Fall detection device for elderly care using IoT, Cloud and Machine Learning

Harigovind Krishnan
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23010@cb.students.amrita.edu

Karishini S
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23013@cb.students.amrita.edu

Vimal P Menon
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23050@cb.students.amrita.edu

Gowri J S
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23055@cb.students.amrita.edu

Akhil V M
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, India
cb.ai.u4aid23055@cb.students.amrita.edu

***Abstract*—Physical health of the elderly population is one of the most worrisome factors regarding their well-being as minor accidents may result in serious injuries that may lead to long lasting or serious effects on their overall health. Falls are one such accidents that have a common occurrence among the elderly. In this paper, a Smart fall detection device is proposed, implementing Internet of Things (IoT) to obtain orientation data from the MPU6050 accelerometer and gyroscope sensor connected to an ESP32 microcontroller, threshold based fall detection algorithms to detect falls and Machine learning models implementing decision tree algorithm to confirm the occurrence of falls and cloud for data transmission between the HiveMQ MQTT broker and its clients via Wi-Fi (ESP32 and ML model) and send alerts as notifications to the caregiver's phone when falls are detected by the ML model in real time via the Blynk App, using HTTP GET request.**

***Index Terms*—Smart fall detection device, IoT, ESP32, Machine learning, cloud, Blynk App**

I. Introduction

Falls while performing daily activities pose a great risk to the health of the elderly population all across the world. Annually, around a third of the elderly demographic experience falls at bathrooms and stairs which are risky areas [1]. WHO has reported that around 28% to 35% of the population nearing 65 of age and 35% to 42% of the population aged 70 years old, experience falls at least once a year, and due to the shortage of nursing homes, are forced to stay at home in about 64 countries [2].

Given the frailty of the elderly population, such falls can lead to serious injuries which may lead to permanent worsened health conditions. Another dangerous condition is when the subject remains on the ground, unable to get up after the fall [3]. This often occurs as the caregivers may be unaware of the fall and thus leads to a delay in aid to the subject. Already existing fall detection systems can be classified based on their implementation into three types, viz. Wearable system (WS), Non wearable system (NWS) and Environment sensing System (ES) [4]. NWS mainly makes use of vision sensors to monitor the subject and this has been proven to be robust and powerful but is usually not preferred as it interferes with the subject's privacy as it demands cameras to be installed in various locations of the house where the elderly subject lives and thus also proves to not be cost efficient [5][6]. The MPU6050 is a 6-axis gyroscope and accelerometer that is the most commonly used accelerometer. It is widely used in smartphones and tracking devices. By combining a 3-axis gyroscope and accelerometer on a single chip along with an onboard Digital Motion Processor (DMP), it can be used to execute advanced 9-axis motion fusion algorithms [4][7].

The goal of this work is to provide an efficient and low-cost real time fall detection device that can accurately detect if a fall has occurred and can send alerts to the caregivers in case of a fall [7]. The implementation leverages Internet of Things (IoT), using the MPU6050 sensor to obtain the subject's orientation data this data is processed by an ESP32 microcontroller on which a threshold algorithm is implemented. When the threshold is breached, indicating a possible fall, the sensor values of that second are sent to a Machine Learning (ML) model implementing decision tree algorithm via the cloud (HiveMQ). If the ML model also predicts a fall, an alert is sent to the caregiver's phone via the Blynk app.

Traditional threshold only systems are prone to trigger false positives and thus, an additional layer of confirmation using ML algorithms enhances the fall detection accuracy. Threshold based algorithms are also computationally light and fast and can run on the ESP32 locally, which eliminates the need of

Identify applicable funding agency here. If none, delete this.

sending all sensor values to the cloud, which may further cause latency. The cloud-based communication also enables scalability which can be a future enhancement to this solution. Therefore, this is a basic algorithm that is widely used but cannot be relied on to provide accurate fall state information at all times [8]. The remainder of the paper is structured as follows: Section II discusses the methodology which is followed by System overview in section III. The results are discussed in section IV and the paper is concluded in section V.

