A Project Report on

FALL DETECTION USING BODY AREA NETWORK

Project Report submitted in partial fulfilment of the requirement for the 8th Semester of
Bachelors of Technology in Information Technology under
University of Calcutta

Submitted By

Name: Indranil Kundu
Roll No.:- T91/IT/216005
Registration No.:- 115-1111-0431-19

&

Name: Kunal Pal
Roll No.:- T91/IT/216006
Registration No.:- 211-1111-0347-19

Under the Supervision of
JOYATI MONDAL
Assistant Professor,

A.K. Choudhury School of Information Technology
UNIVERSITY OF CALCUTTA

www.caluniv.ac.in

A.K. CHOUDHURY SCHOOL OF INFORMATION TECHNOLOGY

UNIVERSITY OF CALCUTTA

University of Calcutta, JD-2 Sector-III, Saltlake, Kolkata-700106, INDIA

CERTIFICATE

This is to certify that the project synopsis entitled **"FALL DETECTION USING BODY AREA NETWORK"** submitted for partial fulfilment of the requirements of 8th Semester of Bachelors of Technology under University of Calcutta, has been carried out by **Indranil Kundu(Roll No- T91/IT/216005 and Registration No- 115-1111-0431-19) & Kunal Pal(Roll No- T91/IT/216006 and Registration No- 211-1111-0347-19)** under the supervision of **Joyati Mondal, Assistant Professor.**

_____ _____

CHAIRMAN, BOARD OF EXAMINER PROJECT SUPERVISOR

_____

EXTERNAL EXAMINER(S)

ACKNOWLEDGEMENT

We wish to express our profound sense of gratitude to our project supervisor, Joyati Mondal, Assistant Professor, for her unwavering support, inspiration, and guidance. She has introduced us to various perspectives for approaching a problem, and we have learned from her that persistent effort is essential to achieve our goals. I am immensely grateful to her for devoting her valuable time, for her constant advice, and for the insightful discussions that helped shape our project. It has been a truly enriching learning experience to work under her mentorship.

We are also thankful to our department, A.K. Choudhury School of Information Technology, University of Calcutta, for providing the necessary infrastructure and resources for carrying out this project. We would like to extend our heartfelt thanks to Dola Bhattacharjee Gupta, DST Women Scientist at the University of Calcutta, whose extensive support and guidance played a pivotal role in the successful completion of our project. Her constant encouragement, technical insights, and willingness to help at every step made a significant difference in our work.

We are also grateful to Prof. Amlan Chakrabarti, Professor and Director, A.K. Choudhury School of IT, University of Calcutta, for his valuable suggestions. His critical feedback helped us identify and rectify important aspects of our project, ultimately improving its quality. Lastly, we express our sincere appreciation to our seniors, friends, and everyone who directly or indirectly offered their guidance and support throughout the course of this project. Their encouragement has been a continuous source of motivation.

THANK YOU

Date: _____ _____

Indranil Kundu AKCSIT, University of Calcutta

_____

Kunal Pal AKCSIT, University of Calcutta

ABSTRACT

The Fall Detection and Real-Time Health Monitoring System is a wearable solution designed to continuously monitor the user's vital health metrics and detect emergency events such as accidental falls. This system integrates multiple biomedical and motion sensors—including heart rate, temperature, and accelerometer modules—into a compact wristband controlled by NodeMCU (ESP8266). The collected data is wirelessly transmitted in real-time to a cloud backend via the ThingSpeak IoT platform.

A binary classification machine learning model, trained on over 370,000 labelled data points, is used to distinguish between fall and non-fall events with high precision. In the event of a detected fall, the system captures the user's GPS coordinates and triggers an automated alert through email, which also includes a list of nearby hospitals based on location.

The entire ecosystem is designed for seamless cloud integration and supports future features such as AI-based health metric interpretation using Bio-GPT and a Flutter-based mobile application for real-time visualization. The robust architecture makes this system scalable, portable, and ideal for monitoring elderly or health-vulnerable individuals.

SYNOPSIS

The Fall Detection and Real-Time Health Monitoring System is a wearable, IoT-based solution developed to monitor vital health parameters and detect fall events using embedded sensors and machine learning. The system is aimed at addressing the growing need for continuous and non-intrusive healthcare monitoring, particularly for elderly individuals and patients with high health risk.

This wrist-worn device integrates various sensors—including the MAX30100 for heart rate and SpO_2, DS18B20 for body temperature, and MPU6050 for motion detection—interfaced through a NodeMCU (ESP8266) microcontroller. The collected sensor data is transmitted to the ThingSpeak IoT platform via Wi-Fi for real-time monitoring and analytics.

A binary classification machine learning model has been trained on over 370,000 data points to accurately detect fall events. Upon detecting a fall, the system retrieves GPS coordinates using the Neo-6M module and automatically triggers an alert via email, which includes the user's location and nearby hospital information, ensuring a quick emergency response.

The backend infrastructure is deployed on a cloud platform (Render), allowing for scalable and efficient data processing. The system also includes future provisions for AI-based health interpretation using Bio-GPT and a Flutter-based mobile dashboard for user interaction.

The proposed system demonstrates an effective combination of hardware, software, cloud services, and artificial intelligence to deliver a low-cost, real-time, and reliable health monitoring solution. It holds significant potential for integration into telemedicine platforms and smart healthcare ecosystems.

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS / NOTATIONS

Abbreviation	Full Form
IoT	Internet of Things
ML	Machine Learning
GPS	Global Positioning System
API	Application Programming Interface
SpO_2	Peripheral Capillary Oxygen Saturation
SMTP	Simple Mail Transfer Protocol
JSON	JavaScript Object Notation
TNR	Times New Roman (used font)
IMU	Inertial Measurement Unit
HTTP	Hypertext Transfer Protocol

Notations and Symbols:

Symbol	Meaning
μ	Mean value
σ	Standard deviation
HR	Heart rate
bpm	Beats per minute
°C	Degree Celsius

TABLE OF CONTENTS

3. CHAPTER 3 : Implementation

4. CHAPTER 4 : Results and Discussions

5. CHAPTER 5 : Conclusion and Future scope

CHAPTER 1

INTRODUCTION

1.1 Motivation

With the growing number of elderly individuals living independently, there is an urgent need for technologies that ensure their safety and well-being. Accidental falls remain a leading cause of injury and even mortality among senior citizens, particularly in the absence of immediate assistance. Traditional healthcare systems are often reactive and do not provide real-time, continuous health monitoring. Motivated by this critical gap, this project aims to develop a wearable solution that not only monitors vital parameters in real time but also detects fall events and sends out alerts instantly—empowering caregivers to respond without delay.

