IoT Fall Detection – prototype evaluation

Assessment 3 CC5903 – Jarman Giffard

The design problem

Each year in Australia, roughly 5,300 people die from falls while 231,000 people are hospitalised (Injury in Australia: Falls, 2021) . The most common complications and rise in fatality rates resulting from falls is due to fall victims not being immediately discovered and who are unable to call for help. Consequently, a device that is able to automatically identify when a person has fallen and send an alert to a caregiver/emergency contact

The prototype device created for this exercise is designed to be worn on the body and identify if the wearer falls, at which point it will send an alert via email to a caregiver or emergency contact. Devices that are similarly purposed are available on the market, however are known to suffer from a high false-positive rate resulting in customer dissatisfaction and a reluctance to wear them (Fall detection in older adults with mobile IoT devices and machine learning in the cloud and on the edge, 2020) . In light of this, it is of importance that the sensors are functional and correctly detect changes in the wearer's movement/position – but also that the ensuing code logic that decides whether sudden changes are actually a fall or simply sudden, natural movements, is of sufficient complexity to maintain accuracy and specificity.

Sensors selected to solve the problem

The primary sensors selected for this device are the MPU-6050 dual gyroscope and accelerometer, and Pololu 38 khz infrared proximity sensor. Most commercially available fall-sensing devices use both an accelerometer and gyroscope to identify movement and relative position of the wearer, as they are effective at identifying the attributes of a fall: sudden movement followed by a change in position from vertical to horizontal. For this purpose, the selected MPU-6050 is a perfectly suitable choice from a design point of view – being cost effective, highly accurate, low power requirements, and being able to track 3 axes across both acceleration *and* relative position.

The Pololu infrared proximity sensor is a dual infrared emitter and receiver; designed to emit infrared waves out and then detect bounces of these waves which is used to estimate the distance from the sensor to the nearest object (within 60 cms) (Pololu 38 kHz IR Infrared Proximity Sensor with 60cm Range, n.d.). For this device and design problem, a proximity sensor offers near to zero benefit – it is unsuitable to measure the distance from the wearer to the floor as false-positive readings could be picked up from limbs, clothing, walls etc. From a pragmatic design view, an infrared sensor designed to measure heart rate is a suitable addition for this device, as it has been established an irregular heart rate is a reliable indicator of fainting, falls, strokes, and heart attacks (Heart beat rate monitoring using optical sensors, 2018).

Although the Pololu proximity sensor cannot be easily reconfigured to instead measure heart-rate instead of distance, it has been included in this prototype representing the nearest-available substitute for an IR heart rate sensor. Readings from this IR proximity sensor are captured in the code logic and presented to the end-user in the instance of a fall, demonstrating a working-state and how an actual IR heart rate sensor could be utilised.