II. METHODOLOGY

This system proposes a wearable fall detection device which involves various processes working together, from data acquisition, data transmission using MQTT protocols, ML based fall prediction and alerts using the notification mechanism. The sensor readings from the MPU6050 is processed by the ESP32 and the threshold algorithm is applied on this data. In case a fall is detected, the sensor readings are published to the MQTT broker via Wi-Fi and the ML model subscribes to this data and predicts if a fall has occurred or not. If the ML model predicts a fall, a notification is sent to the caregivers phone via the Blynk App. Figure 1 illustrates the process flow from the detection of a fall from the MPU6050 sensor, the transfer of data via the ESP32 over Wi-Fi to the MQTT broker, processing in Google Colab and the sending of the alert via the Blynk app.

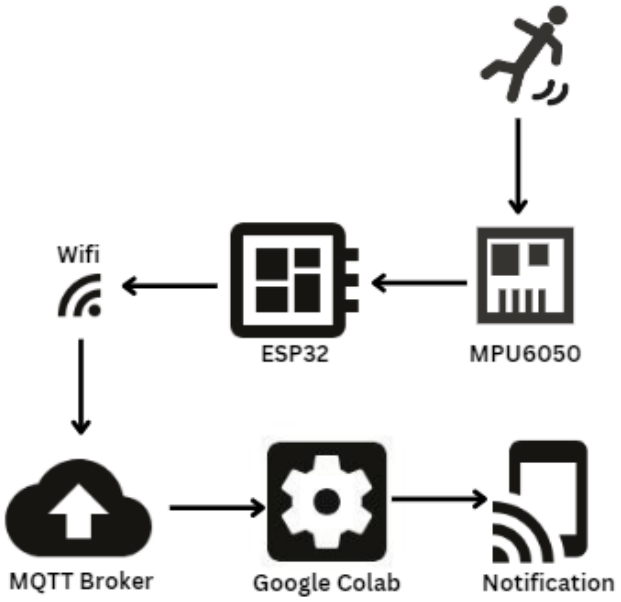

Fig. 1: Process flow of the Fall Detection System

A. Data acquisition:

The MPU6050 sensor is used to acquire orientation data of the subject i.e. acceleration and angular velocity along the X, Y and Z axes when the device is worn. The MPU6050 is a 6-axis gyroscope and accelerometer sensor. The sensor raw data is represented in the form of 16-bit signed integers[7]. The accelerometer uses microelectromechanical systems (MEMS) technology to measure the linear acceleration in the 3 directions. It measures tilt by measure the capacitive changes caused by movement of MEMS structures when external force is applied. The gyroscope also uses MEMS technology to measure angular velocity. It detects rotation be detecting the Coriolis effect on the vibrating masses inside the sensor. The gyroscope is set to its default output rate of 8 kHz. The MPU6050 in this approach has been set to ±2g and ±2000°/s.

The acceleration and gyroscope readings are obtained using the getEvent() function. This function returns the acceleration values in terms of m/s2 and the angular rotation in terms of °/s. To convert these values to raw 16-bit sensor values, the following formulas are used:

1) *To convert acceleration values to raw sensor values::*

$$Raw\ value = \frac{Acceleration\ in\ m/s^2 * 2048}{9.81} \quad (1)$$

2) *To convert angular velocity values to raw sensor values::*

$$Raw\ value = \frac{Values\ in\ degree/s}{16.8} \quad (2)$$

B. Threshold based algorithm:

The initial threshold values for the threshold based fall detection algorithm were taken as 1.95 g and 1.52 deg/s for FT1 and FT2 respectively, from an already published study [9]. However, these values did not work perfectly in our case and thus the threshold values were fine-tuned using trial and error until desired accuracy was obtained for detecting falls. Figure 2 presents a flowchart illustrating the working of the threshold based algorithm.