1.2 Objectives

The key objectives of this project are as follows:

• To develop a compact, wrist-worn device using NodeMCU and embedded sensors for real-time health data acquisition.

• To implement a machine learning model for accurate binary classification of fall vs. non-fall events.

• To integrate GPS modules for precise location tracking in emergencies.

• To implement an automated alert system using SMTP email services, including nearby hospital identification.

• To design and deploy a mobile-friendly health dashboard using Flutter.

• To utilize cloud platforms (e.g., Render) for scalable data processing and backend services.

• To integrate GenAI for translating raw health metrics into user-friendly health insights.

1.3 Problem Statement

Elderly individuals who live alone are highly vulnerable during medical emergencies, especially when accidental falls occur. The lack of real-time monitoring and delay in emergency response can lead to severe health consequences. Despite advances in health wearables, existing solutions are often expensive, lack AI-powered insights, and fail to provide immediate contextual alerts to caregivers. This project addresses these shortcomings by delivering a reliable, affordable, and scalable real-time health monitoring and fall detection system.

1.4 Scope of the Project

This project brings together the power of IoT, machine learning, GPS, and cloud computing to build an end-to-end health monitoring solution. The hardware is modular and supports easy addition or replacement of sensors. The software stack is cloud-enabled and API-integrated, allowing for future enhancements such as telemedicine integration or multi-user family dashboards. The scope also includes real-time alerting, mobile data visualization, and health data translation through AI—all within a single, unified ecosystem.

1.5 Contribution of the Project

This project contributes an end-to-end system that combines embedded systems, machine learning, cloud infrastructure, and mobile development to solve a real-world healthcare problem. Key contributions include:

• A fully functional wearable device integrating heart rate, body temperature, accelerometer, and GPS sensors.

• Real-time email alerts with GPS location and hospital mapping.

• A mobile app built using Flutter for health metric visualization and user interaction.

• AI-powered health data translation using Bio-GPT for making metrics easier to interpret.

• Complete cloud-based deployment (via Render) for backend ML inference and alert processing.

CHAPTER 2

SYSTEM DESIGN

2.1 System Architecture Overview

The system is designed as a modular, scalable architecture that seamlessly integrates hardware, cloud services, machine learning models, and mobile applications. The core idea is to collect real-time physiological and motion data using sensors, transmit it via Wi-Fi to a cloud platform, analyse it using a pre-trained ML model, and alert emergency contacts in case of a fall. Additionally, a mobile dashboard provides a live view of the user's health metrics.

The architecture consists of four major layers:

• **Hardware Layer** – A wearable wristband embedded with sensors and NodeMCU.

• **IoT Platform Layer** – Wi-Fi-based data transmission to ThingSpeak.

• **Cloud Processing Layer** – Cloud backend hosting the ML model and alert engine.

• **Application/UI Layer** – A Flutter-based mobile app for real-time monitoring.

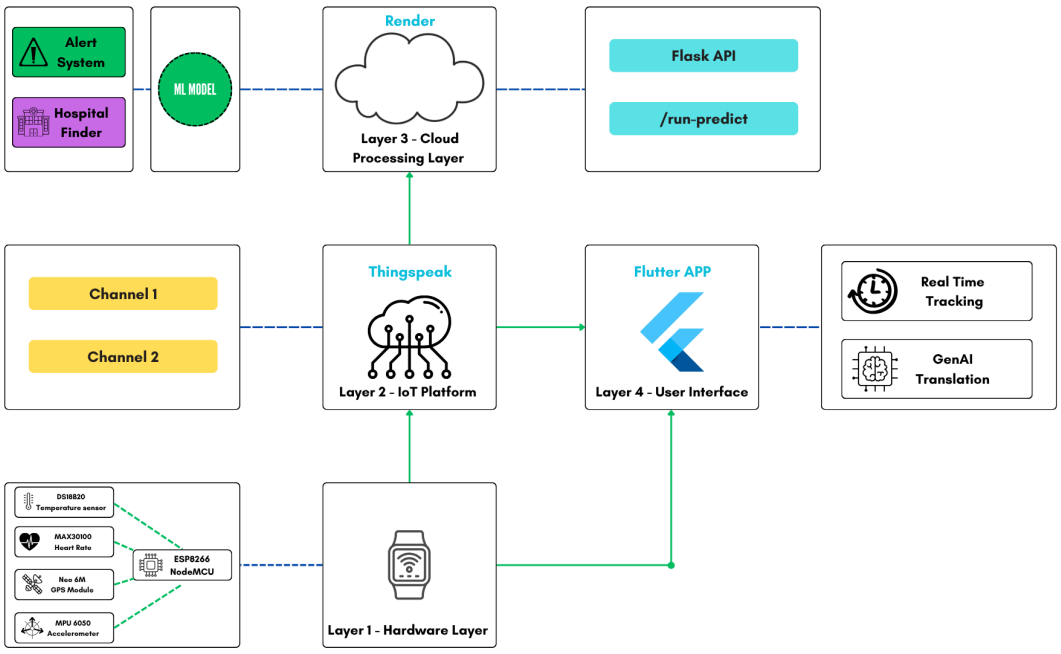


Fig 1.1

2.2 Hardware Design

2.2.1 Sensor Details and Specifications

The wristband includes the following sensors:

Sensor	Purpose
MAX30100	Measures heart rate and SpO_2 levels
DS18B20	Captures body temperature
MPU6050	Detects motion using a 6-axis IMU
Neo 6M GPS	Tracks geographical coordinates

Each sensor is selected for its low power consumption, compact design, and compatibility with NodeMCU (ESP8266).

2.2.2 Pin Mapping and Wiring

The components are wired to NodeMCU as follows:

Sensor	Connected Pins
MAX30100	SDA – D2, SCL – D1, VCC – 3.3V, GND – GND
DS18B20	Data – D5, VCC – 3.3V, GND – GND
MPU6050	Data – D5, VCC – 3.3V, GND – GND
Neo 6M GPS	RX – D6, TX – D7

A common ground is maintained across all components. The wristband form factor was 3D printed to hold the circuit securely.

2.3 Data Pipeline Architecture

The data flow begins from the sensors, which continuously push readings to the NodeMCU. The microcontroller sends this data at regular intervals to the ThingSpeak cloud using HTTP. ThingSpeak acts as a buffer and visualization layer while also triggering events via APIs.

