessibility in mind, utilizing readily available electronic components and open-source software.

The primary goal is to create a reliable system that can automatically detect a serious fall and notify designated contacts without requiring any action from the user, who may be injured or unconscious. This paper details the hardware and software architecture, the fall detection algorithm, the functionality of the mobile application, and the iterative debugging process that led to a stable prototype. Our work makes the following contributions:

1. Prototype Design: The creation of a low-cost, ESP32-based wearable wristband with an integrated MPU6050 accelerometer for fall detection.
2. System Integration: The development of a companion Android application that pairs with the wristband via Bluetooth, manages emergency contacts pulled from the phone's address book, and dispatches SMS alerts.
3. Enhanced Alerting: Integration of GPS functionality to include the user's precise location and a Google Maps link in the emergency SMS, significantly reducing response time for caregivers and emergency services.
4. Problem Identification and Refinement: A detailed account of the iterative development process, including overcoming challenges in Android permissions, data persistence, and build environment inconsistencies, resulting in a stable and functional application.

II. LITERATURE SURVEY (RELATED WORK)

Domain / Topic	Key Findings & Prior Art	Limitations of Prior Art	Relevance to This Project
Ambient & Vision-Based Detection	Early systems used cameras, microphones, or pressure sensors installed in a living space to monitor for falls.	Limited to a fixed location (e.g., one room), raises significant privacy concerns, and can be very expensive to install and maintain.	Our project adopts a wearable-based approach specifically to overcome these limitations, providing continuous monitoring wherever the user goes.
Wearable Accelerometer Sensors	The use of MEMS accelerometers in wearable devices is the most common and accepted method for fall detection. Research papers [1, 2] validate this approach.	The primary challenge is distinguishing real falls from "fall-like" Activities of Daily Living (ADLs) like sitting down quickly or dropping the device.	Our project is built on this established foundation, using a standard MPU6050 accelerometer, which is proven to be effective for this application.
Fall Detection Algorithms	1. Threshold-Based: Simple, computationally inexpensive algorithms that detect a period of weightlessness (free-fall) followed by a sharp impact spike. This is a well-documented method [2]. 2. Machine Learning (ML): More advanced systems use ML models to analyze sensor data for higher accuracy and fewer false positives [3].	Threshold-Based: Prone to false positives from vigorous movements. Machine Learning: Often requires significant processing power and large datasets for training, making them complex to implement on low-cost microcontrollers.	Our system implements a robust dual-threshold algorithm, which is perfectly suited for the ESP32's processing capabilities and provides a reliable balance between accuracy and low cost.
Hardware Platforms	1. High-End Commercial: Devices like the Apple Watch integrate fall detection with a suite of other health features using powerful, custom processors. 2. Low-Cost Microcontrollers: Open-source projects frequently use platforms like Arduino or ESP32 due to their affordability and built-in features [4].	High-End: Very expensive, proprietary ecosystems, and often require subscriptions for full functionality. Low-Cost: Can have limitations in processing power and battery management if not carefully programmed.	Our project specifically targets the low-cost category, using the ESP32 to create an affordable and accessible alternative to expensive commercial options.
Notification & Alert Systems	Commercial systems typically link to a dedicated call center via a cellular connection, often for a monthly fee. Other projects use Bluetooth to connect to a smartphone app.	Call center models are expensive. Basic smartphone apps in other projects often lack enhanced features like precise location tracking or automated follow-ups.	Our system improves upon the smartphone model by creating a full-featured companion app that manages contacts, provides a cancellation window, and critically, enriches the SMS alert with GPS coordinates and

III. METHODOLOGY AND SYSTEM DESIGN

A. Overview

The system operates as a two-component client-server model. The ESP32 wristband acts as the client, continuously monitoring the user's movement. The user's Android smartphone acts as the server, passively listening for a signal.

- Step 1: Detection. The wristband's MPU6050 accelerometer data is analyzed in real-time. If the data crosses the pre-defined thresholds for a fall, the wristband enters an alert state.