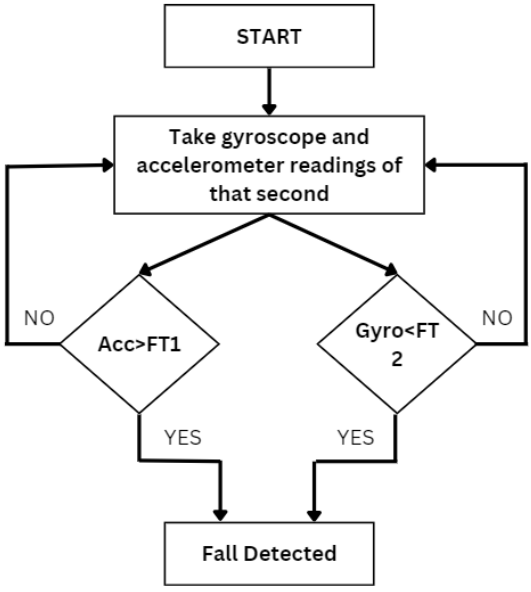

Fig. 2: Flowchart describing threshold algorithm

FT1 is the lower acceleration threshold and FT2 is the upper angular velocity threshold.

C. Data transmission using MQTT protocol:

The free public HiveMQ MQTT broker is used in this work. The ESP32 and the ML model in Google Colab are the clients that send and receive data, thereby sharing a server-client relationship with the HiveMQ broker.

The ESP32 publishes sensor readings that breach the threshold to the specified topic of the broker via Wi-Fi and the ML model subscribes to this topic to receive the data being sent. The ML model then predicts whether a fall has occurred of this given orientation data. In case of a threshold breach, the data from the sensors are first converted to raw sensor values using the formulas mentioned previously and then published to the specified topic.

D. Dataset:

The Machine Learning model has been trained on the publicly available Sisfall dataset [11]. The dataset contains orientation data collected from 38 individuals across various age brackets. It consists of data collected from 23 young adults performing 19 activities of daily living (ADL) and 15 fall activities, 15 ADL types performed by 14 elderly subjects aged over 62 and data from one elderly subject aged 60 who performed ADLs as well as falls [10]. It consists a total of 2706 ADL and 1798 falls, constituting a total of 1,58,58,929 data instances.

The values in the dataset were collected using two accelerometers, the ADXL345 and MMA8451Q, and one gyroscope, the ITG3200 sensor [10]. Each data instance in the dataset is represented as a set of 9 values. The first 3 values are the raw accelerometer data from the ADXL345 sensor with a full-scale range of ±16g and a resolution of 13 bits. The next 3 values are the angular velocity values from the ITG3200 sensor with a full-scale range of ±2000°/s and resolution of 16 bits. The last 3 values are the raw accelerometer data from the MMA8451Q sensor with a full-scale range of ±8g and resolution of 14 bits.

E. Data acquisition:

The MPU6050 has a resolution of 16 bits and the full-scale range has been set to ±16g. Therefore, there arises the need for a conversion of the MPU6050 sensor data in order for being used to predict a fall by the ML model which has been trained on the Sis fall dataset. The accelerometer data from the MPU6050 sensor must be converted to a resolution of 14-bit and 13-bit with full-scale ranges of ±8g and ±16g respectively [11].

Conversion to 13-bit with full-scale range of ±16g: This requires conversion from a resolution of 16 bits to 13 bits. This is done by discarding the 3 Least significant bits (LSB) of the former. Thus, the sensor data is divided by 23. There is no conversion needed in the full-scale range from ±16g to ±16g. The conversion formula is as follows:

$$13\ bit\ \pm 16g\ raw\ value = \frac{Raw\ sensor\ value}{8} \qquad (3)$$

Conversion to 13-bit with full-scale range of ±16g: This requires conversion from a resolution of 16 bits to 14 bits. This is done by discarding the 2 Least significant bits (LSB) of the former. Thus, the 16-bit data is divided by 22. The conversion of full-scale range from ±16g to ±8g requires division of the sensor value by 2.

$$14\ bit\ \pm 8g\ raw\ value = \frac{Raw\ sensor\ value}{8} \qquad (3)$$

F. Machine Learning Model:

The Decision tree algorithm is employed to classify the sensor data from MPU6050 sensor. The raw sensor readings are used as features for training the model. The tree is trained to classify the given sensor data as a fall or non-fall event based on these features. The Logistic regression algorithm was also applied to the Sisfall dataset, which resulted in an accuracy of 76.94%.