Key components in the pipeline:

• **NodeMCU:** Gathers and formats sensor data

• **ThingSpeak:** Collects, stores, and displays real-time data

• **Render Cloud Platform:** Hosts the ML model and handles fall detection & alerting via REST APIs

• **Flutter APP:** Displays data from Thingspeak and GenAI Translation.

2.4 Cloud and API Design

The cloud backend is deployed on Render and performs the following functions:

• **/api/fall-detection:** Accepts sensor data payload and returns fall prediction

• **/api/send-alert:** Sends email alerts with GPS and nearest hospitals using SMTP and Google Maps API

The API is secured using environment-based credentials. The ML model (model.pkl) and standard scaler (scaler.pkl) are loaded at startup and kept in memory for fast prediction. The alert module uses SMTP with a configured Gmail account for outgoing messages.

CHAPTER 3

IMPLEMENTATION

3.1 Sensor Integration and Calibration

The implementation began with the integration of multiple biomedical and motion sensors with the NodeMCU ESP8266 microcontroller. The sensors include MAX30100 (heart rate and SpO_2), DS18B20 (body temperature), MPU6050 (accelerometer + gyroscope), and Neo 6M (GPS). Each sensor was interfaced using I²C or digital pins as per their specifications.

Calibration routines were developed in the Arduino IDE to ensure accurate sensor readings. For example:

• MAX30100 readings were filtered to remove noise due to hand motion.

• DS18B20 was validated using reference thermometers.

• MPU6050 output was scaled and normalized to detect orientation and sudden motion.

• GPS data was parsed using TinyGPS++ library for accurate latitude and longitude.

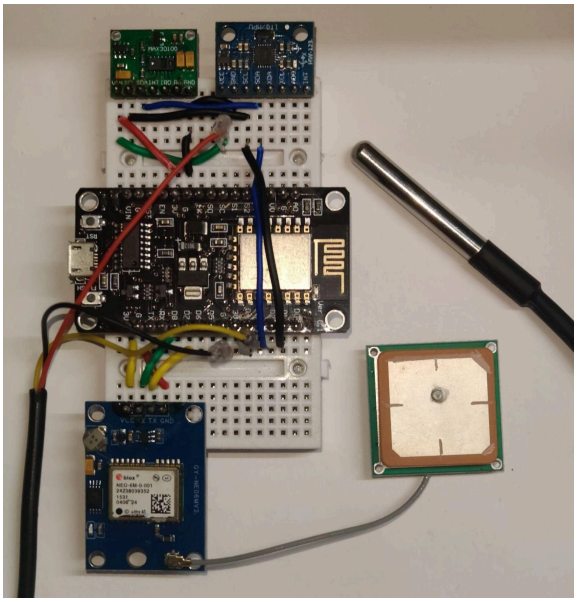


Fig 2.1

3.2 Firmware Development (NodeMCU)

The NodeMCU was programmed using Arduino C++. The code was modularized to handle:

• Sensor initialization and reading

• Wi-Fi connectivity

• Data formatting into JSON

• HTTP POST requests to ThingSpeak

• Conditional triggers for fall detection alerts

Interrupts were used for real-time motion detection from the MPU6050, and millis() was used for timing operations instead of delay() to ensure non-blocking execution.

Library Used :

Wire.h : I2C Communication

MAX30100_PulseOximeter.h : Heart Rate and Blood Oxygen Monitoring

ESP8266WiFi.h : WiFi Connectivity for ESP8266

ESP8266HTTPClient.h : Making HTTP Requests

WiFiClient.h : WiFi Client Operations

time.h : Time Functions and NTP Synchronization

OneWire.h : OneWire Communication Protocol

DallasTemperature.h : Temperature Sensor Control

SoftwareSerial.h : Additional Serial Communication Ports

TinyGPS++.h : GPS Data Parsing and Location Tracking

MPU6050.h : 6-Axis Motion Sensor (Accelerometer + Gyroscope)

3.3 ML Model Training and Deployment

3.3.1 Dataset Summary

The ML model was trained using a dataset of over 370,000 data points collected from actual and simulated fall scenarios. The data included:

time : Timestamp of data collection (milliseconds or seconds)

ax : X-axis of accelerometer signal (g-force units)

ay : Y-axis of accelerometer signal (g-force units)

az : Z-axis of accelerometer signal (g-force units)

w : Quaternion W component of gyroscope orientation

x : Quaternion X component of gyroscope orientation

y : Quaternion Y component of gyroscope orientation

z : Quaternion Z component of gyroscope orientation

droll : Angular velocity of gyroscope (roll rotation rate)

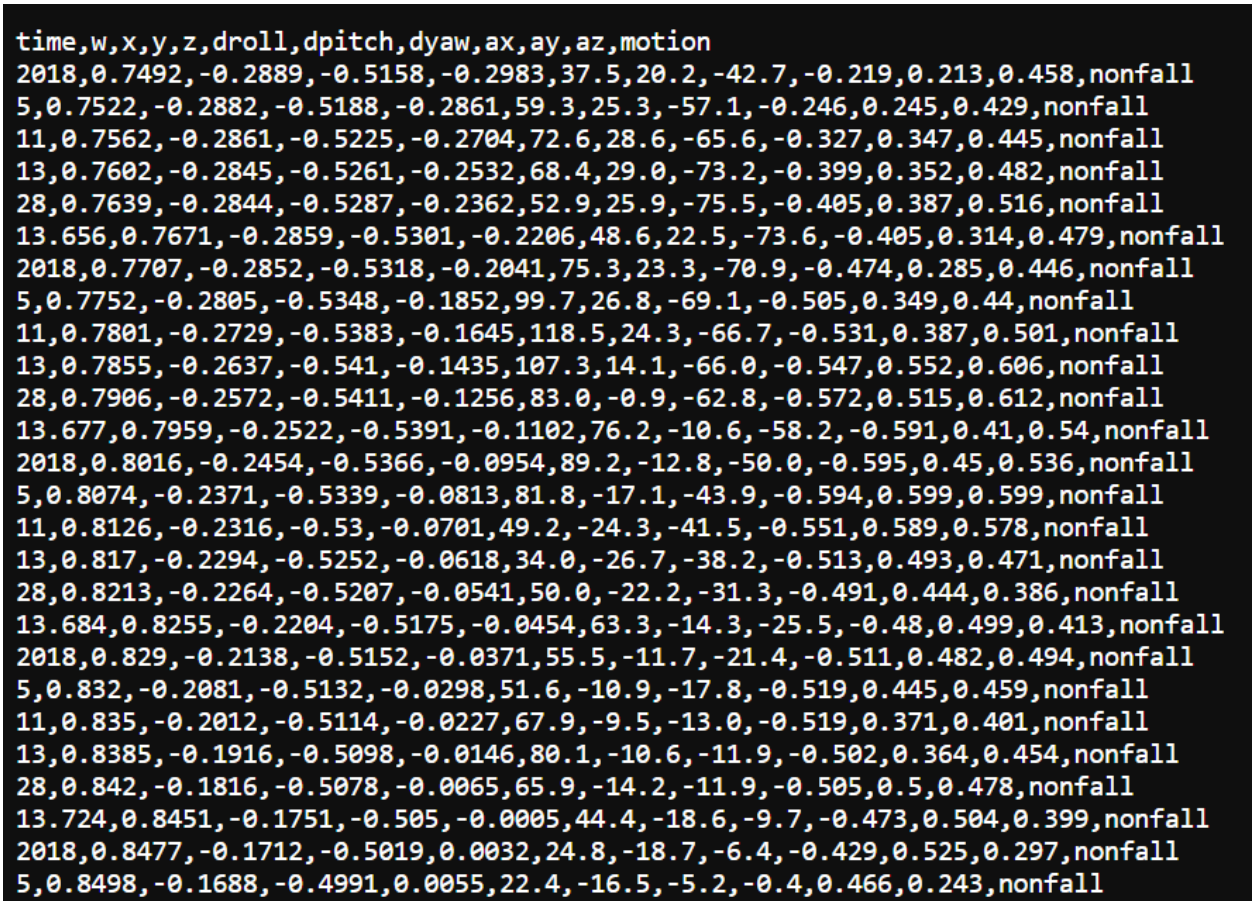
dpitch : Angular velocity of gyroscope (pitch rotation rate)