- Step 2: Communication. The ESP32 immediately sends a "FALL DETECTED" message via a pre-established Bluetooth Serial Port Profile (SPP) connection to the paired Android phone.
- Step 3: App Response. The Android app receives the message, triggers a visual and audible alarm on the phone, and starts a 10-second countdown. This allows the user to cancel the alert from the app if it's a false alarm.
- Step 4: Notification. If the alarm is not canceled, the app retrieves the phone's last known GPS coordinates and sends a formatted SMS message to all emergency contacts stored in the app. A follow-up message is sent after 60 seconds.

B. Hardware Components

Our prototype consists of the following off-the-shelf components, chosen for their low cost, wide availability, and strong community support.

Component	Purpose
ESP32 Dev Board	The core microcontroller. Chosen for its integrated Bluetooth, Wi-Fi capabilities, and sufficient processing power for the detection algorithm.
MPU6050	A 6-axis motion tracking device (3-axis accelerometer, 3-axis gyroscope). Used to measure acceleration and detect the patterns of a fall.
Buzzer	Provides audible feedback on the wristband itself, confirming a detected fall and signaling the start of the alert process.
Push Button	A physical button on the wristband enclosure allowing the user to manually cancel a false alarm directly, in addition to the app-based cancel.

C. Software Implementation

The software is divided into two distinct parts: the embedded firmware for the ESP32 wristband and the Android application.

1) ESP32 Firmware (Arduino C++)

The firmware is responsible for reading sensor data, implementing the fall detection logic, and communicating with the phone.

1. Sensor Initialization: The Adafruit_MPU6050 library is used to interface with the accelerometer over I2C. The accelerometer range is set to 8G to capture the high forces of an impact.
2. Fall Detection Algorithm: The core logic runs in the main loop. It calculates the total magnitude of acceleration from the X, Y, and Z axes.
   - Free-fall Detection: If the acceleration magnitude drops below FREE_FALL_THRESHOLD_G (e.g., 0.6g), the system flags a potential free-fall and records the timestamp.
   - Impact Detection: If, within a specified IMPACT_WINDOW_MS (e.g., 1200ms) after the free-fall, the acceleration magnitude exceeds the IMPACT_THRESHOLD_G (e.g., 2.5g), a fall is confirmed.
3. Bluetooth Communication: The BluetoothSerial library is used to create a serial communication link. Upon a confirmed fall, the string "FALL DETECTED" is sent to the paired device. The firmware also listens for a "CANCEL_ALARM" message from the app to stop the local buzzer.

2) Android Application (Kotlin)

The application serves as the user interface and the alert dispatch system.

1. Permissions: The app requests a comprehensive set of permissions on startup, including Bluetooth (Scan and Connect), SMS (Send), Location (Fine and Coarse), and Read Contacts.
2. Bluetooth Handling: The app scans for paired devices matching the ESP32's name ("ESP32-FallBand") and establishes a connection. A dedicated thread listens for incoming data from the wristband.
3. Contact Management: Users can add emergency contacts by tapping a button that launches the phone's native contact picker. The selected contact's name and number are retrieved, stored in a Contact data class, and saved persistently to the phone's Shared Preferences as a JSON string. This

ensures the contact list is available even after the app restarts. The list is displayed in the UI and updates immediately upon adding or deleting a contact.

4. Alert Logic: Upon receiving the "FALL DETECTED" message, the app displays a prominent alert and starts a 10-second timer. If this timer completes, the send Emergency Sms function is called.
5. SMS and Location: The Fused Location Provider Client is used to get the phone's last location. This location is formatted into a Google Maps URL and appended to a pre-defined
6. alert message. The Sms Manager is then used to send this message to every contact in the emergency list. A robust delivery confirmation system using a Broad cast Receiver provides feedback on the success or failure of the SMS transmission.

IV. RESULTS AND DISCUSSION

The integrated system was tested to validate the functionality of each component. Simulated fall events were created by dropping the wristband from a height of approximately one meter onto a cushioned surface.