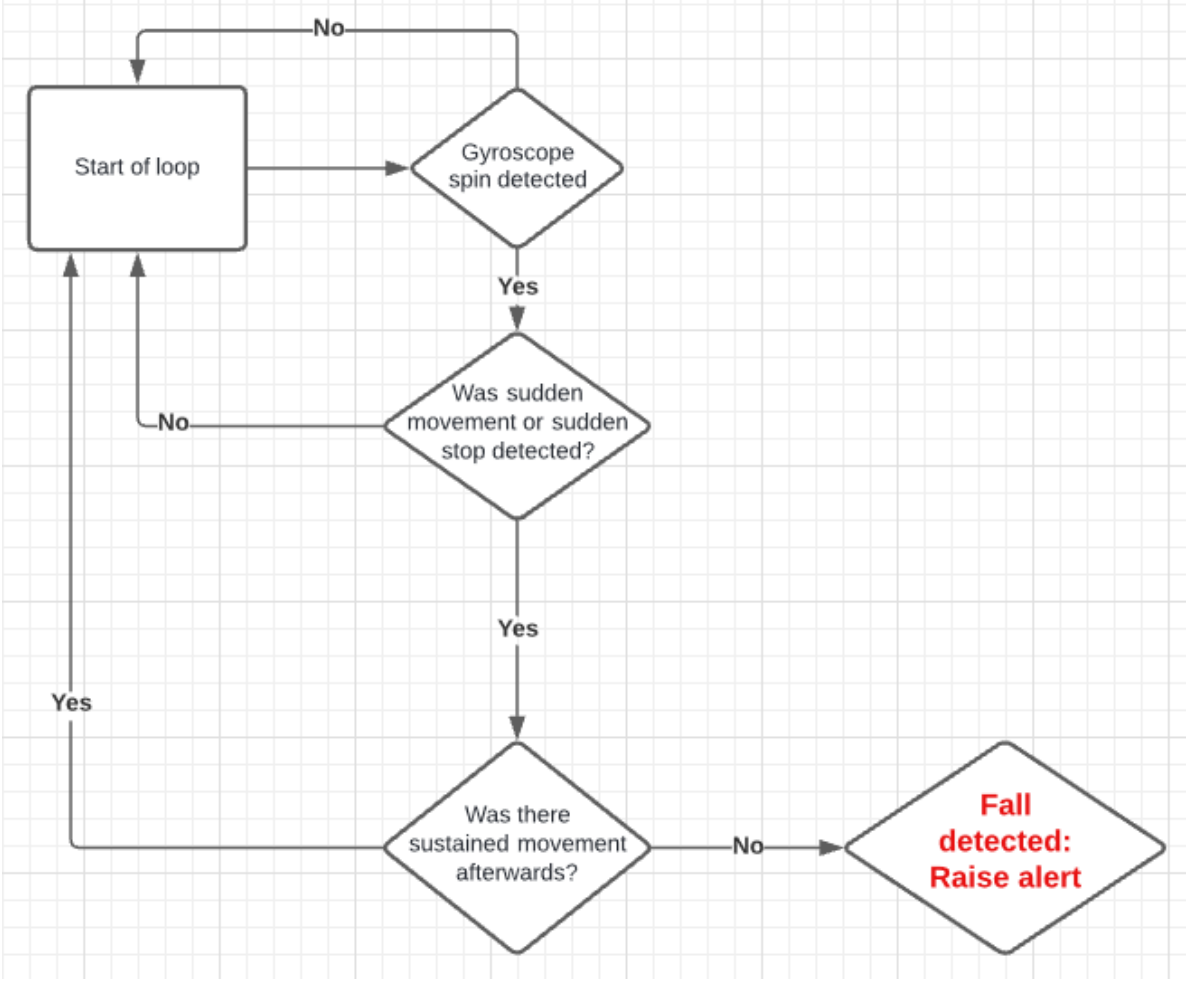
HOW DATA IS ACQUIRED

Once generated from the sensors and relayed to the breadboard, the data is then then processed on the Arduino board using C++ code. The logic for this program is lengthy and at times somewhat complex, but has been surmised in succinct detail below:

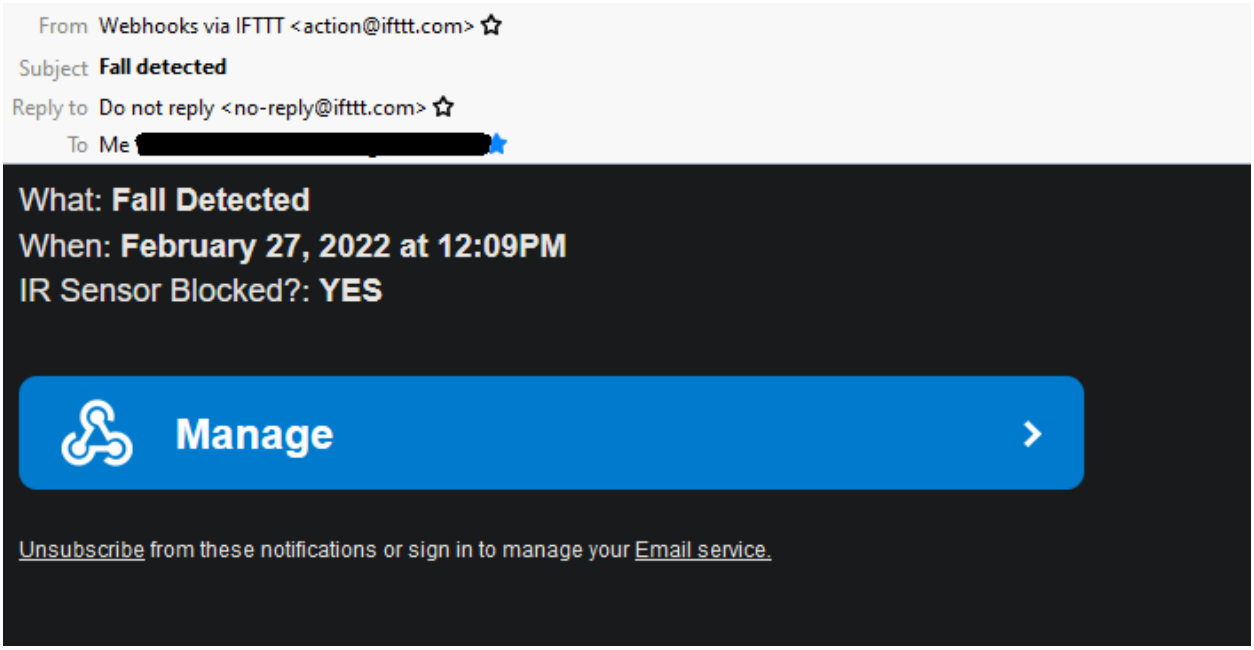
- Wire and ESP8266WiFi libraries are imported; to more easily manipulate and access data submitted across ports on the Arduino board (Arduino - Wire, n.d.) , and to access Wifi protocols
- Variables are initialized and their format defined (Boolean, integer, character etc). These variables are mostly orientated towards variables created from the gyroscope and accelerometer readings, and subsequently-derived variables used to identify potential fall 'triggers' (discussed further below).
    - There are additional global variables declared here, including the Wifi credentials and the If-This-Then-That URL and associated privatekey used to access the API for this specific applet created on their website
- The **void setup()** code chunk assigns the baud rate to 9600, wakes up the MPU-6050 sensor by making a registry value change, and connects to the wifi using credentials initialized in the global variables section described above
- The **void loop()** code chunk contains the majority of the code logic and starts off by extracting the complete compound value from the MPU-6050 sensor which contains the embedded values for the X, Y, and Z axis for the accelerometer and gyroscope.
- These 6 values (3 axes * 2 sources) are then extracted in order of precedence using a bitwise shift from the initial compound value and are then cleaned by dividing the accelerometer values by 16,384 and the gyroscope values by 131 – representing the sensitivity scale factor (RudraNarayanG, 2021)
- Next, the variable p2p (point-to-point movement) is created, measuring the relative change in distance from readings taken from the accelerometer across all three axes. This is used to measure overall change in speed and position further on
- The next steps consist of detecting specific events that are likely to be reliable criteria that indicate a fall has occurred:
    - **Sudden stop**: Detecting if the device has suddenly stopped by examining the point-to-point distance travelled and checking if there is little change
    - **Sudden move**: Detecting if the device has suddenly and rapidly moved – by counting the duration between loops and the relative distance travelled during a 3 second period using the aforementioned p2p variable

    - **Gyro spin**: Detecting if the total rotational spin across a 3 second period is above 15 and below 270 degrees
        - In the event that **sudden stop** is detected, followed by **gyro spin** – without any subsequent movement – the outcome is **fall detected**
        - in the event that a sudden move is detected, followed by gyro spin - without any subsequent movement – the outcome is **fall detected**
- in the event of a fall detected, the code will
    - connect to IFTTT.com, concatenating the public URL with the name of the key word defined during the applet creation ("FALL_DETECTED") – as well as the private API key provided by IFTTT during the account setup
    - state whether the IR proximity sensor is blocked
        - recall that this IR sensor represents an imperfect substitution for an IR pulse sensor which would be used to extend the alert-trigger logic
    - try to submit this string is a connection key until success
    - once succeeded, will terminate the connection and stop the wifi client
    - and finally, reset the trigger counters used in the preceding logic so the code doesn't get stuck in a loop

A visual representation of the critical logic gates used in identifying a fall is displayed below:



- once IFTTT receives the webhook request, an email is issued confirming a fall was detected along with the date, time, and whether the IR sensor was blocked.