dyaw : Angular velocity of gyroscope (yaw rotation rate)

motion : Motion classification label (fall or non-fall)

Each sample was labelled as either fall or non-fall, forming a binary classification problem.

https://github.com/Kunal70616c/Fall-Detection-and-Alert-System-Using-Body-Area-Network/blob/main/dataset/fall_dataset.csv

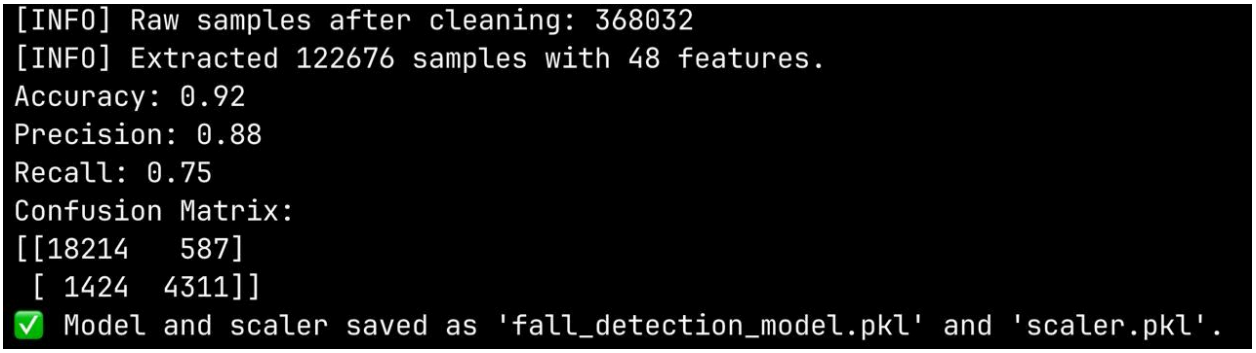


```
time,w,x,y,z,droll,dpitch,dyaw,ax,ay,az,motion
2018,0.7492,-0.2889,-0.5158,-0.2983,37.5,20.2,-42.7,-0.219,0.213,0.458,nonfall
5,0.7522,-0.2882,-0.5188,-0.2861,59.3,25.3,-57.1,-0.246,0.245,0.429,nonfall
11,0.7562,-0.2861,-0.5225,-0.2704,72.6,28.6,-65.6,-0.327,0.347,0.445,nonfall
13,0.7602,-0.2845,-0.5261,-0.2532,68.4,29.0,-73.2,-0.399,0.352,0.482,nonfall
28,0.7639,-0.2844,-0.5287,-0.2362,52.9,25.9,-75.5,-0.405,0.387,0.516,nonfall
13.656,0.7671,-0.2859,-0.5301,-0.2206,48.6,22.5,-73.6,-0.405,0.314,0.479,nonfall
2018,0.7707,-0.2852,-0.5318,-0.2041,75.3,23.3,-70.9,-0.474,0.285,0.446,nonfall
5,0.7752,-0.2805,-0.5348,-0.1852,99.7,26.8,-69.1,-0.505,0.349,0.44,nonfall
11,0.7801,-0.2729,-0.5383,-0.1645,118.5,24.3,-66.7,-0.531,0.387,0.501,nonfall
13,0.7855,-0.2637,-0.541,-0.1435,107.3,14.1,-66.0,-0.547,0.552,0.606,nonfall
28,0.7906,-0.2572,-0.5411,-0.1256,83.0,-0.9,-62.8,-0.572,0.515,0.612,nonfall
13.677,0.7959,-0.2522,-0.5391,-0.1102,76.2,-10.6,-58.2,-0.591,0.41,0.54,nonfall
2018,0.8016,-0.2454,-0.5366,-0.0954,89.2,-12.8,-50.0,-0.595,0.45,0.536,nonfall
5,0.8074,-0.2371,-0.5339,-0.0813,81.8,-17.1,-43.9,-0.594,0.599,0.599,nonfall
11,0.8126,-0.2316,-0.53,-0.0701,49.2,-24.3,-41.5,-0.551,0.589,0.578,nonfall
13,0.817,-0.2294,-0.5252,-0.0618,34.0,-26.7,-38.2,-0.513,0.493,0.471,nonfall
28,0.8213,-0.2264,-0.5207,-0.0541,50.0,-22.2,-31.3,-0.491,0.444,0.386,nonfall
13.684,0.8255,-0.2204,-0.5175,-0.0454,63.3,-14.3,-25.5,-0.48,0.499,0.413,nonfall
2018,0.829,-0.2138,-0.5152,-0.0371,55.5,-11.7,-21.4,-0.511,0.482,0.494,nonfall
5,0.832,-0.2081,-0.5132,-0.0298,51.6,-10.9,-17.8,-0.519,0.445,0.459,nonfall
11,0.835,-0.2012,-0.5114,-0.0227,67.9,-9.5,-13.0,-0.519,0.371,0.401,nonfall
13,0.8385,-0.1916,-0.5098,-0.0146,80.1,-10.6,-11.9,-0.502,0.364,0.454,nonfall
28,0.842,-0.1816,-0.5078,-0.0065,65.9,-14.2,-11.9,-0.505,0.5,0.478,nonfall
13.724,0.8451,-0.1751,-0.505,-0.0005,44.4,-18.6,-9.7,-0.473,0.504,0.399,nonfall
2018,0.8477,-0.1712,-0.5019,0.0032,24.8,-18.7,-6.4,-0.429,0.525,0.297,nonfall
5,0.8498,-0.1688,-0.4991,0.0055,22.4,-16.5,-5.2,-0.4,0.466,0.243,nonfall
```

