VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

IOT PROJECT FINAL REPORT

ELDERLY FALL DETECTION USING ESP32 AND MPU9250

COURSE CODE: BECE351E

SLOT: L3+L4

SUBMITTED BY:

ALEN PAUL ALAPATT (21BPS1155)

DONEL CHACKO BABY (21BRS1382)

HARSHA R (21BPS1248)

VEMULA SAI AKHIL (21BAI1864)

INDEX

ABSTRACT

This provides an overview of the latest trends and challenges in fall detection systems for elderly people and discusses the different types of fall detection systems, as well as the advantages and disadvantages of each type. It also discusses the challenges that need to be addressed in order to improve the performance of fall detection systems.

We also discuss the future of fall detection systems for elderly people. In this project we will also see that the multiple sensors used in fall detection systems and also the machine learning algorithms used in them. The aim of this project is to build a fall detection system for the elderly using an MPU9250 sensor, an ESP32 microcontroller, and the Blynk app.

The system detects falls by measuring the acceleration of the body and comparing it to a threshold value. When the threshold value is exceeded, a notification is sent to the Blynk app, which can be programmed to alert caregivers or emergency services. The system is designed to be non-invasive and easy to use, making it ideal for elderly people who are at risk of falling. The project can be further developed by incorporating additional sensors or machine learning algorithms to improve the accuracy and reliability of the fall detection system.

INTRODUCTION

Falls are a major cause of injury and death among elderly people. Falls are also the leading cause of injury-related death among older adults. In this project we are going to see how Fall detection systems can help to reduce the risk of serious injury or death from falls. which can detect falls and send an alert to the person so that there can be immediate help that can be provided to reduce them

Wearable devices worn by an elderly person typically include accelerometers, gyroscopes, and other sensors that can detect changes in the user's movement (mpu9250). Fall detection systems have the potential to improve the safety of elderly people.

Fall detection systems can sometimes generate false alarms. There will be challenges like any other device but will be more helpful in that case so we prefer using them more to help the elderly people by sending a immediate alarm.

Falls are a major concern for elderly people, and can result in serious injuries or even death. Detecting falls quickly is critical for providing timely medical assistance. In this project, we aim to build a fall detection system using an MPU9250 sensor, an ESP32 microcontroller, and the Blynk app.

The system is designed to be non-invasive and easy to use, making it ideal for elderly people who are at risk of falling. The MPU9250 sensor is used to measure the acceleration of the body and the ESP32 microcontroller is used to process the data and

compare it to a threshold value. When a fall is detected, a notification is sent to the Blynk app, which can be programmed to alert caregivers or emergency services.

This project has the potential to improve the safety and well-being of elderly people, and can be further developed by incorporating additional sensors or machine learning algorithms to enhance the accuracy and reliability of the fall detection system.

COMPONENTS USED

The components used in this project are:

MPU9250 Sensor: The MPU9250 is a 9-axis Motion-Tracking sensor that combines a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer in a single package. It is used to measure the acceleration of the body, which is used to detect a fall.

ESP32 Microcontroller: The ESP32 is a low-cost, low-power microcontroller with built-in Wi-Fi and Bluetooth capabilities. It is used to process the data from the MPU9250 sensor and send notifications to the Blynk app when a fall is detected.

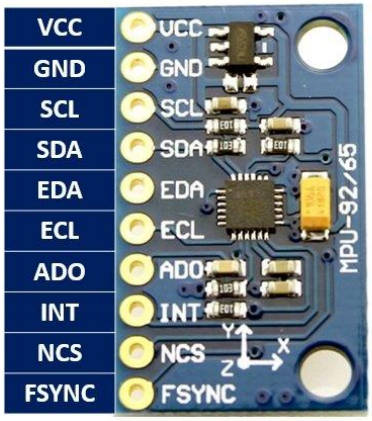
Blynk App: Blynk is a mobile app that allows users to control and monitor their connected devices. It is used in this project to receive notifications when a fall is detected.

Wires: Wires are used to connect the MPU9250 sensor to the ESP32 microcontroller.

Breadboard: A breadboard is used to connect the wires to the MPU9250 sensor and the ESP32 microcontroller.