HOW THE USER INTERACTS

This device is primarily catered to elderly people as they are significantly more likely to be involved in a fall and also suffer from injury or death following a fall; representing 94% of deaths recorded in accidental-falls (Injury in Australia: Falls, 2021).Subsequently, it is critical that the design of a product engineered for this demographic is simple to use and requires as little configuration as possible.

Elders' acceptance poses a major problem since they may not be familiar with electronic devices.
To overcome this challenge, the way the system operates is essential.
The detector should activate and operate automatically, without user intervention

(Igual, 2013)

In support of this the device requires no manual interaction in order to trigger the alert – the alert is raised automatically based on the predefined ruleset defined in the code.

The delay between:

- sensor readings qualifying as a 'detected fall' by the ensuing code,
    - establishing a connection to the wifi client
        - the submission to the IFTTT servers via API
            - IFTTT complying with the applet rules and sending an email,
                - finally followed by the email being received by the user

is very short – coming in at roughly 5 seconds from start to finish, indicating that serves as a wholly viable method of an early-warning detection system.

EVALUATION OF DESIGN

From a commercial minimum-viable-product (MVP) perspective there are several glaring issues that would require resolution before this device could be considered for market. These include the large weight and size of the device – coming in at 450 grams and 18cms in length – makes this unwieldy and cumbersome to transport in its current prototype state.

Secondly, as a result of substituting an infrared heart rate sensor with an infrared proximity sensor, important functionality that is useful to the success of identifying a fall is missing from the prototype in its current state. Future revisions of this prototype should implement an actual IR sensor for measuring heart rate, with appropriate code logic leveraging this data to properly identify sudden changes which may be indicative of a medical emergency. A suitable sensor that fulfills this function is the *DFRobot Heart Rate Monitor Sensor* (Gravity: Heart Rate Monitor Sensor, n.d.) which allows the output to be read as both an ECG and analogue (offering greater depth of recorded observations), while also offering "…an AD8232 chip on the PCB which will provide a clear signal from the PR and QT Intervals".

The method of communication is reliant on Wifi, meaning that this device is only suitable for use at home ~46 meters unobstructed (How far will your Wifi signal reach?, 2018). This can be remedied via use of a Bluetooth communication protocol, by pairing the device to the wearers smartphone and triggering alerts online through the smartphone's internet connection.

Power requirements are very small overall, with the current battery pack being more than enough to keep it running for at least 2 weeks. Any future MVP designs will naturally need to account for a battery source that is smaller in size and capacity, and to also allow easy recharging.

Due to the lack of a screen, a smartphone app allowing the user to interact with the device and reduce/increase sensitivity, assign the contact details for an emergency contact(s), and being able to view basic stats from the device's readings would be a very useful addition. Being able to nominate multiple contact points in the event of an alert being raised, as well as nominating different types of alerts – phone call (IVR), email, SMS, push alert etc, provides greater flexibility to the user to leverage their most appropriate contact channels.

Ultimately, although there are many opportunities for improvements before this device could see market combability – it does deliver on the functional requirements. It is truly wireless – without the need to be connected to a mains power supply and uses wireless communication. It correctly identifies falls – the code logic being sufficiently clever to have strategies to avoid false positives while also being sensitive enough to trigger when required. And it responds to falls appropriately – with an automatic email being triggered within seconds of the fall being detected.