Fig 3.1

3.3.2 Model Structure and Accuracy

A Random Forest Classifier was selected for its robustness and low latency. The model achieved over 92% accuracy during cross-validation. The final model and scaler were saved using joblib as model.pkl and scaler.pkl.



```
[INFO] Raw samples after cleaning: 368032
[INFO] Extracted 122676 samples with 48 features.
Accuracy: 0.92
Precision: 0.88
Recall: 0.75
Confusion Matrix:
[[18214   587]
 [ 1424  4311]]
✅ Model and scaler saved as 'fall_detection_model.pkl' and 'scaler.pkl'.
```

Fig 3.2

3.4 Cloud Deployment and Alert System Integration

3.4.1 Fall Detection Cloud Deployment

- **Backend:** Flask API hosted on Render.
- **Prediction:** Fetches live sensor data from ThingSpeak, runs ML model, detects fall.
- **Alerts:** Sends email with location & nearby hospitals.
- **Security:** /run-prediction route protected with a token (?token=...).
- **Test:** Use Postman or browser with the correct token to trigger prediction.

3.4.2 Deployment Process (Render):

1. **Create Flask App** – Main logic in app.py, routes, and ML prediction.
2. **Prepare File**s – Include requirements.txt, Procfile, and all modules.
3. **Push to GitHub –** Push full project code to a public or private GitHub repo.
4. **Deploy on Render** –Create new Web Service on render.com.
5. Link GitHub repo.
6. **Set build command:** pip install -r requirements.txt
7. **Set start command:** gunicorn app:app
8. **Set Environment Variables** – Add API_SECRET_TOKEN in Render's environment settings.
9. **Access API –** Use the Render URL with token to trigger the fall detection.

3.4.3 Alert System Integration :

1. Once a fall is detected by the backend API, the system:
    a. Retrieves current GPS coordinates from the Neo 6M module
    b. Uses the Google Maps API to find nearby hospitals within a 5 km radius
2. Sends an email alert to predefined emergency contacts using SMTP (Gmail API).
    a. Fall status with Location (latitude & longitude) and Google Maps link, List of nearby hospitals.
    b. The alert system is fully automated and does not require user intervention.

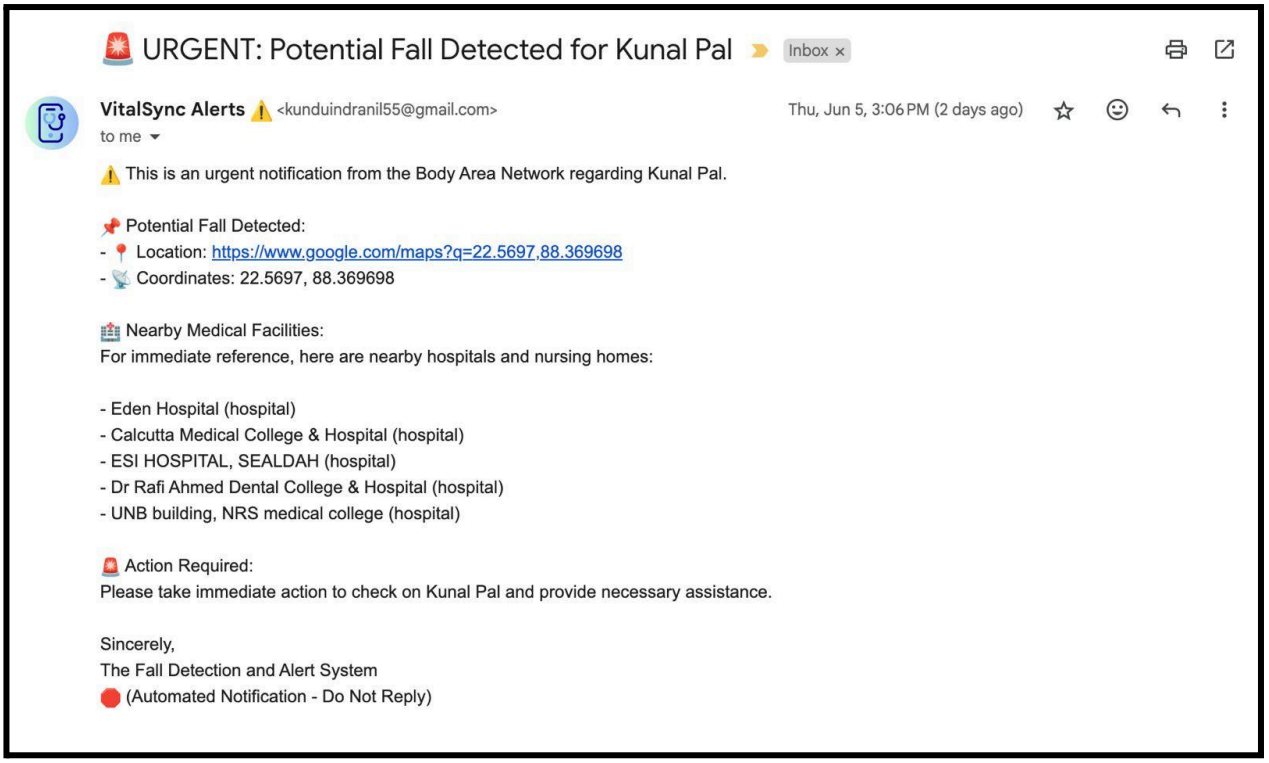


Fig 3.3

3.5 API Endpoint Structure And Response type

The system uses RESTful APIs deployed on Render:

Endpoint	Method	Function
/api/fall-detection	POST	Predict fall from sensor data
/api/send-alert	POST	Send emergency email alert

All endpoints accept or return JSON objects and include error handling mechanisms for invalid or missing data.