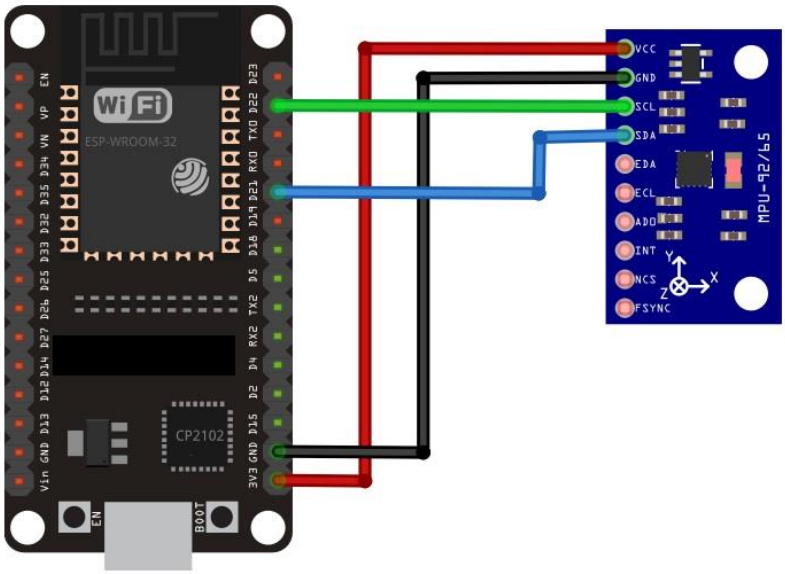
Power Supply: A power supply, such as a USB cable or battery, is used to power the ESP32 microcontroller and the MPU9250 sensor.

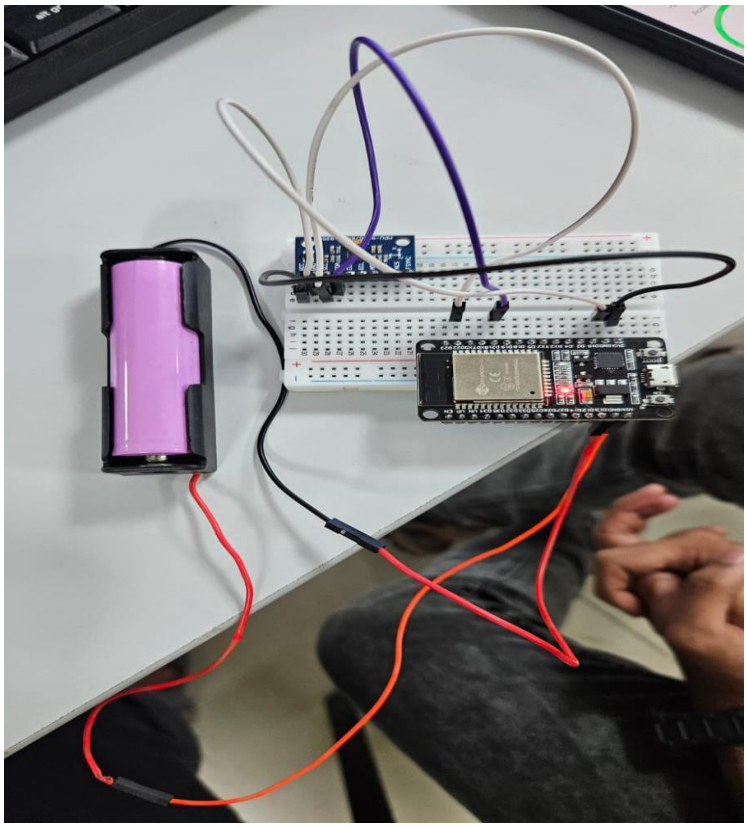
Overall, these components are used to create a simple and effective fall detection system that can be easily deployed in homes or care facilities to improve the safety and well-being of elderly people.



ARCHITECTURE

- Connect the power supply pins of the module to the ground and the 3.3 volts pin of the ESP32.
- Connect the SDA and SCL pins of the Sensor module to default I2C communication pins of ESP32
- Make sure to use long jumpers to avoid disturbance because we would rotate the module three-dimensionally to observe the result.