G. Notification and alert system:

When the Machine Learning model predicts a fall, it triggers an HTTP GET request. This updates the value of a switch created on the Blynk web dashboard. The switch is connected to a virtual pin with possible values 1 and 0 representing ON and OFF state respectively. An automation is created to send an in-app notification when the pin is ON. Thus, each time the HTTP GET request updates the pin value to 1, an in-app notification is sent to the caregiver's phone. Figure 3 shows a screenshot of the notification received on the user's phone via the Blynk app when a fall has occurred.

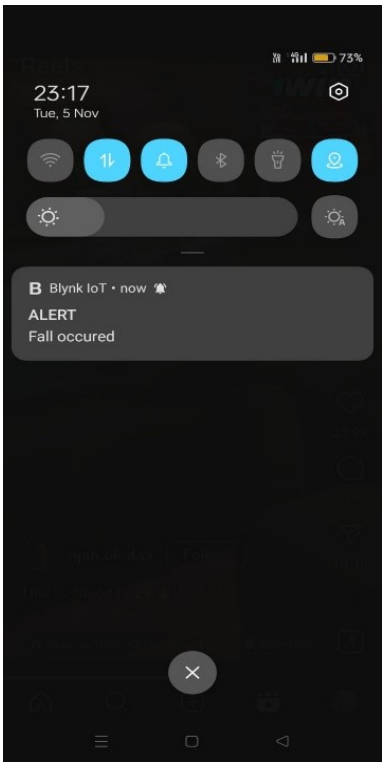

Fig. 3: Test notification sent via Blynk IoT app

III. SYSTEM OVERVIEW

The system consists of various components such as hardware for data collection and processing, communication protocols for data transmission, cloud based ML model for confirming fall occurrence and notification mechanism to alert caregivers in case of a fall.

A. Hardware Components:

The hardware components used in this device are the ESP32 microcontroller, MPU6050 sensor and a power source to power the ESP32. The role of each component is as follows:

- ESP32 : It is the main processing unit of the device. It is a low cost and power efficient microcontroller that has an integrated Wi-Fi and Bluetooth module making it the preferred choice in IoT applications. In this approach, the ESP32 is connected to the MPU6050 sensor to obtain the gyroscope and accelerometer readings. A threshold based fall detection algorithm is implemented using these sensor readings, that is run locally on the ESP32. The ESP32 is also connected to the cloud via Wi-Fi and the sensor readings are sent via this Wi-Fi connection in case the threshold is breached and a fall is detected.
- MPU6050 : The MPU 6050 is a 6-axis gyroscope and accelerometer sensor, measuring the angular velocity and acceleration along the 3 axes respectively. The sensor provides raw data as 16 bit signed integers.

Fig. 4: Compact and wearable device as proposed

Figure 4 shows the wearable hardware of the fall detection device, which is integrated to a belt worn around the user's waist.

The range of the accelerometer and gyroscope of the MPU6050 sensor can be set to fit different applications. These settings determine the maximum range of data the sensor can measure. The accelerometer can be configured from ±2g (most sensitive) to ±16g (least sensitive). Table I shows the ranges and sensitivity configurations of the accelerometer:

Table I: Range and corresponding sensitivity values for accelerometer

Range	Sensitivity
±2g	16384 LSB/g
±4g	8192 LSB/g
±8g	4096 LSB/g
±16g	2048 LSB/g

The gyroscope can be configured from ±250 degrees per second to ±2000 degrees per second. Table II shows the range configurations along with their sensitivities for the gyroscope:

Table II: Range and corresponding sensitivity values for gyroscope

Range	Sensitivity
±250°/s	131 LSB/°/s
±500°/s	65.5 LSB/°/s
±1000°/s	32.6 LSB/°/s
±2000°/s	16.4 LSB/°/s

- Power management: The system is powered by a portable power bank that connects to the ESP32 via USB cable. The power bank's large capacity ensures long operational hours and supports device portability. Figure 5 illustrates the circuit diagram of the fall detection device.