Response (type : JSON) :

Fall Detected	NO - Fall Detected

{
"status": "success",
"fall_detected": true,
"location": {
"latitude": 22.5726,
"longitude": 88.3639
}
}
	// When no fall is detected
	{
	"status": "success",
	"fall_detected": false
	}

Cloud Structure :

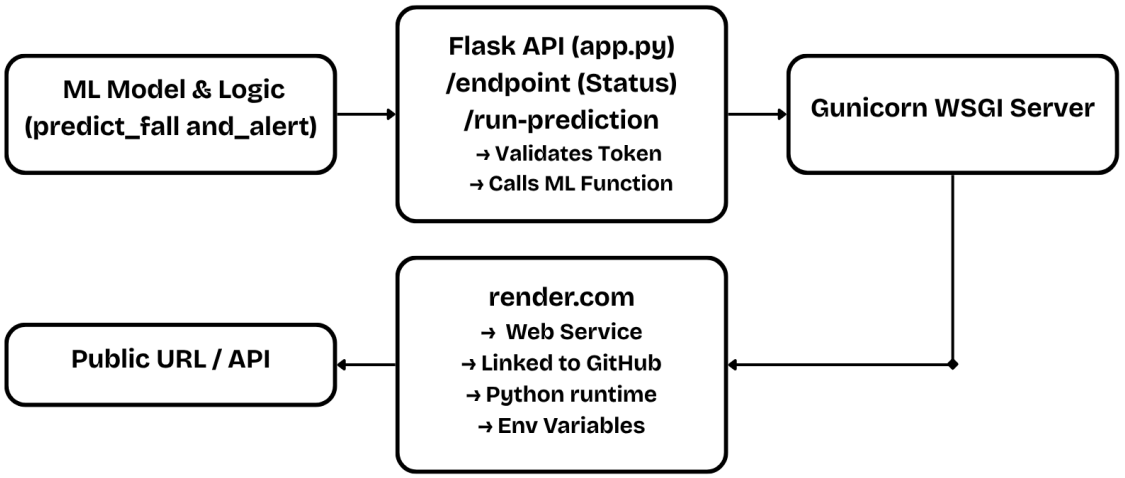


Fig 3.4

3.6 GenAI Health Data Translation

VitalSync integrates **Google Gemini AI** (gemini-2.0-flash) to transform raw health metrics into understandable insights.

The AI model:

- **Translates vital data** (heart rate, temperature, BMI, user profile) into human-readable explanations.
- **Provides personalized advice**, offering general tips for normal readings, basic suggestions for slight deviations, and urgent advisories for abnormal values.
- Delivers all insights as **clean, plain text**.

3.7 Flutter Mobile App Development

VitalSync is a Flutter mobile app that provides a personalized health dashboard. It connects your **real-time vital signs** from ThingSpeak with **AI-powered health insights** from Google Gemini.

Key Features:

- **User Setup:** Easy input of personal details (name, age, weight, height) using intuitive pickers.
- **Device Pairing:** Securely links to a hardware device via **QR code scanning** with built-in device ID verification.
- **Live Monitoring:** Fetches **heart rate and temperature** from ThingSpeak, with BMI calculated, updating every **15 seconds**.
- **AI Insights:** Uses Google Gemini to translate raw data into **plain-text, actionable health advice**, including warnings for critical values.
- **Connectivity:** Monitors internet status and displays alerts, pausing/resuming updates automatically.
- **Navigation:** Streamlined flow from setup to dashboard, with back button configured to **exit the app**.

Tech Stack:

Built with **Flutter (Dart)**, using provider for state, http for APIs (ThingSpeak, Gemini), and packages like connectivity_plus, qr_code_scanner, permission_handler, and flutter_launcher_icons.

VitalSync empowers users with simple, real-time health understanding directly on their mobile device.

App Wireframe :

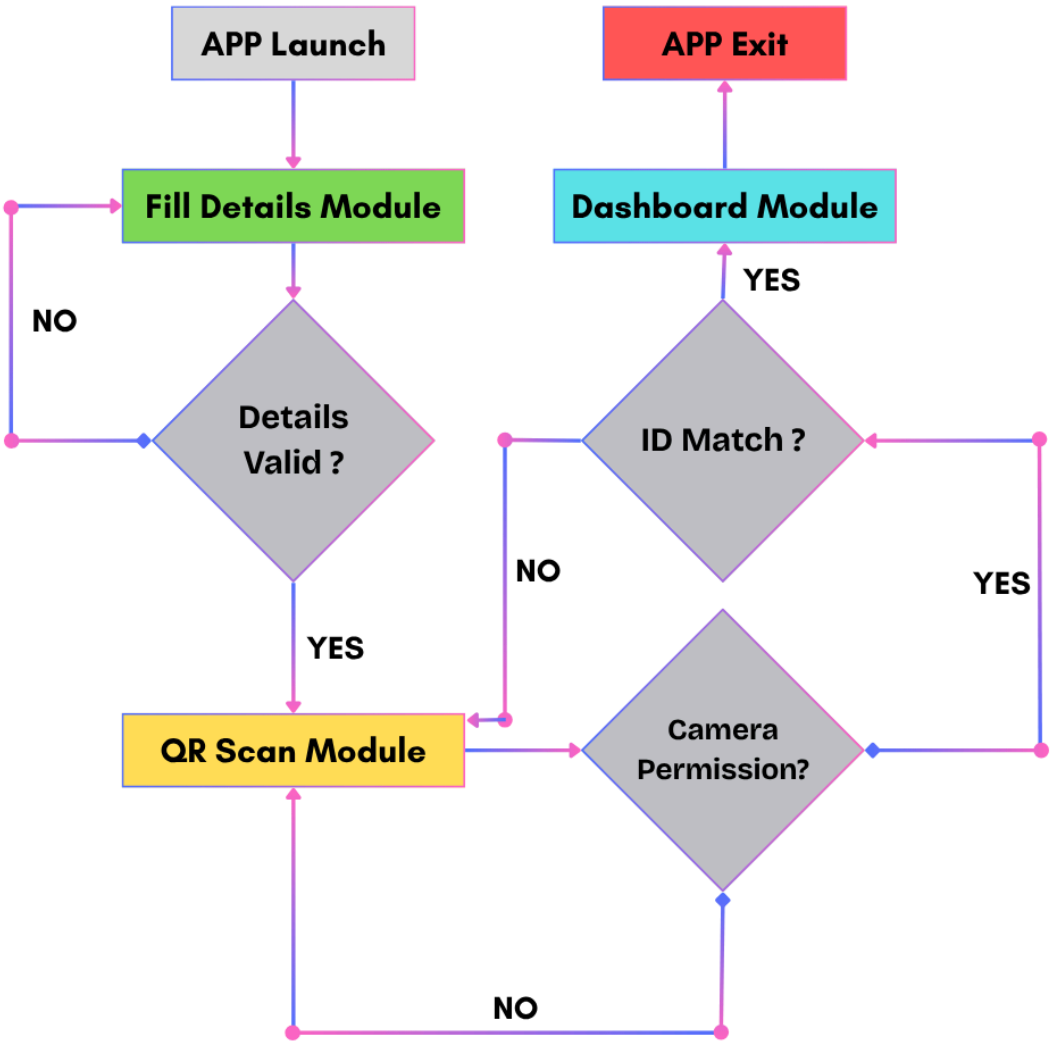


Fig 3.5

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Live Dashboard Outputs (ThingSpeak + Flutter)