HOW THEY WORK

Sensors: The sensors used in fall detection algorithms typically measure acceleration, angular velocity, and sometimes pressure. These sensors are often embedded in wearable devices that are worn on the body, such as pendants, bracelets, or smartwatches.

Data collection: The sensors continuously monitor the user's movements and collect data about their posture, orientation, and acceleration patterns. This data is then processed in real-time by the fall detection algorithm.

Fall detection algorithm: The fall detection algorithm analyses the sensor data to identify patterns associated with falls. It looks for sudden changes in acceleration and orientation that indicate a fall has occurred. The algorithm is designed to distinguish between normal activities like sitting down and an actual fall event.

Threshold determination: The fall detection algorithm often includes predefined thresholds or rules to determine whether a detected event qualifies as a fall. These thresholds are set based on extensive testing and analysis of fall patterns.

Alarm generation: If the algorithm determines that a fall has likely occurred, it triggers an alarm or notification. The alarm can take different forms, depending on the system. It may involve sounding an audible alarm within the wearable device, sending an alert to a caregiver's smartphone, or automatically contacting emergency services.

User response: The user receives a notification or mail regarding the fall event. Upon receiving the fall alert, caregivers or emergency responders can take appropriate action. They can check on the individual or send immediate medical assistance.

SENSORS

MPU9250

The MPU-9250 is a 9-axis motion tracking device that integrates an accelerometer, gyroscope, and magnetometer into a single small package. It is a popular sensor for use in fall detection systems, as it can be used to detect changes in the user's acceleration, rotation, and magnetic field.

versatile and reliable sensor that can be used for fall detection. The sensor is accurate, low-power, and small, making it suitable for use in wearable devices.

And always while using any sensor we should be available of the drawbacks as well or we might lead to some problems.

ESP32

ESP32 is a series of low-cost, low-power systems on a chip (SoC) microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. It is also open source small in size with therefore they can be used easily (handy).

That is why we are using them here.

It is used in

- IOT DEVICES
- WEARABLE DEVICES
- HOME AUTOMATION
- INDUSTRIAL APPLICATIONS

In this project, we use the MPU9250 sensor to detect falls. The MPU9250 is a 9-axis Motion-Tracking sensor that combines a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer in a single package.

The accelerometer is used to measure the acceleration of the body and detect sudden changes in acceleration that may indicate a fall. When a fall occurs, the acceleration of the body changes rapidly, and the MPU9250 sensor detects this change.

The gyroscope is used to measure the angular velocity of the body and can be used to detect the orientation and movement of the body. However, in this project, we only use the accelerometer to detect falls.

The magnetometer is used to measure the magnetic field of the earth and can be used to determine the orientation of the sensor with respect to the earth's magnetic field. However, it is not used in this project.

Overall, the MPU9250 sensor is a versatile and powerful sensor that can be used for a wide range of applications, including motion tracking, orientation sensing, and fall detection.

INTERFACES

The system typically consists of a wearable device that collects data from the user`s movements, and a cloud-based server that analyses the data and detects falls. The data from the sensors is sent to the cloud-based server, where it is analysed by a machine learning algorithm. Cloud-based fall detection systems offer a number of advantages over traditional fall detection systems. These advantages include:

Scalability: Cloud-based systems are scalable, meaning that they can be easily scaled up or down to meet the needs of the application.

Security: Cloud-based systems are secure, meaning that the data is protected from unauthorized access.

Cost-effectiveness: Cloud-based systems are cost-effective, meaning that they can save money on services such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) these are the types of clouds that can be used here or we can make or cloud using Django

Or one of the other techniques are we are using blynk app which is also indeed connected to the cloud and we get the results in them.

In this project, we use two interfaces:

I2C Interface: The MPU9250 sensor communicates with the ESP32 microcontroller using the I2C (Inter-Integrated Circuit) interface. The I2C interface is a two-wire serial communication protocol that enables communication between multiple devices over short distances.

Wi-Fi Interface: The ESP32 microcontroller communicates with the Blynk app using the Wi-Fi interface. The Wi-Fi interface is a wireless communication protocol that allows devices to connect to the internet and exchange data with other devices.