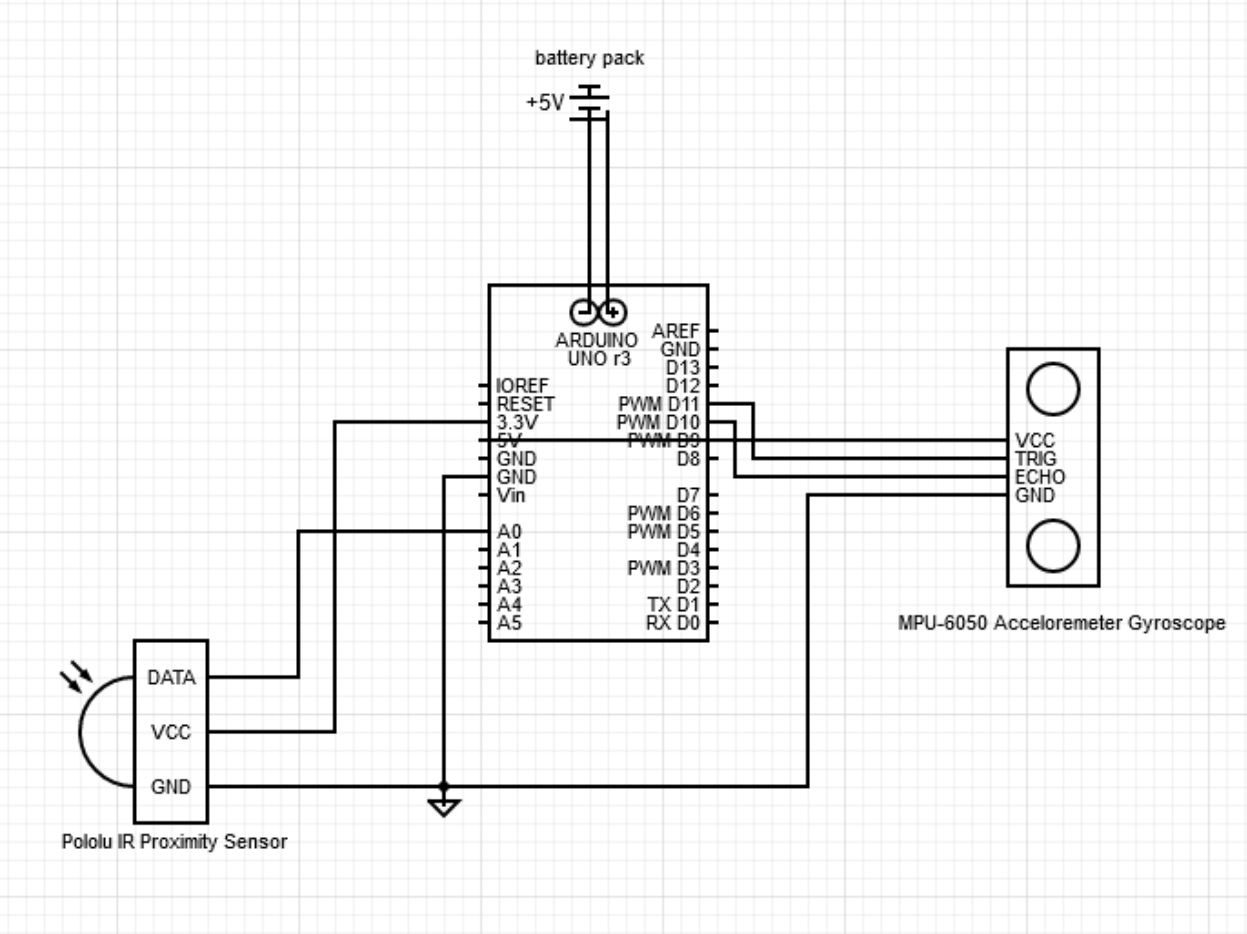
FUTURE RECOMMENDATIONS

In its current state, the user is unaware of an alert being triggered due to a detected fall. Under ideal operating conditions this is not a problem – but in the event the alert is a false positive and triggered incorrectly, it would be highly beneficial to warn the user beforehand. Implementation of this addition would likely take the form of a series of rapid beeps emitted from a speaker on the device, warning the user the device has detected a fall and allowing the user to press a button on the side to cancel the alert. This cancellation should be recorded and submitted – along with the previous ~20 seconds of sensor readings – back to the development team for examination, so that later updates/enhanced machine learning models can leverage the information to reduce the incident rate of false positives in the future.

A chassis to hold the device should be small and discrete and have contact against an artery to detect blood flow changes. A bracelet or necklace is capable of meeting these criteria and is also likely to be welcomed by elderly demographics which are the target audience for this device being more vulnerable.