The system successfully transmits real-time sensor data from the wearable device to the ThingSpeak IoT platform. Separate channels were created for heart rate, body temperature, and accelerometer data, allowing continuous monitoring.

The Flutter mobile app mirrors this data by pulling it from the cloud backend, showing:

- **Welcome Message:** Displays "Welcome user_name" (indicating the user's name).
- **Heart Rate**.
- **Temperature**
- **Weight**
- **Height**
- **BMI**
- **AI Health Insight:** Provides a personalized text insight, starting with "Hello User_name," and continuing with an analysis of health parameters, mentioning the BMI being in the normal range and suggesting maintaining a balanced diet.

Screenshots from the dashboard confirm that the system performs well .

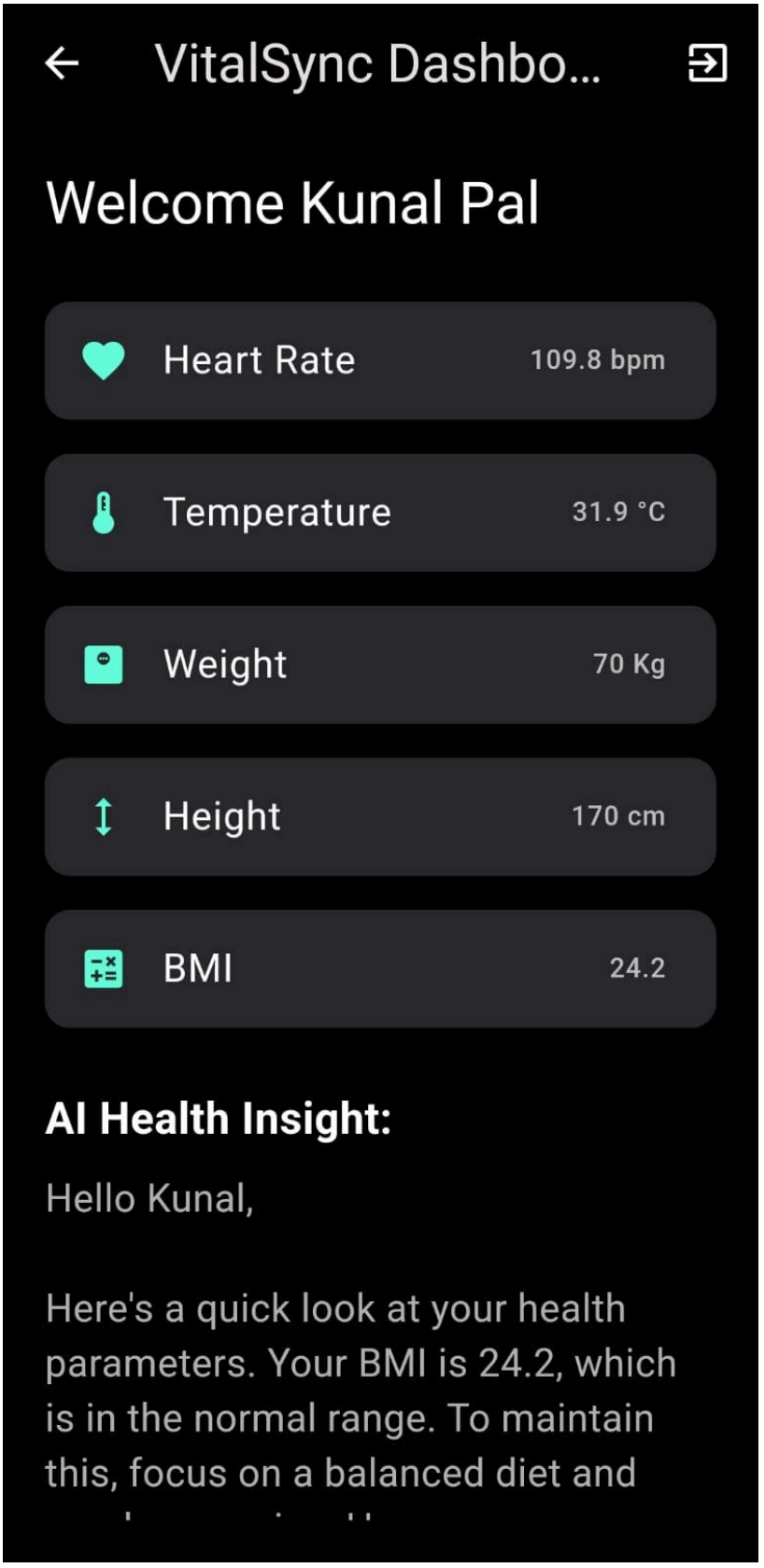

VitalSync Dashbo...
Welcome Kunal Pal
Heart Rate
109.8 bpm
Temperature
31.9 °C
Weight
70 Kg
Height
170 cm
BMI
24.2
AI Health Insight:
Hello Kunal,
Here's a quick look at your health parameters. Your BMI is 24.2, which is in the normal range. To maintain this, focus on a balanced diet and

Fig 4.1

4.2 Real-Time Fall Detection Testing

The machine learning model was rigorously tested with live and replayed sensor data. Controlled falls and normal movements (e.g., sitting, lying down, jumping) were used to assess its accuracy. Out of **50** test scenarios:

- **Fall cases detected correctly: 22**
- **False negatives (missed falls): 1**
- **False positives (non-fall misclassified): 3**

This yields a **precision of 0.88 (88%)** and an **accuracy of 92%**, confirming the model's reliability for real-world use.