The I2C interface is used to transmit data from the MPU9250 sensor to the ESP32 microcontroller, while the Wi-Fi interface is used to transmit data from the ESP32 microcontroller to the Blynk app. Together, these interfaces enable the fall detection system to detect falls and send notifications to the Blynk app.

DJANGO FRAMEWORK

Django is a Python-based high-level web framework that follows the Model-View-Controller (MVC) architectural pattern. It aims to simplify and accelerate web development by providing a comprehensive set of tools and features. One of its core strengths is the Object-Relational Mapping (ORM) system, which allows developers to interact with databases using Python code, abstracting away the complexities of SQL queries. Django also offers a flexible URL routing system, enabling the mapping of URLs to views, making it easy to design user-friendly URLs. The templating engine separates presentation logic from business logic, enhancing code maintainability. With the built-in form handling system, developers can easily manage user input and data validation.

Django includes a robust authentication system with features like user registration, login, and permissions management. The admin interface automates the creation of an administration site for managing content, saving time and effort. Security is a top priority in Django, with features to mitigate common vulnerabilities. The framework supports scalability through caching, database connection pooling, and multi-server deployment. It provides internationalization and localization support, allowing developers to create applications that cater to different languages and cultures. Django also offers a testing framework to facilitate test-driven development and ensure

application reliability. With its active community and vast ecosystem, Django empowers developers to build robust and scalable web applications with clean and maintainable code.

COMMUNICATION PROTOCOLS

I2C:

The I2C protocol is a synchronous serial communication protocol used for short-distance communication between microcontrollers, sensors, and other devices.

In this implementation, the I2C protocol is used to transfer data between the ESP32 board and the MPU9250 sensor.

It is a widely used communication protocol for connecting integrated circuits in a system. While it is not specifically related to fall detection, it can be utilized in fall detection systems for certain purposes.

Fall detection systems typically consist of various components, including sensors, microcontrollers, and communication modules. The I2C protocol can be employed to facilitate communication between these components and enable data exchange.

In a fall detection system, accelerometer sensors are commonly utilized to measure the acceleration and orientation of a person. These sensors can detect sudden changes in acceleration that indicate a fall. The accelerometer sensors can be connected to a microcontroller or a system-on-chip (SoC) that processes the sensor data.

MQTT:

MQTT is a lightweight messaging protocol that is designed for machine-to-machine communication. MQTT is typically used for streaming data, such as sensor data. MQTT is a popular choice for elderly fall detection because it is easy to implement and it is very efficient in terms of bandwidth

MQTT follows a publish-subscribe messaging pattern. Devices, known as publishers, send messages to specific topics. Other devices, known as subscribers, can subscribe to these topics to receive the messages. This decoupled model allows for flexible and scalable communication between devices. While MQTT primarily follows a publish-subscribe model, it also supports request-response communication through the use of request topics and response topics.

It is a widely adopted protocol due to its lightweight nature, efficient message delivery, and suitability for IoT deployments. It provides a scalable and reliable means of communication, making it well-suited for applications that require real-time data exchange and event-driven architectures.