APPENDIX

Appendix 1 – Circuit Schematic



https://www.circuit-diagram.org/editor/

Appendix 2 – Fall Detection Code

```
// import Wire and WiFi package for easier manipulation of registry values
and wifi config, respectively
#include <Wire.h>
#include <ESP8266WiFi.h>

/* initialise global variables */
/* -------------------------- */

// initialising the infrared readings, assigning as int as measurements vary
from ~20 - 1000
```

```
int ir_reading = 0;

// I2C address of the MPU-6050
const int MPU=0x68;

// X + Y + Z axes for accelerometer and gyroscope
int16_t acc_x,acc_y,acc_z,temp,gyro_x,gyro_y,gyro_z;

// 'cleaned' variables of the above
float a_x=0, a_y=0, a_z=0, g_x=0, g_y=0, g_z=0;

// localised chunks to identify a potential fall
bool fall_detected = false;
bool trig_low_thresh=false;
bool trig_high_thresh=false;
bool trig_orientation=false;

// number of ticks after respective trigger
byte trig_low_thresh_count=0;
byte trig_high_thresh_count=0;
byte trig_orientation_count=0;
int gyro_rotation=0;

// wifi details (pls don't dox me)
const char *ssid =  "Yodafone4G";
const char *pass =  "waterycoconut778";

// if-this-then-that (IFTTT) is the website hosting the alert capability
// the connection is created over wifi via API which requires a private key,
assigned below
void send_event(const char *event);
const char *host = "maker.ifttt.com";
const char *privateKey = "v7ukvAXTERqbeJ26GfGgKlMaOx1KvJUsHFi5MDKo2c";


void setup(){

  // assigning input from A0 pin to measure infrared proximity recordings
  /* ---------------------------------------- */
  pinMode(A0,INPUT);

  // set baud rate, wake up MPU
  /* ---------------------------------------- */
  Serial.begin(9600);
  Wire.begin();                      // Initialize comunication
  Wire.beginTransmission(MPU);       // Start communication with MPU6050 //
MPU=0x68
  Wire.write(0x6B);                  // Talk to the register 6B
  Wire.write(0x00);                  // Make reset - place a 0 into the 6B
register
  Wire.endTransmission(true);        //end the transmission

  // initial connection to wifi
  /* ---------------------------------------- */
  WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
      {delay(500);
```

```
      Serial.print(".");
      }
    Serial.println("");
    Serial.println("WiFi connected");
 }



void loop(){

/* extract core value from MPU registry*/
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);

//taking IR reading from A0 input pin
  ir_reading =  analogRead(A0);

/*bit wise movements to resolve and assign correct values from 'whole' value
above */
  Wire.requestFrom(MPU,14,true);    // request a total of 14 registers
    acc_x  = Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C
(ACCEL_XOUT_L)
    acc_y  = Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E
(ACCEL_YOUT_L)
    acc_z  = Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40
(ACCEL_ZOUT_L)
    temp   = Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H)   & 0x42
(TEMP_OUT_L)
    gyro_x = Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H)  & 0x44
(GYRO_XOUT_L)
    gyro_y = Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H)  & 0x46
(GYRO_YOUT_L)
    gyro_z = Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H)  & 0x48
(GYRO_ZOUT_L)

// clean accelerometer/gyroscope readings
   a_x = (acc_x-2050)/16384.00;
   a_y = (acc_y-77)/16384.00;
   a_z = (acc_z-1947)/16384.00;
   g_x = (gyro_x+270)/131.07;
   g_y = (gyro_y-351)/131.07;
   g_z = (gyro_z+136)/131.07;

 /* accelerometer: distance in point-to-point (p2p) movement  */
 float p2p = pow(pow(a_x,2)+pow(a_y,2)+pow(a_z,2),0.5);

/* gyroscope: measuring rotational spin across all 3 axes */
 float gyro_rotation = pow(pow(g_x,2)+pow(g_y,2)+pow(g_z,2),0.5);

 /* now scale values so they are more distinguishable */
 int p2p_clean = p2p * 10;

/* for debugging
 Serial.print(a_x); Serial.print(" -- ");
 Serial.print(a_y); Serial.print(" -- ");
 Serial.print(a_z); Serial.print(" -- ");
```

```
 Serial.print(g_x); Serial.print(" -- ");
 Serial.print(g_y); Serial.print(" -- ");
 Serial.print(g_z); Serial.print(" -- ");
 Serial.println(p2p_clean);
*/

/* identify when accelerometer suddenly stops */
 Serial.println(p2p_clean);
 if (p2p_clean<=3 && trig_high_thresh==false){
   trig_low_thresh=true;
   Serial.println("SUDDEN STOP");  }


/* identify when accelerometer quickly moves */
 if (trig_low_thresh==true){
   trig_low_thresh_count++;
   if (p2p_clean>=6){
     trig_high_thresh=true;
     Serial.println("SUDDEN MOVE");
     trig_low_thresh=false;
   trig_low_thresh_count=0;
    }}


 /* identify if gyroscope records relative change in position (total sum
across three dimensions) of 90 degrees */
 if (trig_high_thresh==true){
   trig_high_thresh_count++;
   Serial.println(gyro_rotation);
   if (gyro_rotation>=15 && gyro_rotation<=270){ // 3 angles * 90 degrees
possible rotation = max 270
     trig_orientation=true;
   trig_high_thresh_count=0;
     Serial.println(gyro_rotation);
     Serial.println("GYRO SPIN");
}}


/* if (sudden stop or sudden move) AND rotation detected, send alert */
 if (((trig_high_thresh == true) || (trig_low_thresh == true))
  && (trig_orientation == true)){
   Serial.println("FALL DETECTED");

/* now connect to IFTTT with alert package */
/* sourced from: https://stackoverflow.com/questions/39707504/how-to-use-
esp8266wifi-library-get-request-to-obtain-info-from-a-website */
          Serial.print("Connecting to ");
          Serial.println(host);
          WiFiClient client;
          const int httpPort = 80;
          if (!client.connect(host, httpPort)) {
            Serial.println("Connection failed");
            return;
          }
          String url = "/trigger/";
          url += "FALL_DETECTED";
          url += "/with/key/";
```

```
          url += privateKey;
          Serial.print("Requesting URL: ");
          Serial.println(url);
          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 {
              delay(50);
            };
          }
          Serial.println();
          Serial.println("closing connection");
          client.stop();

/* reset counters so the alert doesn't get stuck in loop */
   fall_detected=false;
   trig_high_thresh=false;
   trig_low_thresh=false;
   trig_orientation=false;
   }

/* allow 3 seconds (6 loops of 0.5 seconds) for triggers to be deactivated if
false-positive */
 if (trig_high_thresh_count>=6){
   trig_high_thresh=false;
   trig_high_thresh_count=0;
   Serial.println("HIGH THRESH - DEACTIVATED");
   }
 if (trig_low_thresh_count>=6){
   trig_low_thresh=false;
   trig_low_thresh_count=0;
   Serial.println("LOW THRESH - DEACTIVATED");
   }
  delay(500);
}
```

Bibliography

Arduino - Wire. (n.d.). Retrieved from Arduino Creative Commons: https://www.arduino.cc/en/reference/wire

Fall detection in older adults with mobile IoT devices and machine learning in the cloud and on the edge. (2020, 10 01). Retrieved from sciencedirect: https://www.sciencedirect.com/science/article/pii/S0020025520304886

Gravity: Heart Rate Monitor Sensor. (n.d.). Retrieved from Pakronics: https://www.pakronics.com.au/products/gravity-heart-rate-monitor-sensor-dfsen0213

Heart beat rate monitoring using optical sensors. (2018, 03 13). Retrieved from International Journal of Biosensors and Biolelectronics: https://medcraveonline.com/IJBSBE/heart-beat-rate-monitoring-using-optical-sensors.html

How far will your Wifi signal reach? (2018, 10 07). Retrieved from OpenWeb: https://openweb.co.za/how-far-will-your-wifi-signal-reach/

Igual, R. (2013, 07 06). *Challenges, issues and trends in fall detection systems - BioMedical Engineering OnLine*. Retrieved from BioMed Central : https://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-12-66

Injury in Australia: Falls. (2021, 12 09). Retrieved from Australian Institute of Health and Welfare: https://www.aihw.gov.au/reports/injury/falls

Pololu 38 kHz IR Infrared Proximity Sensor with 60cm Range. (n.d.). Retrieved from BananaRobotics: https://www.bananarobotics.com/shop/Pololu-38-kHz-IR-Infrared-Proximity-Sensor-with-60cm-Range

RudraNarayanG. (2021, 06 04). *MPU 6050 Gyro,Accelerometer Communication With Arduino (Atmega328p)*. Retrieved from Instructables Circuits: https://www.instructables.com/Accelerometer-MPU-6050-Communication-With-AVR-MCU/