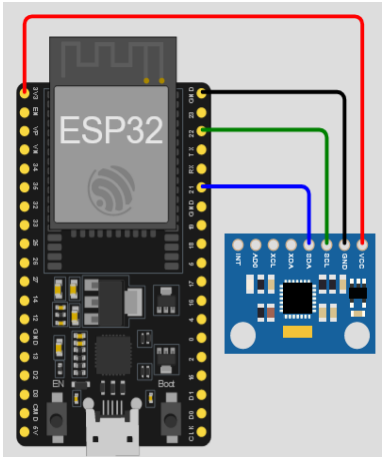

Fig. 5: ESP3 and MPU6050 Circuit connection

B. Data transmission:

Data transmission is carried out between the HiveMQ broker and its clients via MQTT protocol. HiveMQ broker used in this work is a public broker with port 1883. The 2 clients in this system are the ESP32 and the ML model in Google Colab. The ESP32 publishes sensor readings to a particular topic to the broker in case the threshold algorithm detects a fall. The ML model in Google Colab subscribes to this topic and predicts if a fall has occurred using this data.

C. Machine Learning model:

The Machine Learning model in Google Colab, subscribes to the topic and the sensor values published by the ESP32 is given to the model that implements the Decision Tree algorithm to predict if a fall has occurred or not.

D. Notification System:

The notification mechanism uses Blynk cloud services to send in app notification in the Blynk mobile app of the caregiver. The notification is triggered by a virtual pin update via API call done by executing an HTTP request to Blynk cloud API.

IV. RESULTS & DISCUSSION:

Threshold based algorithm had been successfully implemented on ESP32 in our work with threshold values determined by trial and error. In our framework, both Decision Tree and Logistic Regression algorithms were applied to the Sisfall dataset. The decision tree algorithm achieved an accuracy of 87% while logistic regression resulted in an accuracy of 76.94%. Decision tree is one of the most popular and accurate algorithms used in fall detection systems, resulting in accuracies up to 96% [12]. Table III draws comparison between the results of different Machine Learning algorithms implemented on the Sisfall dataset for fall detection.

Table III: Performance comparison of Machine Learning algorithms

Algorithm	Accuracy	F1 Score
Decision Tree	87.91%	0.87
Logistic Regression	76.94%	0.76
SVM [12]	84.17%	0.84

The confusion matrix of the machine learning model implementing Decision tree algorithm is given in figure 6.

The confusion matrix illustrates the model's performance in classifying between fall and non-fall events.

- Actual non-fall vs predicted non-fall: Top left box shows 1,962,701 non-fall events were correctly classified, indicating a high number of true negatives.
- Actual non-fall vs predicted fall: The top right box shows 267,319 non-fall events that were classified as falls, representing false positives.
- Actual fall vs predicted non-fall: The bottom left box shows 260,679 fall events that were classified as non-fall, representing false negatives.
- Actual fall vs predicted fall: The bottom right box shows 1,357,435 fall events that were correctly classified as fall, indicating a high number of true positives.

Figure 7 and Figure 8 illustrate the variation of accelerometer and gyroscope values along the X, Y and Z axes across a time span of 15 seconds with the fall occurring at the 13th second.

Figure 7 depicts the acceleration (in m/s^2) vs time (in seconds) graph, measured by the accelerometer of the MPU6050.