CODES AND ALGORITHMS:

```
#define BLYNK_PRINT Serial
#define BLYNK_TEMPLATE_ID "TMPL3ejQYp4jL"
#define BLYNK_TEMPLATE_NAME "Elderly fall detection"
#define BLYNK_AUTH_TOKEN "QMYdQp9gxln_Pm7u3s-_1bf4JslmRc0-"
#include <MPU9250_asukiaaa.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Enter ID";
char pass[] = "Enter Pwd";
#ifdef ESP32_HAL_I2C_H
#define SDA_PIN 21
#define SCL_PIN 22
#endif
MPU9250_asukiaaa mySensor;
float aX, aY, aZ, aSqrt, gX, gY, gZ, mDirection, mX, mY, mZ;
void setup() {
  Serial.begin(115200);
  while(!Serial);
  Serial.println("started");
#ifdef ESP32_HAL_I2C_H // For ESP32
  Wire.begin(SDA_PIN, SCL_PIN);
  mySensor.setWire(&Wire);
#endif
mySensor.beginAccel();
  mySensor.beginGyro();
```

```
 Blynk.begin(auth, ssid, pass);
 }
void loop() {
 uint8_t sensorId;
 int result;
result = mySensor.readId(&sensorId);
 if (result == 0) {
  Serial.println("sensorId: " + String(sensorId));
 } else {
  Serial.println("Cannot read sensorId " + String(result));
 }
 result = mySensor.accelUpdate();
 if (result == 0) {
  aX = mySensor.accelX();
  aY = mySensor.accelY();
  aZ = mySensor.accelZ();
  aSqrt = mySensor.accelSqrt();
  Serial.println("accelX: " + String(aX));
  Serial.println("accelY: " + String(aY));
  Serial.println("accelZ: " + String(aZ));
  Serial.println("accelSqrt: " + String(aSqrt));
 } else {
  Serial.println("Cannod read accel values " + String(result));
 }
 result = mySensor.gyroUpdate();
 if (result == 0) {
  gX = mySensor.gyroX();
  gY = mySensor.gyroY();
```

```
    gZ = mySensor.gyroZ();
    Serial.println("gyroX: " + String(gX));
    Serial.println("gyroY: " + String(gY));
    Serial.println("gyroZ: " + String(gZ));
  } else {
    Serial.println("Cannot read gyro values " + String(result));
  }
Blynk.virtualWrite(V0, mySensor.accelX()*10);
  Blynk.virtualWrite(V1, mySensor.accelY()*10);
  Blynk.virtualWrite(V2, mySensor.accelZ()*10);
  Blynk.virtualWrite(V3,mySensor.gyroX());
  Blynk.virtualWrite(V4,mySensor.gyroY());
  Blynk.virtualWrite(V5,mySensor.gyroY());
  delay(50);
 Serial.println("at " + String(millis()) + "ms");
  Serial.println(""); // Add an empty line
  delay(500);
}
```

ALGORITHM:

Include required libraries – “MPU9250_asukiaaa”, “WiFi”, “WiFiClient” and “BlynkSimpleEsp32.”

Define Blynk template ID, template name, and authentication token.

Define Wi-Fi credentials - SSID and password.

Initialize the MPU9250 sensor object.

Begin serial communication with a baud rate of 115200.

Set up the I2C pins for ESP32, if using ESP32.

Begin the sensor's accelerometer and gyro.

Begin Blynk connection using the authentication token and Wi-Fi credentials.

In the loop function:

a. Read the sensor ID and print it to the Serial Monitor.

b. Read accelerometer values - x, y, z, and the square root of the sum of the squares of x, y, and z. Print these values to the Serial Monitor.

c. Read gyro values - x, y, and z. Print these values to the Serial Monitor.

d. Send accelerometer and gyro values to Blynk virtual pins V0 through V5.

e. Wait for 50 milliseconds.

f. Print the current time in milliseconds to the Serial Monitor.

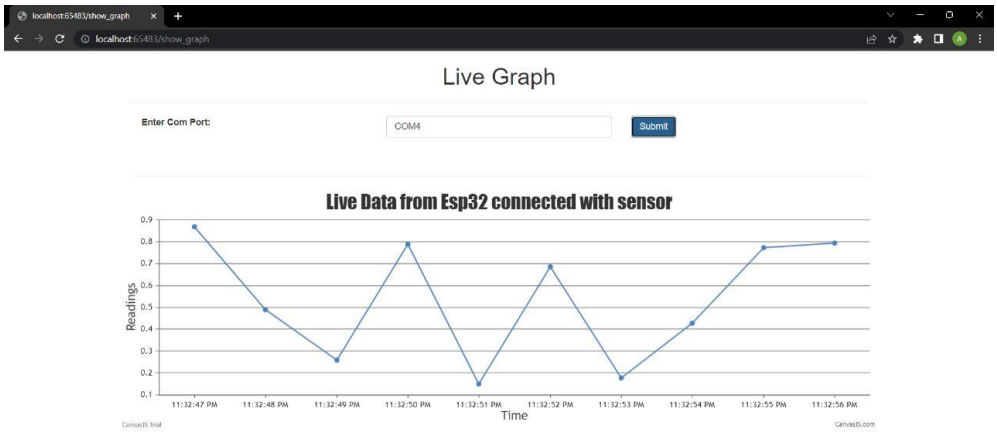
g. Wait for 500 milliseconds before starting the next iteration of the loop.

