IoT BASED FALL DETECTION SYSTEM

PROJECT REPORT

TEAM MEMBERS:

Vishalini P – 20BML0045

Varshinne P – 20BML0041

Kavinesh K – 20BCR7022

Hemanth Adithya – 20BCE7229

INTRODUCTION

1.1 Overview

The project proposed is a cost-effective and reliable fall detection system to detect fall in elderly patients and to alert the care-taker or the physician of the patient's condition for help and support. The proposed work incorporates ESP32 and MPU6050 sensor module with an inbuilt accelerometer and gyroscope to detect the change in acceleration and body orientation of the patient respectively. By coupling the accelerometer with gyroscope, the accuracy of the system was improved due to reduction in false positives and true negatives. The pre-set threshold conditions are programmed in the ESP32 such that it checks for change in acceleration and orientation accordingly and alerts on detecting fall. The collected fall detection result will be published from Wokwi to IBM cloud. The data is then sent to Node-red to display the alert message and from node-red the alert message is sent to the “Fall Alert” app created using MIT-APP inventor. Care takers can directly view the status of the patient using the app. Moreover, this device requires less implementation cost and provides a quick response.

1.2 Purpose

One of the most frequent reasons why elderly individuals need medical treatment is after falling. To reduce the risk of elderly people getting hurt from falls, medical attention must be offered quickly away. As a result, a reliable fall detection system can help detect falls in older citizens and connect them to the caretaker or the closest healthcare institution for support and assistance. For elderly people to utilize the fall detection system more simply, it must be user-friendly. The equipment must also not interfere with or impair an elderly person's regular activities. The system must be durable and cost-effective. The primary goal of this suggested project is to assist old individuals or patients by utilising the MPU6050 sensor module. While the gyroscope establishes direction, the accelerometer provides information on the angular parameter, such as the X, Y, and Z-axis data. In order to establish whether any false triggering is occurring or the user has genuinely fallen, we attach this sensor to a microcontroller that is at odds with it and maintain different acceleration thresholds. If readings exceed thresholds, the Fall Alert app on the Android mobile notifies the caretaker.

LITERATURE SURVEY

2.1 Existing Problem

The most common cause of harm for elderly people is falls. These falls result in numerous debilitating fractures that may eventually lead to complications and death. The majority of seniors (those over 75) fall at least once every year, and 24% of them suffer serious injuries. Health and healthcare expenses will be significantly impacted by this critical public health issue. The expense and workload associated with providing care for senior citizens are rising. The chance of falling increases three times more in those with Alzheimer's disease. The most critical components of a fall detection system that must be taken into account when creating a robust system were highlighted in a standard database structure for fall studies, which also addressed the limitations and obstacles. A further challenge for training systems is obtaining autumn activity patterns. The fall detection capabilities of these systems are successful. However, concentrating just on rapid acceleration might lead to a lot of false positives from fall-like actions like jogging and helping down rapidly. Additionally, to find the fall in earlier research, sophisticated algorithms like the Markov model and support vector machine (SVM) were applied.

2.2 Proposed solution

The sensor for the new gadget we suggest is the MPU6050 Accelerometer and Gyro Chip. Accelerometer and gyroscope were used to measure the faller's acceleration and body tilt, respectively. The outcome will be sent and stored in the IBM cloud. Through Node-red, cloud data is delivered to the “Fall Alert” created using the MIT-APP inventor. A very user-friendly app has been created. Family members or caretakers of the patient will be able to find out the patient's condition and if the patient has fallen or not.

THEORETICAL ANALYSIS

Fall Detection Algorithm

Both dynamic and static acceleration are contained in the total sum acceleration vector, Acc. Components, as shown in Eq. (1)11, are computed from sampling data.

Acc=sqrt((Ax)^2 + (Ay)^2 + (Az)^2).....(1)

Where, respectively, Ax, Ay, and Az represent the acceleration along the x, y, and z axes.

The angular velocity is estimated from sampled data in a manner similar to how the acceleration is, as shown in Eq. (2)

w = sqrt((Wx)^2 + (Wy)^ 2 + (Wz)^2).....(2)

Where Wx, Wy, and Wz, respectively, represent the acceleration along the x, y, and z axes.

Tri-axial accelerometer readings at rest show constant acceleration magnitude, Acc, and zero degrees per second of angular velocity. When a subject falls, the acceleration changes quickly, and the angular velocity generates a range of signals along the fall direction.

The Fall Index (Acc) will overlook falls that occur slowly since it demands a high sample frequency and rapid acceleration changes. As a result, Acc is rarely employed until we wish to compare the performance of our systems to that of earlier research that used the same locations but different speeds and accelerations.
The following is a derivation of the lower and higher fall thresholds for the acceleration and angular velocity needed to detect the fall:

1- Lower fall threshold (LFT):
The signal lower peak refers to the negative peaks for each recorded activity's outcome.(LPVs) values. The level of the lowest magnitude lower fall peak (LFP) ever recorded serves as the LFT for the acceleration signal.

2- Upper fall threshold (UFT):
The signal upper peak values (UPVs) are the positive peaks for the recorded signals for each recorded activity. The level of the smallest magnitude UPV collected was used to set the UFT for each acceleration and angular velocity signal. The peak impact force that a body segment experiences during the impact phase of a fall is related to the UFT.
The two sets of fall detection algorithms that use thresholds are typically based on the LFT comparison and the UFT comparison of acceleration data, respectively.

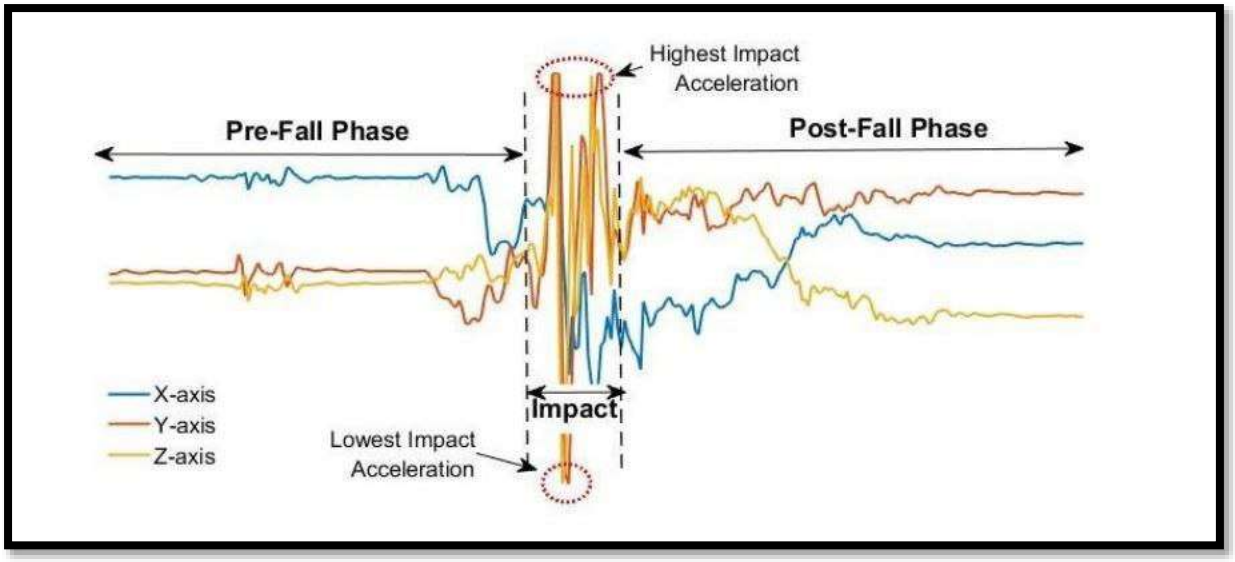


Fig.1. Graph depicting highest and lowest impact acceleration

3.1 Block Diagram

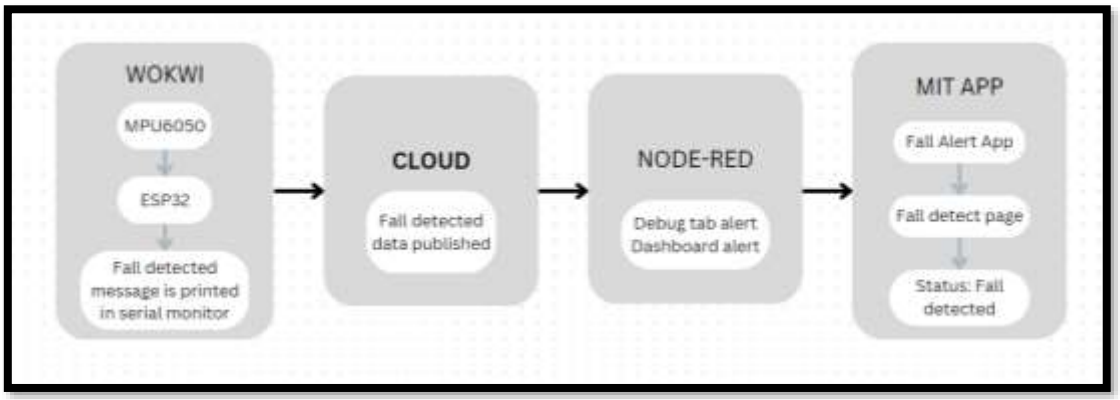


Fig.2. Block diagram

3.2 Hardware / Software designing

Hardware:

ESP32:

The ESP32 is a flexible microcontroller-based development board featuring a dual-core CPU, built-in Wi-Fi and Bluetooth, a wealth of peripherals, vast programming choices, and power economy. With its strong processing powers, smooth communication, and flexibility for quick prototyping and development, it is commonly employed in IoT applications.

MPU6050 sensor module:

The MPU6050 sensor module combines a 3-axis accelerometer and a 3-axis gyroscope into a small, reasonably priced gadget. It can monitor acceleration, tilt, rotation, and angular velocity and has motion sensing capabilities. It is frequently employed in systems for robotics, drones, and motion tracking.

Software: Wokwi, IBM Cloud, Node-red, MIT-APP inventor

EXPERIMENTAL INVESTIGATIONS

Developing a fall-detection system involves several crucial analyses to ensure its effectiveness and reliability. The process typically includes data collection, algorithm development, performance evaluation, validation testing, false alarm mitigation, user experience assessment, and scalability and reliability analysis.

False Alarm Mitigation: False alarms can undermine the usability of a fall-detection system. An analysis is conducted to identify common causes of false alarms, such as abrupt movements, sudden stops, or specific body postures. Techniques like feature engineering, threshold

adjustment, or incorporating contextual information may be employed to minimize false positives.

User Experience Assessment: Analysing the user experience is essential to ensure the system is practical, user-friendly, and acceptable to the target users. This analysis involves obtaining feedback from caregivers, elderly individuals, or other stakeholders who interact with the fall-detection system. It helps identify usability issues, concerns, and suggestions for improvement.

FLOWCHART

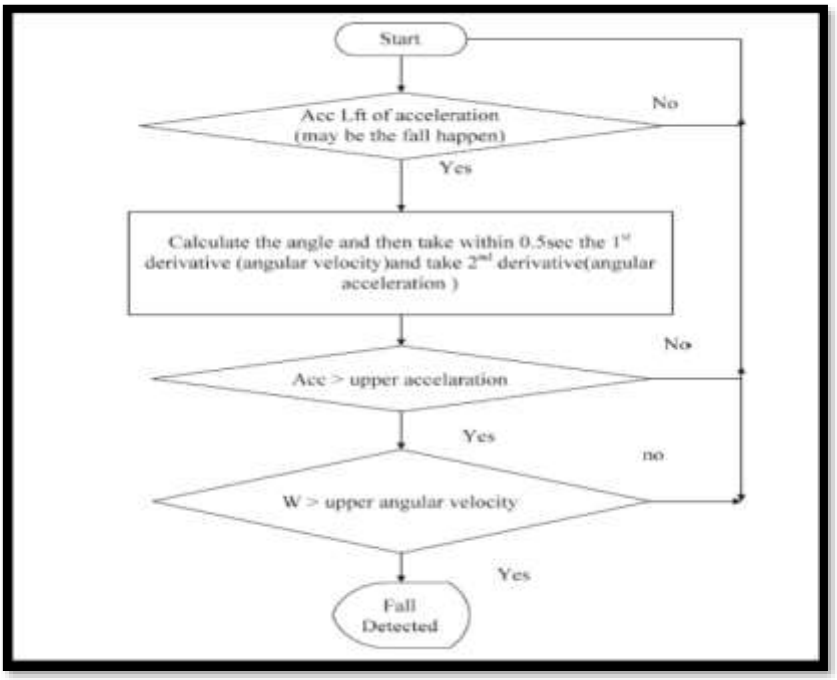


Fig.3. Flowchart

RESULTS

The fig.4 shows that once the sensor reading exceeds the threshold values, a notification for fall detection is displayed on the serial monitor of Wokwi.

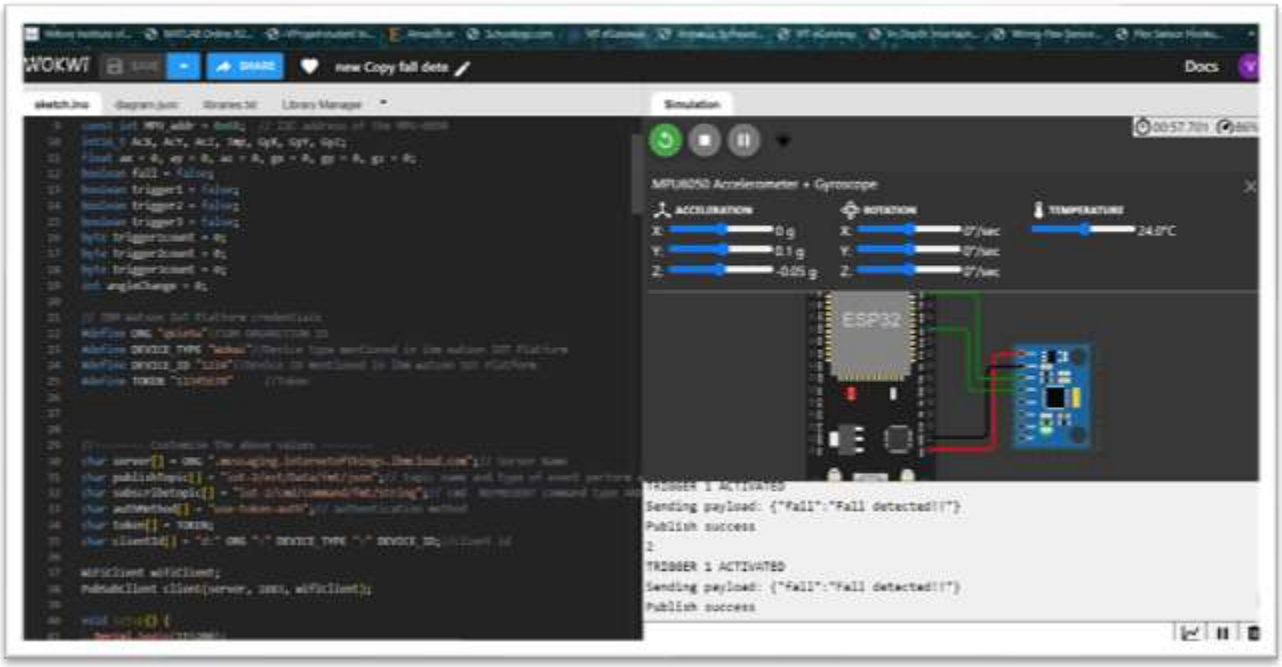
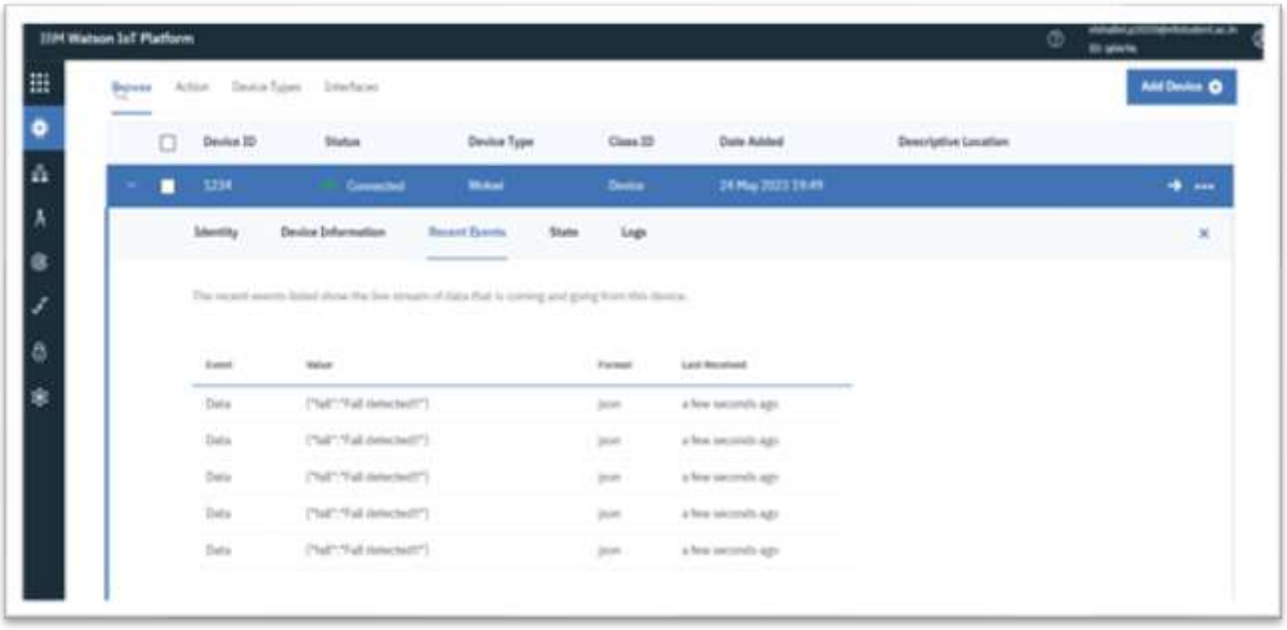
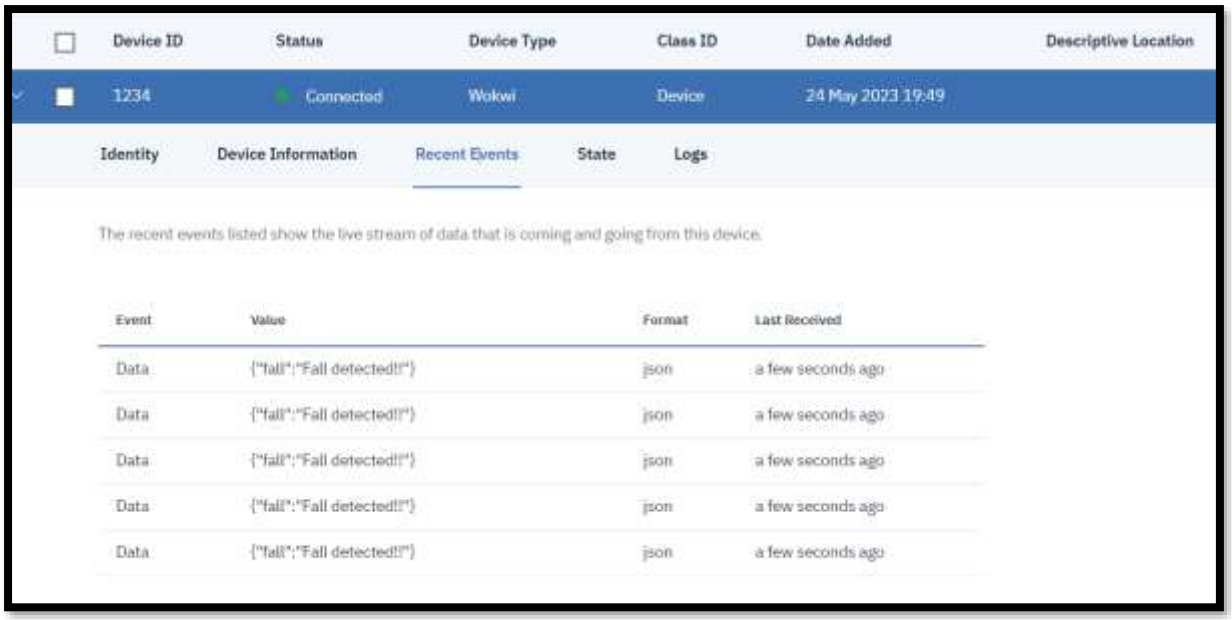


Fig.4. Serial monitor of Wokwi showing the “Fall detected!!” alert



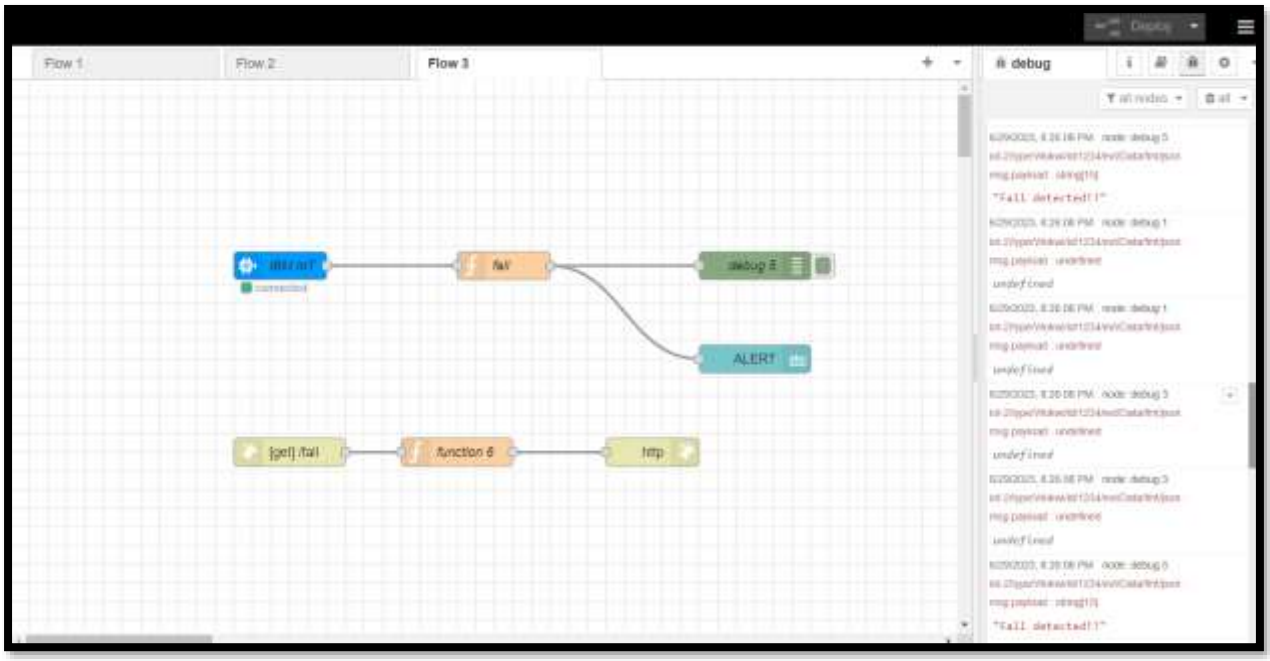
(a)



(b)

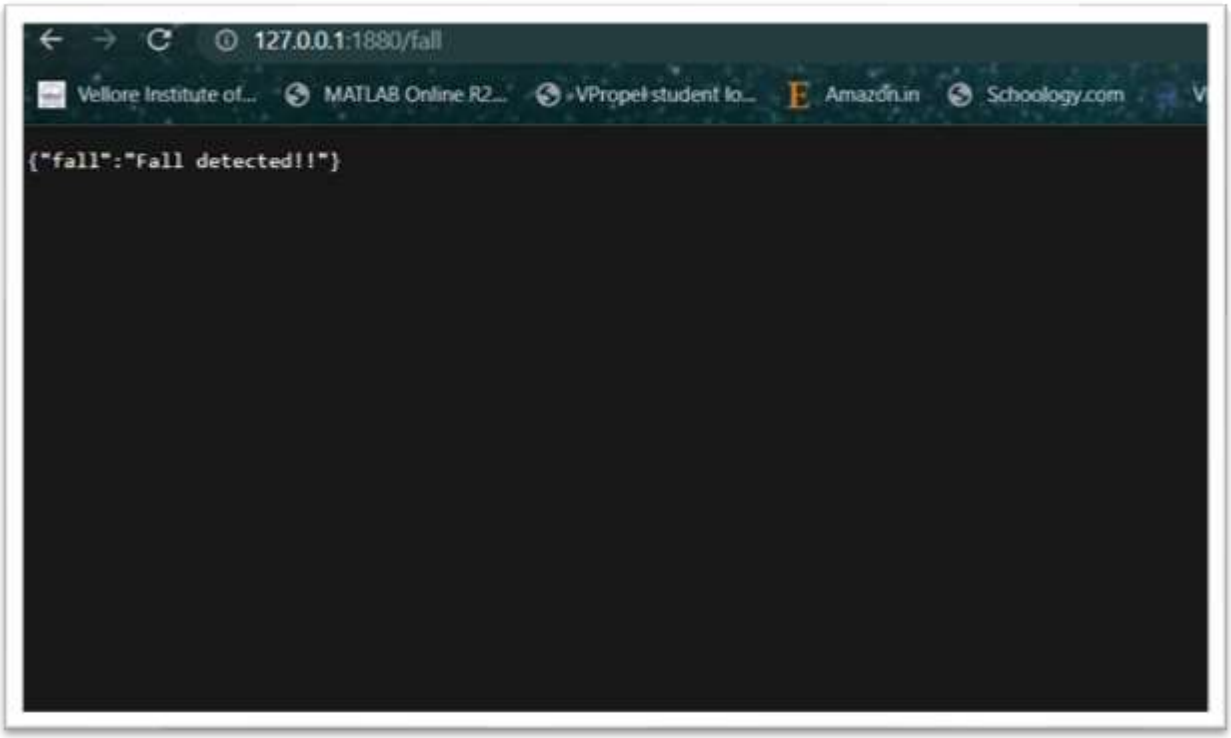
Fig.5.(a) & (b) IBM Cloud output

In fig 5 (a) & (b), it is inferred that the data from Wokwi is successfully published to IBM Cloud and the “Fall detected!!” alert message is published in the recent events tab.



(a) Debug tab

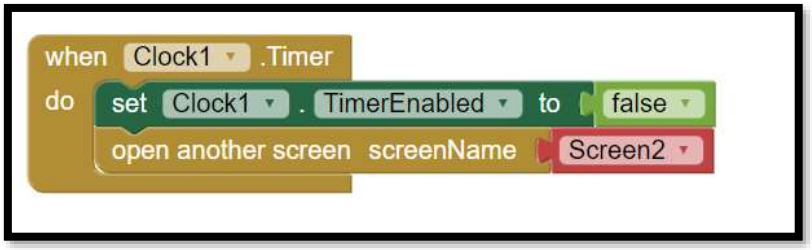
(b)Dashboard of Node-red



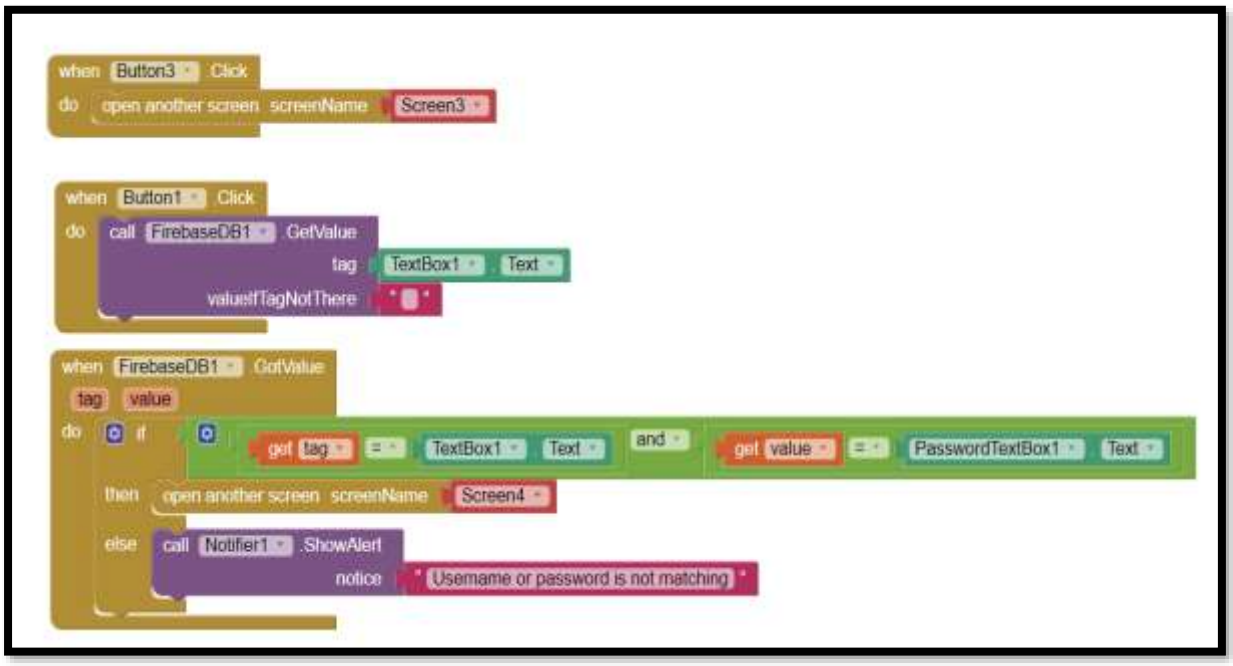
(c)Data published using GET command

Fig.6. (a), (b) & (c) Output from the Node red platform

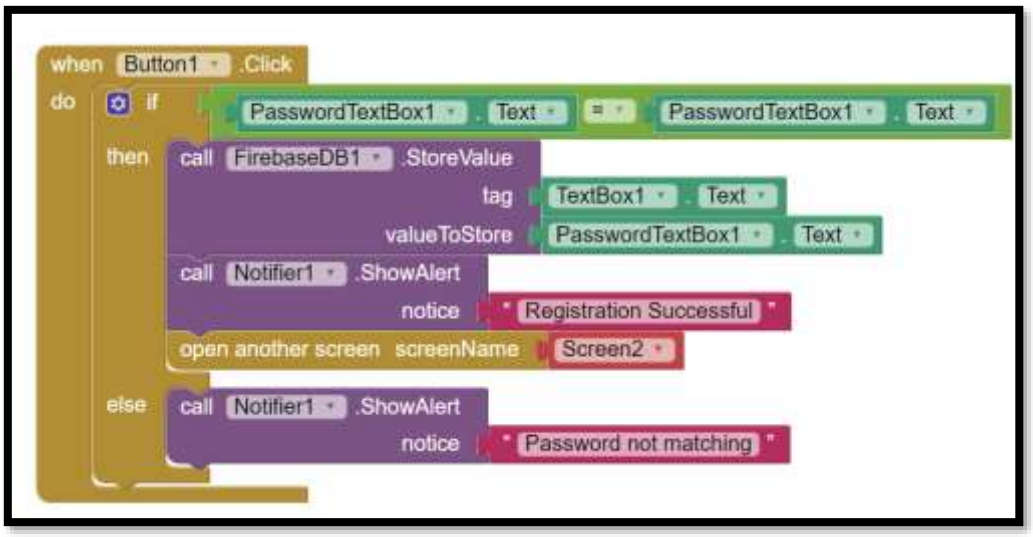
In fig. 6 (a) it is inferred that the data from cloud is again sent to Node-red and the fall detection message is displayed in the debug tab using Debug block. The alert message is also displayed in the dashboard of Node-red in fig.6 (b). The alert message is in turn displayed using the GET command in fig.6 (c).



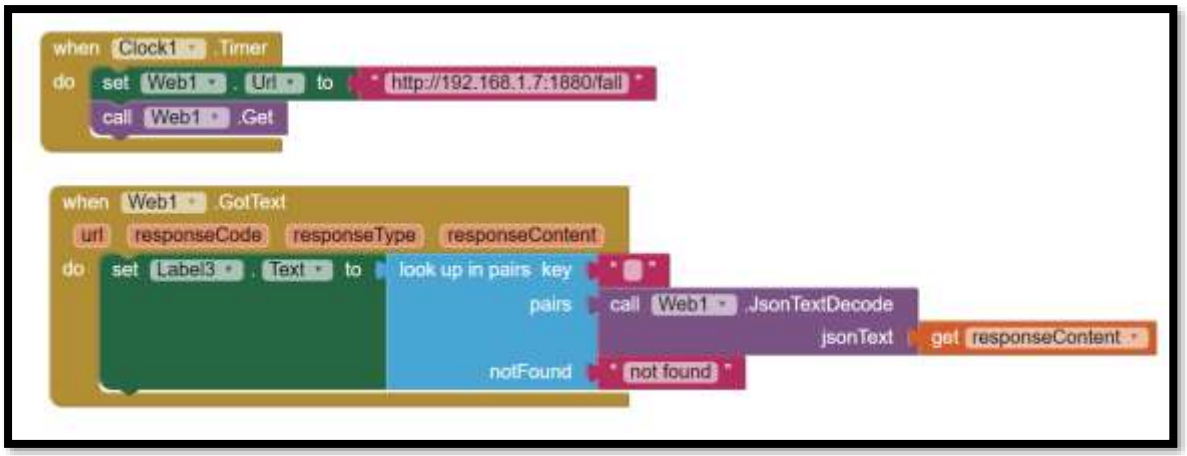
(a)Screen 1



(b)Screen 2



(c)Screen 3

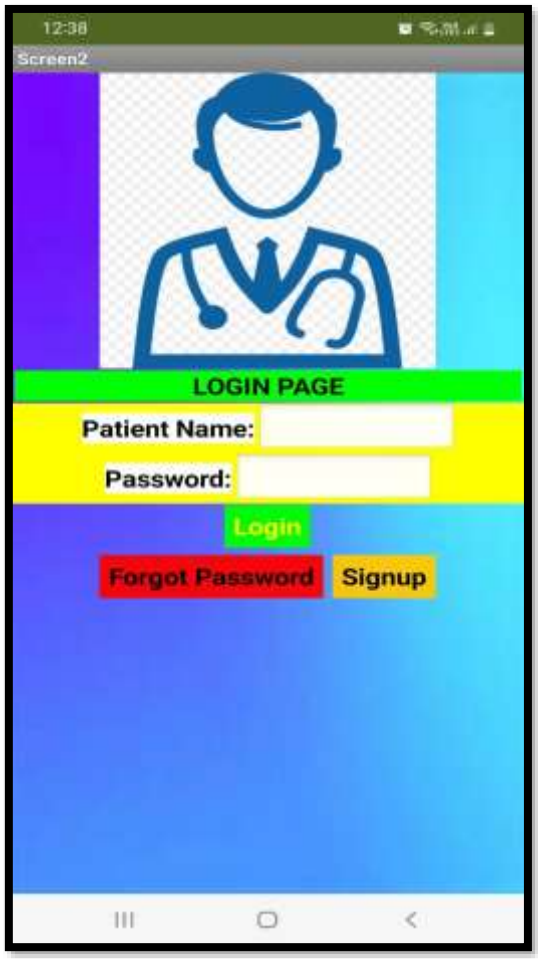


(d)Screen 4

Fig.7. (a), (b), (c) & (d). Screenshots of Blocks from MIT-APP inventor

The fig.7 shows the blocks designed using the MIT-APP inventor to build a Fall Alert app to notify the caretaker/ relative of the patient incase of fall detection.

(a)Welcome page



(b)Login page

(c)Patient name and password registration page

(d)Login page

(e)Fall detection page

Fig.8. (a), (b), (c), (d) & (e). Output screenshots of the Fall Alert app

The fig 8 shows all the screens designed and the fall detection page shows the status as “Fall detected!!” in case the patient has fallen down.

ADVANTAGES

A fall detection system offers several advantages in enhancing safety and providing timely assistance in case of falls. Here are some key advantages of a fall detection system:

Rapid response: Systems for detecting falls are created to act quickly when one occurs. They can send out automated alerts to authorised contacts, emergency agencies, or carers, assuring prompt assistance. This quick action is essential for reducing the risks of injury or other issues following a fall.

Independence and confidence: Fall detection systems allow people to preserve their freedom and confidence, especially elderly persons or people with mobility challenges. People may go about their regular activities with less fear when they have the certainty that assistance will be sent out automatically in the case of a fall.

24/7 monitoring: Fall detection systems are capable of continuously monitoring for falls, offering protection and continual watch. This ongoing monitoring makes sure that falls are caught even when the person is alone or at night, when urgent aid might not be possible.

Versatile deployment: Fall detection systems can be deployed in various settings, including homes, assisted living facilities, hospitals, or even wearable devices. They can be tailored to meet the specific needs of different individuals and environments, making them adaptable and versatile.

Personalized alerts and notifications: Fall detection systems can be configured to send alerts and notifications to multiple recipients, such as family members, caregivers, or medical professionals. This enables prompt communication and coordination among the concerned parties, ensuring that appropriate assistance is provided.

DISADVANTAGES

False alerts: On occasion, fall detection systems may produce false alarms by mistaking motions or actions that aren't falls for falls. This may result in pointless notifications and reactions, which could irritate the user as well as the carers. False alarms can also make people less responsive to true fall signals and decrease their faith in the system.

Limited accuracy: Fall detection systems are not perfect and can misidentify falls. The system may have a harder time detecting some falls, such those with slow or steady descents. The user's body position, their attire, and the location of the device can all have an impact on how accurate the system is.

APPLICATIONS

1. Elderly care in facilities and homes.
2. Monitoring patients in hospitals and rehabilitation centres.
3. Enhancing workplace safety in industries.
4. Incorporating into sports and physical activity trackers.
5. Improving safety in public spaces and transportation.

CONCLUSION

The several fall-feature parameters of the 6-axes acceleration were introduced and applied according to the algorithm. Possible falls were chosen through the simple threshold and then applied to the MPU to solve the problems such as deviation of interpersonal falling behavioural patterns and similar fall actions. The parameters of upper and lower of acceleration and velocity have been adjusted to give best fall detection with sensitivity, specificity, and accuracy which were over than 95 %. These results demonstrate the reduction of the computing effort and resources, compared to those of using all the events applied. The proposed algorithms were very simple because they depend on a simple sensor (measure the angle) and the program calculates the angular velocity and acceleration. In the future, if the proposed algorithms are implemented to the embedded system, its performance will be tested in real-time.

FUTURE SCOPE

Improved accuracy: Future fall detection systems are anticipated to have more sophisticated sensors and algorithms, which will increase the accuracy in detecting falls. This may entail using machine learning and artificial intelligence approaches to distinguish between falls and regular activities more effectively, lowering false alarm rates and boosting dependability.

Integration with wearable technology: Wearable gadgets, such fitness trackers or smartwatches, are gaining popularity. Future fall detection systems may use wearable technology to easily incorporate fall detection features into commonplace products, doing away with the need for separate specialised equipment.

BIBLIOGRAPHY

1. Hwang, J.Y., Kang, J.M., Jang, Y.W., Kim, H.C.: Development of novel algorithm and real time monitoring ambulatory system using Bluetooth module for fall detection in the elderly. In: Engineering in Medicine and Biology Society, IEMBS 2004, 26th Annual International Conference of the IEEE, vol. 1, pp. 2204–2207, September 2004
2. B. Najafi, K. Aminian, F. Loew, Y. Blanc and P. A. Robert, "Measurement of stand-sit and sit-and transitions using a miniature gyroscope and its application in fall risk evaluation in the elderly", IEEE Trans. Biomed. Eng., vol. 49, no. 8, pp. 843-851, 2002.
3. E. Mattila, I. Korhonen, J. Merilahti, A. Nummela, M. Myllymaki, and H. Rusko, “A concept for personal wellness management based on activity monitoring,” in Pervasive Computing Technologies for Healthcare,2008. Pervasive Health 2008. Second International Conference on, 302008-feb. 1 20 08, pp. 32 –36.
4. M. Alwan, D. Mack, S. Dalal, S. Kell, B. Turner, and R. Felder, “Impact of passive in-home health status monitoring technology in home health: Outcome pilot,” in Distributed Diagnosis and Home Healthcare, 2006.D2H2. 1st Transdisciplinary Conference on, 2006, pp. 79 –82.
5. Zhang, T., Wang, J., Xu, L., Liu, P.: Fall detection by wearable sensor and one-class SVM algorithm. In: Huang, D.-S., Li, K., Irwin, G.W. (eds.) ICIC 2006. LNCIS, vol. 345, pp. 858–863. Springer, Heidelberg (2006)

APPENDIX

(a) Source code

Link for source code: https://wokwi.com/projects/368604930638147585

CODE:

```
#include <Wire.h>

#include <Adafruit_MPU6050.h>

#include <WiFi.h>
```

```
#include <PubSubClient.h>
const int MPU_addr = 0x68;  // I2C address of the MPU-6050
int16_t AcX, AcY, AcZ, Tmp, GyX, GyY, GyZ;
float ax = 0, ay = 0, az = 0, gx = 0, gy = 0, gz = 0;
boolean fall = false;
boolean trigger1 = false;
boolean trigger2 = false;
boolean trigger3 = false;
byte trigger1count = 0;
byte trigger2count = 0;
byte trigger3count = 0;
int angleChange = 0;
// IBM Watson IoT Platform credentials
#define ORG "q6ie9a"//IBM ORGANITION ID
#define DEVICE_TYPE "Wokwi"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);
void setup() {
Serial.begin(115200);
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1 register
```

```
  Wire.write(0);     // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
  Serial.println("Wrote to IMU");
  wificonnect();
  mqttconnect();
}
void loop() {
mpu_read();
  ax = (AcX - 2050) / 16384.00;
  //Serial.println(ax);
  ay = (AcY - 77) / 16384.00;
  //Serial.println(ay);
  az = (AcZ - 1947) / 16384.00;
  gx = (GyX + 270) / 131.07;
  gy = (GyY - 351) / 131.07;
  gz = (GyZ + 136) / 131.07;
  float Raw_Amp = pow(pow(ax, 2) + pow(ay, 2) + pow(az, 2), 0.5);
  int Amp = Raw_Amp * 10;
  Serial.println(Amp);
  if (Amp <= 2 && trigger2 == false) {
    trigger1 = true;
    Serial.println("TRIGGER 1 ACTIVATED");
  }
  if (trigger1 == true) {
    trigger1count++;
    if (Amp >= 12) {
      trigger2 = true;
      Serial.println("TRIGGER 2 ACTIVATED");
      trigger1 = false;
      trigger1count = 0;
    }
  }
```

```
  if (trigger2 == true) {
    trigger2count++;
    angleChange = pow(pow(gx, 2) + pow(gy, 2) + pow(gz, 2), 0.5);
    if (angleChange >= 3 || Amp >= 15) {
      trigger3 = true;
      Serial.println("TRIGGER 3 ACTIVATED");
      trigger2 = false;
      trigger2count = 0;
    }
  }
  if (trigger3 == true) {
    trigger3count++;
    if (Amp <= 2 && trigger3count >= 4) {
      fall = true;
      Serial.println("FALL DETECTED");
      trigger3 = false;
      trigger3count = 0;
    } else if (Amp <= 2 && trigger3count < 4) {
      trigger3count = 0;
    }
  }
  if (Amp<=2) {
    PublishData("Fall detected!!");
    fall = false;
  }
  delay(100);
}
void mpu_read() {
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr, 14, true);  // request a total of 14 registers
```

```
  AcX = Wire.read() << 8 | Wire.read();  // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY = Wire.read() << 8 | Wire.read();  // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ = Wire.read() << 8 | Wire.read();  // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp = Wire.read() << 8 | Wire.read();  // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX = Wire.read() << 8 | Wire.read();  // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY = Wire.read() << 8 | Wire.read();  // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ = Wire.read() << 8 | Wire.read();  // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
}
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");
  WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void mqttconnect() {
  if (!client.connected()) {
    Serial.print("Reconnecting client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {
      Serial.print(".");
      delay(500);
    }

     initManagedDevice();
```

```
    Serial.println();
  }
}
void initManagedDevice() {
  if (client.subscribe(subscribetopic)) {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {
    Serial.println("subscribe to cmd FAILED");
  }
}
void PublishData(String fall) {
  mqttconnect();
  String payload = "{\"fall\":\"";
  payload += fall;
  payload += "\"}";
  Serial.print("Sending payload: ");
  Serial.println(payload);
  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish success");
  } else {
    Serial.println("Publish failed");
  }
  delay(2000);
}
```