4.3 Email Alert and Location Case Studies

Upon detecting a fall, the system sends an automatic alert email using SMTP with the following details:

• Message: "Fall detected for [User]"

• GPS Location: (e.g., Latitude: 22.5726, Longitude: 88.3639)

• Map Link: A clickable Google Maps link to the user's location

• Hospital List: Top 3 nearest hospitals within a 5 km radius

In real-time tests, alerts reached emergency contacts in under 10 seconds from fall detection. These alerts were received on both mobile and desktop email clients without issue.

4.4 System Performance and User Feedback

Backend & Cloud Performance

• Average response time for API prediction: < 1.2 seconds

• Email delivery time after fall detection: ~9 seconds

• Uptime on Render cloud during 72-hour stress test: 100%

User Testing and Feedback

• 10 users (age 20–60) tested the wearable for 2–3 hours each.

• Feedback summary:

- 90% rated the device as comfortable
- 80% found the app interface intuitive
- All users confirmed receiving alerts promptly
- Minor improvement suggested: battery life optimization

The results demonstrate that the system is capable of real-time, reliable health monitoring and fall detection with quick emergency response features. Both technical validation and user trials confirm that the solution meets its objectives effectively.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Summary of Achievements

The Fall Detection and Real-Time Health Monitoring System was successfully designed, implemented, and validated through both technical performance metrics and user feedback. The project achieved its core objectives, which included:

- Development of a compact wearable wristband integrating heart rate, body temperature, motion, and GPS sensors.
- Real-time data transmission to ThingSpeak and cloud backend using Wi-Fi-enabled NodeMCU (ESP8266).
- Machine Learning-based fall detection using a dataset of over 370,000 entries, yielding over 92% accuracy.
- An automated alert system capable of emailing emergency contacts with real-time location and nearby hospital information.
- Integration of GenAI for health data interpretation, translating raw sensor values into understandable health messages.
- Development of a Flutter-based mobile application providing live dashboards, health metric visualization..
- Deployment of backend services on Render cloud for scalability and high availability.

These components were integrated into a unified system that performs reliably, is user-friendly, and aligns with modern smart healthcare goals.

5.2 Limitations

Despite the project's successful outcomes, a few limitations were observed:

1. **Power Consumption:** Continuous Wi-Fi usage and sensor operation can drain the battery quickly.
2. **GPS Indoors:** GPS accuracy decreases significantly indoors or in densely built environments.
3. **Wearability:** The current prototype, while functional, can be further miniaturized for long-term comfort.
4. **Sensor Interference:** Simultaneous readings from multiple sensors can sometimes lead to noise or slight delays.

These limitations highlight areas for improvement in both hardware design and optimization of power and data flow.

5.3 Future Enhancements

The project opens multiple avenues for further development and real-world deployment. Potential future enhancements include:

• **Hospital Emergency Integration:** Direct API linkage with hospital systems for automated ambulance dispatch.

• **Telemedicine Extension:** Add a virtual consultation feature connecting users with doctors via the mobile app.

• **Miniaturization & Custom PCB:** Replace breadboard setup with a custom PCB for better form factor and durability.

• **Data Analytics & Trends:** Implement AI-based predictive analytics for early health issue detection based on trends.

These upgrades can significantly enhance the utility, scalability, and impact of the system in real-world healthcare environments.

REFERENCES

1. Stampfler, Tristan, et al. "Fall detection using accelerometer-based smartphones: Where do we go from here?." Frontiers in public health 10 (2022): 996021.
2. Amir, Nur Izdihar Muhd, et al. "Development of fall detection device using accelerometer sensor." 2021 IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA). IEEE, 2021.
3. Jiang, Zhiyuan, et al. "Fall detection systems for internet of medical things based on wearable sensors: A review." IEEE Internet of Things Journal (2024).
4. Tian, Jingxiao, Patrick Mercier, and Christopher Paolini. "Ultra low-power, wearable, accelerated shallow-learning fall detection for elderly at-risk persons." Smart Health 33 (2024): 100498.
5. González-Castro, Ana, et al. "Predicting fall risk in older adults: A machine learning comparison of accelerometric and non-accelerometric factors." Digital Health 11 (2025): 20552076251331752.
6. Bhargav, M. S. S., et al. "Efficient fall detection for elderly with integrated machine learning and sensor networks." Recent Trends in VLSI and Semiconductor Packaging (2025): 92-100.
7. Khojasteh, Samad Barri, et al. "Improving fall detection using an on-wrist wearable accelerometer." Sensors 18.5 (2018): 1350.
8. Khojasteh, Samad Barri, et al. "Improving fall detection using an on-wrist wearable accelerometer." Sensors 18.5 (2018): 1350.
9. Sucerquia, Angela, José David López, and Jesús Francisco Vargas-Bonilla. "Real-life/real-time elderly fall detection with a triaxial accelerometer." Sensors 18.4 (2018): 1101.
10. Putra, I. Putu Edy Suardiyana, et al. "An event-triggered machine learning approach for accelerometer-based fall detection." Sensors 18.1 (2017): 20.

11. Palmerini, Luca, et al. "Accelerometer-based fall detection using machine learning: Training and testing on real-world falls." Sensors 20.22 (2020): 6479.
12. Wang, Fu-Tai, et al. "Threshold-based fall detection using a hybrid of tri-axial accelerometer and gyroscope." Physiological measurement 39.10 (2018): 105002.
13. Chen, Kuang-Hsuan, Jing-Jung Yang, and Fu-Shan Jaw. "Accelerometer-based fall detection using feature extraction and support vector machine algorithms." Instrumentation Science & Technology 44.4 (2016): 333-342.
14. Kurniawan, A., A. R. Hermawan, and I. K. E. Purnama. "A wearable device for fall detection elderly people using tri dimensional accelerometer." 2016 international seminar on intelligent technology and its applications (ISITIA). IEEE, 2016.
15. Bagala, Fabio, et al. "Evaluation of accelerometer-based fall detection algorithms on real-world falls." PloS one 7.5 (2012): e37062.
16. Kwolek, Bogdan, and Michal Kepski. "Improving fall detection by the use of depth sensor and accelerometer." Neurocomputing 168 (2015): 637-645.
17. Aguiar, Bruno, et al. "Accelerometer-based fall detection for smartphones." 2014 IEEE International Symposium on Medical Measurements and Applications (MeMeA). IEEE, 2014.
18. Gjoreski, Hristijan, Mitja Lustrek, and Matjaz Gams. "Accelerometer placement for posture recognition and fall detection." 2011 Seventh International Conference on Intelligent Environments. IEEE, 2011.