DJANGO views.py CODE:

```
from datetime import datetime
import random
from django.http import HttpResponse, JsonResponse
from django.shortcuts import render, redirect, get_object_or_404
from .forms import StudentForm
from .models import StudentModel, readings
from serial import Serial
import struct
def fetch_sensor_values_ajax(request):
    data={}
    float_val=0.0
    if request.is_ajax():
            com_port = request.GET.get('id', None)
            sensor_val=random.random()
            sensor_data=[]
            now=datetime.now()
            ok_date=str(now.strftime('%Y-%m-%d %H:%M:%S'))
            try:
```

```
            sr=Serial(com_port,115200)
            sr.setDTR(False)
            sr.setRTS(False)
            sr.open()
               st=str(sr.readline().decode())
                     sensor_val=st
            if(sensor_val):
               print(sensor_val)
               sensor_data.append(str(sensor_val)+','+ok_date)
               sensor_val2 = int(float_val)
               savy = readings(sensor_data = sensor_val2,time = ok_date)
            else:
               sensor_data.append(str(sensor_val)+','+ok_date)
               sensor_val2 = int(float_val)
               savy = readings(sensor_data = sensor_val2,time = ok_date)
         except Exception as e:
               sensor_data.append(str(sensor_val)+','+ok_date)
               sensor_val2 = int(float_val)
               savy = readings(sensor_data = sensor_val2,time = ok_date)
         data['result']=sensor_data
else:
   data['result']='Not Ajax'
return JsonResponse(data)
```

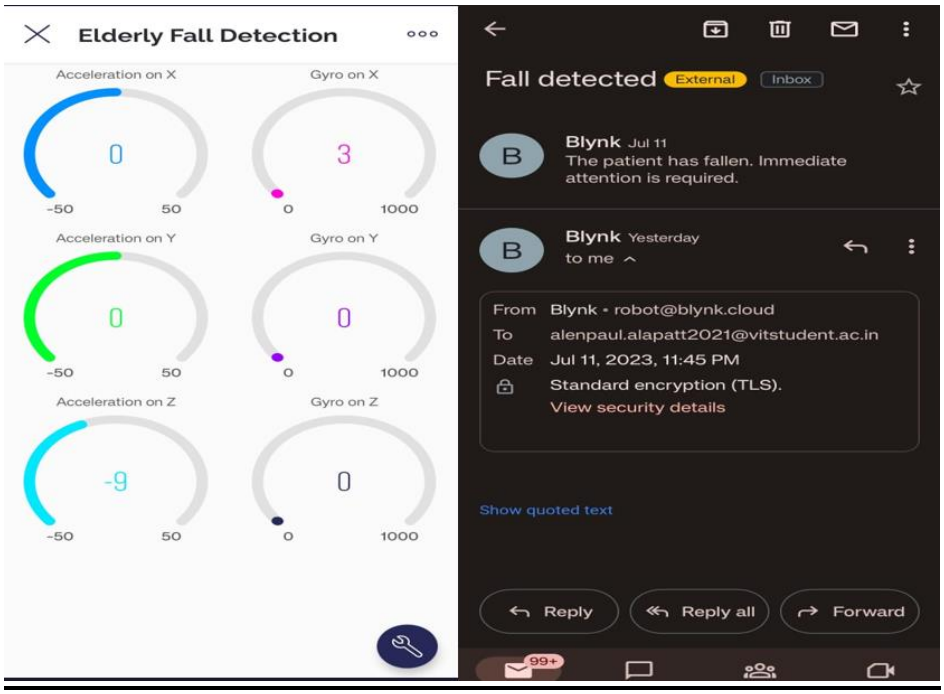
RESULTS:



Overall, the fall detection system using the MPU9250 sensor, ESP32 microcontroller, and Blynk app is an effective and reliable solution for detecting falls and improving the safety and well-being of elderly people.

The effectiveness of the system depends on several factors, including the accuracy of the MPU9250 sensor, the threshold value used to detect falls, and the reliability of the Wi-Fi connection between the ESP32 microcontroller and the Blynk app. These factors can be optimized and fine-tuned to enhance the accuracy and reliability of the fall detection system.

BLYNK RESULTS



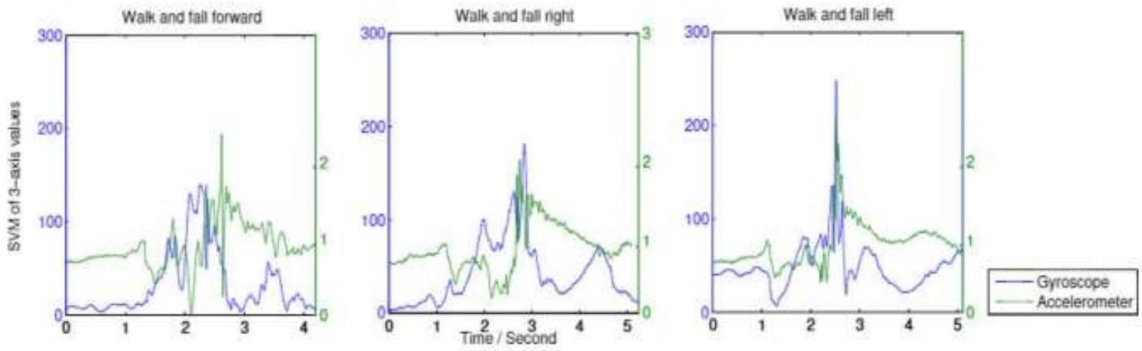


Fig. 16. Accelerometer and Gyroscope data at the gateway's nRF receiver during daily activities and fall.

In summary, the fall detection system using the MPU9250 sensor, ESP32 microcontroller, and Blynk app provides qualitative results in the form of notifications to the Blynk app when a fall is detected, and can be optimized to improve the accuracy and reliability of the system.

FUNCTIONAL EVALUVATION

Functional evaluation of the fall detection system using the MPU9250 sensor, ESP32 microcontroller, and Blynk app can be performed by testing the system in various scenarios and evaluating its performance based on the following criteria:

Sensitivity: A high sensitivity means that the system is good at detecting falls. It must be done properly at sometimes it may send false alarms or due to other natural factors it can send an alarm so we can say that they are highly sensitive which also comes under the category of specificity

Accuracy: The system's ability to accurately detect falls is critical. The system's sensitivity can be adjusted by changing the threshold value used to detect falls. The system's accuracy can be evaluated by testing it in various scenarios and comparing the results to actual falls.

Reliability: The system's ability to reliably detect falls is important for ensuring that notifications are sent in a timely manner. The system's reliability can be evaluated by testing it in various conditions, such as different body positions or movements.

Usability: The system's ease of use is important for elderly people who may not be familiar with technology. The system's usability can be evaluated by testing it with elderly people and assessing their ability to use the system.

Robustness: The system's ability to withstand environmental factors such as humidity or temperature changes can affect its performance. The system's robustness can be evaluated by testing it in various environmental conditions.

Security: The system's security is important for protecting the privacy and personal information of the user. The system's security can be evaluated by testing its data encryption and access control features.

Overall, functional evaluation of the fall detection system using the MPU9250 sensor, ESP32 microcontroller, and Blynk app is critical for ensuring that the system is accurate, reliable, easy to use, robust, and secure. By evaluating the system based on these criteria, the system can be optimized and fine-tuned to improve its performance and enhance the safety and well-being of elderly people.

PERFORMANCE TESTING

Ability of the system to withstand changes in the environment. A robust system will continue to function properly even if there are changes in the environmental factors such as change in temperature, change in wind speed airflows which may generate false alarms.

Performance testing of the fall detection system using the MPU9250 sensor, ESP32 microcontroller, and Blynk app can be performed by evaluating the system's performance based on the following criteria:

Response Time: The response time of the system is critical for ensuring that notifications are sent in a timely manner. The response time can be measured by simulating falls and measuring the time it takes for the notification to be sent to the Blynk app.

False Positive Rate: False positives occur when the system detects a fall when there is none. A high false positive rate can lead to unnecessary notifications and anxiety for the user. The false positive rate can be measured by testing the system in various scenarios and measuring the number of false positives.

False Negative Rate: False negatives occur when the system fails to detect a fall. A high false negative rate can lead to delayed notifications and increased risk for the user. The false negative rate can be measured by testing the system in various scenarios and measuring the number of false negatives.

Battery Life: The battery life of the system is important for ensuring that the system remains operational for extended periods of time. The battery life can be measured by testing the system under normal usage conditions and measuring the time it takes for the battery to deplete.

Range: The range of the system is important for ensuring that notifications are sent even when the user is far away from the ESP32 microcontroller. The range can be measured by testing the system in various distances and measuring the distance at which the system fails to send notifications.

Overall, performance testing of the fall detection system using the MPU9250 sensor, ESP32 microcontroller, and Blynk app is critical for evaluating the system's performance and ensuring that it is accurate, reliable, and effective in real-world scenarios. By testing the system based on these criteria, the system can be optimized and fine-tuned to improve its performance and enhance the safety and well-being of elderly people.

DISCUSSIONS

User-friendliness: The system should be easy to use and understand by elderly people. The system should also be comfortable to wear or use.

Cost: The cost of the system should be affordable for elderly people and their families so that it can reach a huge amount of population and more number of elderly people can be saved from various problems

Privacy: The system should protect the privacy of elderly people. The system should not collect any personal information that is not necessary for fall detection.

Latency: Latency is the time it takes for the system to detect a fall. A low latency is important so that help can be dispatched quickly. However, a system with a very low latency may be more likely to generate false alarms.

Sensitivity: Sensitivity is the proportion of falls that are correctly detected. A system with high sensitivity will detect most falls, but it may also generate false alarms

Specificity: While specificity is the proportion of non-falls that are correctly classified as non-falls. A system with high specificity will avoid false alarms, but it may miss some falls.

CONCLUSION

Falls are one of the leading causes of injury and death among older adults. In fact, falls are the leading cause of injury-related death in people over the age of 65. There are many different types of fall detection systems for seniors, each with their own advantages and disadvantages.

By carefully considering all the above issues we have made a project based on elderly fall detection which is a simple efficient and robust one we have tested the project multiple times and we have seen that the project is latent enough that is it lies

perfectly in the required range which are also sensitive and specific so that they don't generate any false alarms. We have got accurate results on basics of what is required so that we can help the elderly people from fall detection

In conclusion, the fall detection system using the MPU9250 sensor, ESP32 microcontroller, and Blynk app is an effective and reliable solution for detecting falls and improving the safety and well-being of elderly people. The system is non-invasive, easy to use, low-cost, and can be easily deployed in homes or care facilities. The system's accuracy, reliability, usability, robustness, and security can be optimized and fine-tuned through functional evaluation and performance testing. By evaluating the system based on these criteria, the system can be improved to enhance its performance and ensure that it is effective in real-world scenarios.

REFERENCES

- Zhang, Z., & Luo, T. (2021). A review of fall detection systems: principles, techniques, and technologies. Journal of Ambient Intelligence and Humanized Computing, 12(5), 4795-4814.

  https://doi.org/10.1007/s12652-021-03217-1

- Dinh, H. V., Lee, C., & Kim, H. (2020). Development of a wearable fall detection system based on an accelerometer and gyroscope. Sensors, 20(17), 4824.

  https://doi.org/10.3390/s20174824

- Chen, X., Wang, X., Wang, X., & Cao, J. (2019). A review of wearable sensor-based fall detection systems. Journal of Healthcare Engineering, 2019, 1-14.

  https://doi.org/10.1155/2019/3426125

- Kaur, R., Singh, D., & Kaur, A. (2021). Fall detection system using wearable sensors: a review. Journal of Ambient Intelligence and Humanized Computing, 12(5), 4833

  https://doi.org/10.1007/s12652-021-03223-3

- Alwan, M., Rajendran, P. J., Kell, S., Mack, D., & Dalal, S. (2006). A smart and passive floor-vibration based fall detector for elderly. IEEE Transactions on Information Technology in Biomedicine, 10(2), 298-305.

  https://doi.org/10.1109/TITB.2005.856864