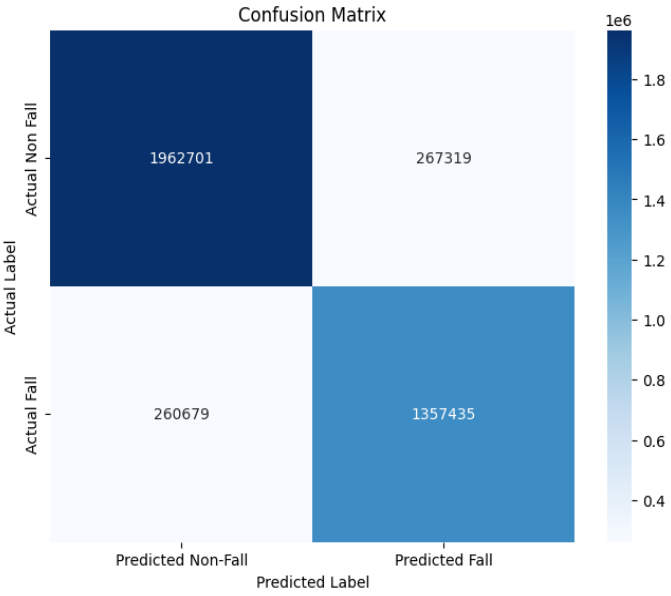

Fig. 6: Confusion Matrix of Decision tree algorithm

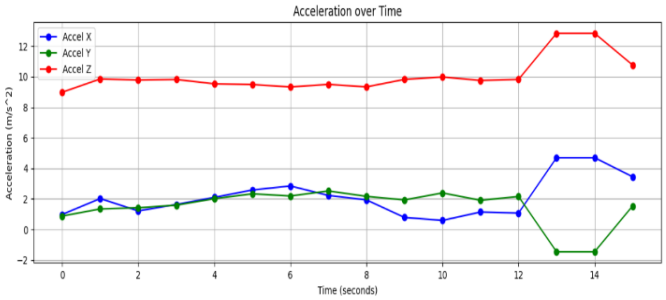

Fig. 7: Acceleration values along X, Y and Z axes

The red line corresponding to the acceleration along the Z-axis shows a noticeable change after the 12^{th} second, increasing from approximately 10 m/s^2 (acceleration due to gravity) to over $12m/s^2$, indicating significant movement. Similarly,the accelerations along the X and Y axes, represented by the blue and green lines, start from near zero values and show variations at the 12^{th} second, further suggesting the occurrence of a fall.

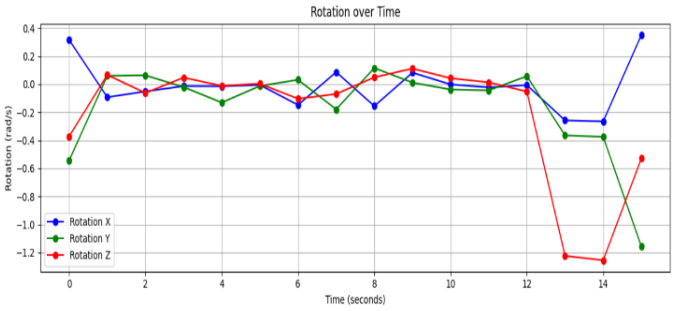

Fig. 8: Rotation values along X, Y and Z axes

Figure 8 illustrates a graphs depicting the rotation rates (in

°/s) versus time (in sec), measured using the gyroscope of the MPU6050. Here, the blue, green and red lines, representing the rotations along the X, Y and Z-axis, respectively, start from near zero values and remain steady. However, significant variations are observed after the 12th second, particularly in the Z-axis (along the direction of acceleration due to gravity), thereby indicating the occurrence of a fall.

V. CONCLUSION:

This work proposes a fall detection device for the elderly leveraging IoT, ML and cloud integrating the MPU6050 sensor, ESP32 microcontroller and a hybrid approach of threshold and Machine learning algorithms thereby providing a cost effective and efficient solution to detect falls and alert the caregivers in real time via notifications in the Blynk app. The solution overcomes the limitations of traditional, non-wearable devices, proving to be a viable and efficient solution that enhances scalability. The hybrid approach, combining the implementation of a threshold algorithm along with an ML algorithm reduces data transmission and processing latency, ensuring faster results, enabling the caretakers to respond quickly to emergencies.

