| **TITLE: Fall Detection using MPU6050 Accelerometer** | NAME:SHEENA S<br>ROLL NO:20L141<br>WSN ASSIGNMENT PRESENTATION |
|---|---|

## OBJECTIVE:

Design and implement a fall detection system using NodeMCU and MPU6050 to:

**Detect Falls**: Utilize the MPU6050 accelerometer and gyroscope to monitor motion and orientation changes, identifying sudden, significant deviations that could indicate a fall.

**Real-time Monitoring**: Implement a real-time monitoring system to continuously analyze the sensor data and apply the fall detection algorithm. The system should operate efficiently and detect falls as quickly as possible.

## SPECIFICATION:

### 1. Node MCU:

The NodeMCU is an open-source development board that is built around the ESP8266 microcontroller. It provides a convenient platform for IoT (Internet of Things) and embedded applications. Below are the specifications and features of a typical NodeMCU development board:

1. Microcontroller:

   ESP8266 Wi-Fi module, usually ESP-12E or ESP-12F.

2. Processor:

   32-bit Tensilica L106 RISC microcontroller core.

3. Clock Speed:

   Up to 80 MHz.

4. Flash Memory:

   Usually 4MB (32Mbit) of Flash memory for program storage.

5. RAM:

   Typically 80 KB available for user applications.

6. Wireless Connectivity:

   Integrated 2.4GHz Wi-Fi (802.11 b/g/n) supporting WPA/WPA2 security.

7. GPIO Pins:

   17 GPIO pins for interfacing with sensors, actuators, displays, and other peripherals.

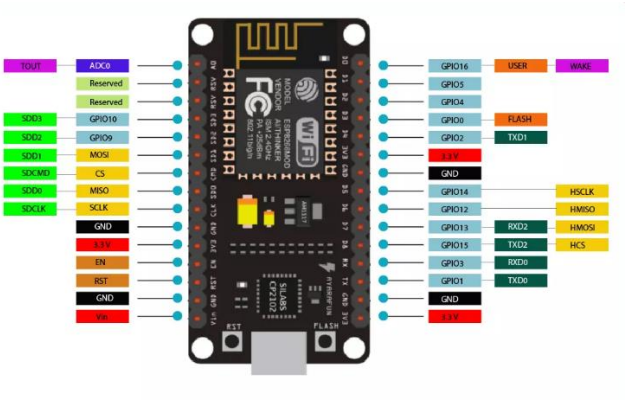



FIG 1 NODE MCU

## 2. MPU 6050:

The MPU-6050 is a popular 6-axis motion tracking device manufactured by InvenSense (now TDK). It combines a 3-axis gyroscope and a 3-axis accelerometer in a single chip. Here are the typical specifications and features of the MPU-6050:

Voltage supply: 3.3 - 5 V DC

Logic level: 3.3 V

Degrees of freedom: 6 x

Interface: I²C

Built-in chip: 16-bit AD converter

Pins: 8 x

Pin spacing: 2.54 mm

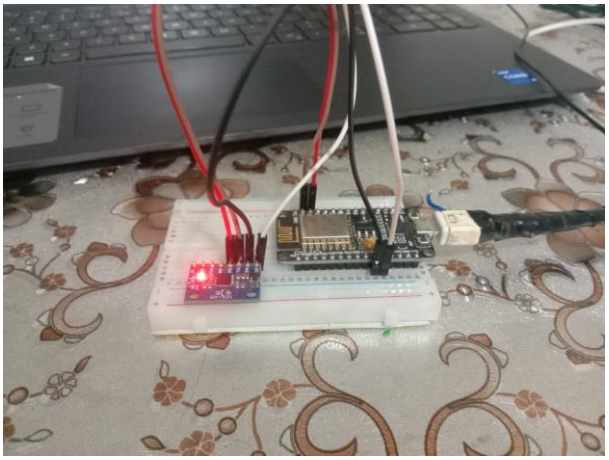
Gyroscopic range: ±250, ±500, ±1000, ±2000 °/ s (16 bits)

3-axis sensing: Full-scale range of ±2g, ±4g, ±8g, or ±16g

Sensitivity: 16384, 8192, 4096, or 2048 LSBs per g


FIG 2 MPU 6050

## HARDWARE SETUP:



## WORKING:

When a fall is detected,it is been sensed by the MPU 6050 module,the data is then displayed in the serial monitor of the Arduino IDE,it is then linked to a message generating website,that sends a message to the user's mobile phone,on detecting a fall.

## SOFTWARE CODE:

```
// IoT based Fall Detection using NodeMCU and MPU6050 Sensor
//https://iotprojectsideas.com
#include <Wire.h>
#include <ESP8266WiFi.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; //stores if a fall has occurred
boolean trigger1=false; //stores if first trigger (lower threshold) has occurred
boolean trigger2=false; //stores if second trigger (upper threshold) has occurred
boolean trigger3=false; //stores if third trigger (orientation change) has occurred
byte trigger1count=0; //stores the counts past since trigger 1 was set true
byte trigger2count=0; //stores the counts past since trigger 2 was set true
byte trigger3count=0; //stores the counts past since trigger 3 was set true
int angleChange=0;
// WiFi network info.
const char *ssid = "vivo 1904"; // Enter your Wi-Fi Name
const char *pass = "jesus1234"; // Enter your Wi-Fi Password
void send_event(const char *event);
const char *host = "maker.ifttt.com";
const char *privateKey = "gj1ervZVYyoqCWe963yuFm2nk9CfIazd7bh-Sl2m0mI";
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");
  Serial.println("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");            // print ... till not connected
  }
  Serial.println("");
  Serial.println("WiFi connected");
 }
void loop(){
 mpu_read();
 ax = (AcX-2050)/16384.00;
 ay = (AcY-77)/16384.00;
 az = (AcZ-1947)/16384.00;
 gx = (GyX+270)/131.07;
 gy = (GyY-351)/131.07;
 gz = (GyZ+136)/131.07;
 // calculating Amplitute vactor for 3 axis
 float Raw_Amp = pow(pow(ax,2)+pow(ay,2)+pow(az,2),0.5);
 int Amp = Raw_Amp * 10;  // Mulitiplied by 10 bcz values are between 0 to 1
 Serial.println(Amp);
 if (Amp<=2 && trigger2==false){ //if AM breaks lower threshold (0.4g)
   trigger1=true;
   Serial.println("TRIGGER 1 ACTIVATED");
   }
 if (trigger1==true){
   trigger1count++;
   if (Amp>=12){ //if AM breaks upper threshold (3g)
     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); Serial.println(angleChange);
   if (angleChange>=30 && angleChange<=400){ //if orientation changes by between 80-100 degrees
     trigger3=true; trigger2=false; trigger2count=0;
     Serial.println(angleChange);
     Serial.println("TRIGGER 3 ACTIVATED");
       }
   }
 if (trigger3==true){
    trigger3count++;
    if (trigger3count>=10){
       angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);
       //delay(10);
       Serial.println(angleChange);
       if ((angleChange>=0) && (angleChange<=10)){ //if orientation changes remains between 0-10 degrees
           fall=true; trigger3=false; trigger3count=0;
           Serial.println(angleChange);
             }
       else{ //user regained normal orientation
          trigger3=false; trigger3count=0;
          Serial.println("TRIGGER 3 DEACTIVATED");
       }
     }
  }
 if (fall==true){ //in event of a fall detection
   Serial.println("FALL DETECTED");
   send_event("fall_detect");
   fall=false;
   }
 if (trigger2count>=6){ //allow 0.5s for orientation change
   trigger2=false; trigger2count=0;
   Serial.println("TRIGGER 2 DECACTIVATED");
   }
 if (trigger1count>=6){ //allow 0.5s for AM to break upper threshold
   trigger1=false; trigger1count=0;
   Serial.println("TRIGGER 1 DECACTIVATED");
   }
  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 send_event(const char *event)
{
  Serial.print("Connecting to ");
  Serial.println(host);
    // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("Connection failed");
    return;
  }
    // We now create a URI for the request
  String url = "/trigger/";
  url += event;
  url += "/with/key/";
  url += privateKey;
  Serial.print("Requesting URL: ");
  Serial.println(url);
  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +"Host: " + host + "\r\n" + "Connection: close\r\n\r\n");
  while(client.connected())
  {
    if(client.available())
    {
      String line = client.readStringUntil('\r');
      Serial.print(line);
    } else {
      // No data yet, wait a bit
      delay(50);
    };
  }
  Serial.println();
  Serial.println("closing connection");
  client.stop();
}
```

# RESULT:

Output  Serial Monitor ×

Message (Enter to send message to 'NodeMCU 1.0 (ESP-12E Module)' on 'COM5')

```
21:21:16.364 -> 1
21:21:16.364 -> TRIGGER 1 ACTIVATED
21:21:16.486 -> 1
21:21:16.486 -> TRIGGER 1 ACTIVATED
21:21:16.486 -> TRIGGER 1 DECACTIVATED
21:21:16.597 -> 1
21:21:16.597 -> TRIGGER 1 ACTIVATED
21:21:16.667 -> 1
21:21:16.667 -> TRIGGER 1 ACTIVATED
21:21:16.779 -> 1
21:21:16.779 -> TRIGGER 1 ACTIVATED
21:21:16.885 -> 1
21:21:16.885 -> TRIGGER 1 ACTIVATED
21:21:16.983 -> 1
21:21:16.983 -> TRIGGER 1 ACTIVATED
21:21:17.097 -> 1
21:21:17.097 -> TRIGGER 1 ACTIVATED
21:21:17.097 -> TRIGGER 1 DECACTIVATED
21:21:17.205 -> 1
21:21:17.205 -> TRIGGER 1 ACTIVATED
21:21:17.302 -> 1
21:21:17.302 -> TRIGGER 1 ACTIVATED
21:21:17.410 -> 1
21:21:17.410 -> TRIGGER 1 ACTIVATED
21:21:17.524 -> 1
21:21:17.524 -> TRIGGER 1 ACTIVATED
21:21:17.604 -> 1
21:21:17.604 -> TRIGGER 1 ACTIVATED
21:21:17.703 -> 1
21:21:17.703 -> TRIGGER 1 ACTIVATED
```