- The wristband successfully and consistently identified the simulated fall events, triggering its local buzzer.
- The "FALL DETECTED" message was reliably transmitted via Bluetooth to the companion app within a range of approximately 10 meters.
- The Android application correctly displayed the alarm screen and initiated the countdown upon receiving the message.
- When the alarm was not cancelled, the application successfully retrieved GPS coordinates and sent the formatted SMS to the test emergency contacts. Delivery was confirmed by the Broad cast Receiver feedback system.
- The contact management system proved stable. Contacts selected from the phone's address book were added to the UI, saved correctly, persisted after an app restart, and were read correctly by the SMS function

Limitation	Cause/Observation	Proposed Solution
False Positives	Dropping the device or sitting down very quickly could occasionally trigger a false alarm.	Implement a more advanced algorithm, potentially a machine learning model, to better distinguish falls from normal activities. Incorporate gyroscope data to check for changes in orientation.
Battery Life	Continuous Bluetooth connection and sensor polling limit the wristband's battery life.	Implement low-power modes, where the ESP32 sleeps and wakes periodically to check the sensor. Use Bluetooth Low Energy (BLE) instead of Bluetooth Classic.
Phone Proximity	The system is entirely dependent on the phone being within Bluetooth range and having a network connection.	Integrate a GSM/LTE module directly into the wristband for a standalone device, though this would increase cost and size.
User Interface	The current Android UI is functional but basic.	Develop a more polished user interface with features like contact editing, a history of alerts, and battery status of the wristband.

V. FUTURE SCOPE

This project serves as a strong foundation for a more advanced personal safety system. Future work will focus on three key areas:

1. Enhanced Detection Accuracy: The next iteration will involve collecting real-world accelerometer data of falls and daily activities. This data will be

used to train a lightweight machine learning model (e.g., using TensorFlow Lite for Microcontrollers) to run directly on the ESP32. This would dramatically reduce false positives and improve the reliability of the detection.

2. Hardware Refinement: The prototype will be miniaturized into a custom PCB and housed in a 3D-printed, waterproof (IP67) enclosure. Power management will be improved with the inclusion of a LiPo battery charging circuit and optimization of the firmware to use the ESP32's deep sleep modes, extending battery life from hours to days.

3. Cloud and Expanded Software Features: A cloud-based backend (e.g., using Firebase) will be developed. This would allow for a history of fall events to be logged, enable remote monitoring by family members through a web dashboard, and could trigger alternative notifications like automated voice calls (using a service like Twilio) or push notifications to other family members' apps if an SMS fails.

VI. CONCLUSION

This paper has detailed the successful design and implementation of a low-cost, IoT-based elderly fall detection system. By integrating an ESP32-based wearable with a feature-rich Android application, the project provides a functional and affordable alternative to expensive commercial services. The system reliably detects simulated falls, leverages the user's smartphone for powerful notification capabilities, and enhances alerts with critical GPS location data. The extensive debugging and refinement process has resulted in a stable prototype that effectively addresses the core requirements of such a safety device. While there is significant scope for future improvements, particularly in detection accuracy and hardware miniaturization, this project serves as a validated proof-of-concept and a scalable platform for future development in the field of personal safety technology.

VII. ACKNOWLEDGEMENT

We would like to express our gratitude to the open-source community, whose libraries and frameworks for the Arduino and Android platforms were instrumental in the development of this project. Their work provided the foundation upon which this system was built.

REFERENCES

[1] N. Noury, A. Fleury, P. Rumeau, A. Bourke, G. ÓLaighin, V. Rialle, and J. Lundy, "Fall
[2] detection–principles and methods," in Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007, pp. 1663–1666.
[3] A. K. Bourke, J. V. O'Brien, and G. M. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," Gait & Posture, vol. 26, no. 2, pp. 194–199, 2007.
[4] S. Chaudhuri, S. K. Gupta, S. Kumar, and S. C. Mishra, "A machine learning based fall detection system using wearable sensors," in 2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS), 2018, pp. 1-5.
[5] Espressif Systems, "ESP32 Series Datasheet," 2023. [Online]. Available: