WORKING:

This program is designed for an IoT-based fall detection system using a NodeMCU (ESP8266) and an MPU6050 sensor. It detects falls by monitoring the sensor data and sends an event to the IFTTT (If This Then That) platform, which can trigger various actions or notifications, such as sending an alert to a caregiver.The program includes the necessary libraries for I2C communication (Wire), ESP8266 WiFi connectivity (ESP8266WiFi), and global variables for sensor data and flags to track fall detection events.The Wire library is initialized to enable communication with the MPU6050 sensor using the I2C protocol.The program specifies the SSID and password for the user's hotspot network to connect the NodeMCU to the internet. It attempts to connect to the WiFi network and waits until the connection is established.The MPU6050 sensor is initialized by configuring its Power Management Register (PWR_MGMT_1) to wake it up.n the main loop, the program continuously reads data from the MPU6050 sensor and performs fall detection based on that data.The function mpu_read() reads accelerometer and gyroscope data from the MPU6050 sensor. This data includes the values for acceleration in three axes (AcX, AcY, AcZ) and angular velocity in three axes (GyX, GyY, GyZ).The program has 3 levels of triggers to detect the fall and the activation message is sent to IFTTT if all the three triggers are activated. Each device has a separate private key to activate the applet on IFTTT website.

Explanation of the trigger stages in more depth:

Trigger Stage 1 (trigger1):

The first stage of the fall detection algorithm focuses on the amplitude of the acceleration data (Amp), which represents the overall intensity of motion experienced by the sensor. When Amp falls below a lower threshold (Amp <= 2), it triggers the first stage (trigger1). This threshold is set to detect the initial, subtle changes in acceleration that might indicate a potential fall. Upon activation of trigger1, it prints a message "TRIGGER 1 ACTIVATED" and starts counting the number of iterations (stored in trigger 1 count).

Trigger Stage 2 (trigger2):

If trigger 1 is activated, the program continuously checks Amp and the number of iterations. When Amp exceeds an upper threshold (Amp >= 12), it triggers the second stage (trigger2). This threshold is higher than the first to ensure that the acceleration increase is significant. Upon activation of trigger2, it prints a message "TRIGGER 2 ACTIVATED" and resets trigger1 and its count.

```
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;
    }
}
```

The triggers will get deactivated if there is a change in the orientation.

```
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);
  }
```

Trigger Stage 3 (trigger3): Once trigger2 is activated, the program starts monitoring the change in orientation based on gyroscope data.It calculates the angleChange, which represents the magnitude of the angular change. If angle change falls within a specific range (typically 80-100 degrees), it triggers the third stage (trigger3). Upon activation of trigger3, it prints a message "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;
  }
```

Confirmation of a Fall:

To confirm a fall, trigger3 needs to remain active for a certain number of iterations (typically 10 in this code). This additional time duration helps filter out false alarms caused by transient changes in orientation. When trigger 3 is confirmed for the required duration, a fall detection event is generated. The program prints "FALL DETECTED," and the send_event() function is called to send this event to IFTTT. The program then resets the flags for trigger stages and their counts to prepare for potential future fall detections.

The three trigger stages work together to gradually confirm a fall event by monitoring both the initial changes in acceleration and subsequent changes in orientation. Each stage helps reduce false alarms, ensuring a more reliable fall detection system. Adjusting the thresholds and time

durations allows fine-tuning the sensitivity and specificity of the fall detection algorithm based on specific use-case requirements.

IFTTT event activation:

IFTTT (If This Then That) is a web-based service that allows you to create custom automation and integration between various apps, services, and devices. It works based on the principle of creating applets, where you define a trigger ("If This") and an action ("Then That").Once the trigger 3 is activated , A fall detection event is confirmed, the program sends an event to IFTTT using a predefined webhook URL and IFTTT Maker Channel key (privateKey). This event can be used to trigger notifications, alerts, or other actions.

IFTTT Webhooks is a service that provides a way to send and receive HTTP requests to trigger custom actions. It acts as a bridge between external services or devices and IFTTT. Using Webhooks an applet created with a Webhooks trigger as the "If This" part of the applet. You define a custom event name that will be used to trigger the applet. The Webhooks service is used to make an HTTP POST request to the IFTTT Webhooks URL, passing the event name and any required data as parameters in the request. When the Webhooks trigger event occurs (i.e., when the HTTP request is received by IFTTT), it activates the associated action, which is typically an action in another service or device. For "Then That" a rich notification is used that gives an emergency alert and with details related to the vehicle and also gives data related to tracking the rider’s mobile phone. IFTTT Webhooks enable custom web-based triggers, while notifications on IFTTT provide actions related to sending notification to family members related to tracking the location of the family member on their devices during an emergency situation. Combining these services allows you to create a wide range of automations and integrations to suit your needs, from IoT device notifications to personalized SMS alerts.