Future enhancements can include fall prediction with the incorporation of an ECG sensor for heart rate monitoring. This can help predict the physiological condition of the subject which may indicate an impending fall as well as the use of multiple sensors can improve fall detection accuracy.

REFERENCES

[1] Sudthongkhong, Chudanat, Siwat Suksri, Chanate Ratanaubol, Sookyuen Tepthong, Jira Jitsupa, and Putawan Suksai. "NVIDIA Jetson Nano and Python-based Economical Human Fall Detection and Analysis System." In 2023 17th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 463-467. IEEE, 2023.

[2] WHO. WHO Global Report on Falls: Prevention in Older Age. World Health Organization, 2007.

[3] Vallabh, Pranesh, and Reza Malekian. "Fall detection monitoring systems: a comprehensive review." Journal of Ambient Intelligence and Humanized Computing 9, no. 6 (2018): 1809-1833.

[4] Fernández-Canosa, Sara, Verónica López González, Antonio Consoli, and Vanesa Soto-León. "Healthcare Based on IoT Using Arduino, MPU6050 Accelerometer and Gyroscope Sensor and FSR-174 Strain Gauge for Fatigue." In International Conference on Pervasive Computing Technologies for Healthcare, pp. 138-148. Cham: Springer Nature Switzerland, 2022.

[5] Yacchirema, Diana, Jara Suárez De Puga, Carlos Palau, and Manuel Esteve. "Fall detection system for elderly people using IoT and big data." Procedia computer science 130 (2018): 603-610.

[6] Mubashir, Muhammad, Ling Shao, and Luke Seed. "A survey on fall detection: Principles and approaches." Neurocomputing 100 (2013): 144-152.

[7] Gupta, Akash, Rohini Srivastava, Himanshu Gupta, and Basant Kumar. "IoT based fall detection monitoring and alarm system for elderly." In 2020 IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), pp. 1-5. IEEE, 2020.

[8] Dias, Paulo Victor GF, Edgar Douglas M. Costa, Michel Pompeu Tcheou, and Lisandro Lovisolo. "Fall detection monitoring system with position detection for elderly at indoor environments under supervision." In 2016 8th IEEE Latin-American Conference on Communications (LATINCOM), pp. 1-6. IEEE, 2016.

[9] Guo, Han Wen, Yi Ta Hsieh, Yu Shun Huang, Jen Chien Chien, Koichi Haraikawa, and Jiann Shing Shieh. "A threshold-based algorithm of fall detection using a wearable device with tri-axial accelerometer and gyroscope." In 2015 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), pp. 54-57. IEEE, 2015.

[10] Sucerquia, Angela, José David López, and Jesús Francisco Vargas-Bonilla. "SisFall: A fall and movement dataset." Sensors 17, no. 1 (2017): 198.

[11] N. V. Nikhil, "SisFall original dataset," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/nvnikhil0001/sis-fall-original-dataset. [Accessed: Nov. 2024].

[12] Badgujar, Sejal, and Anju S. Pillai. "Fall detection for elderly people using machine learning." In 2020 11th international conference on computing, communication and networking technologies (ICCCNT), pp. 1-4. IEEE, 2020.

[13] Sowmya, Aki, and Anju S. Pillai. "Human fall detection with wearable sensors using ml algorithms." In 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), pp. 1092-1095. IEEE, 2021.

[14] Hong, Wong Chiew, Yee Siong Seng, Wan Nur Arifah Mior Idris, Tang Chee Yee, Teoh Hui Mun, and Zarina Tukiran. "Development of IoT-based Health Monitoring System using Blynk." Evolution of Information, Communication and Computing System 2, no. 1 (2020): 76-84.

[15] Purushothaman, Anagha, K. V. Vineetha, and Dhanesh G. Kurup. "Fall detection system using artificial neural network." In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), pp. 1146-1149. IEEE, 2018