Measuring Involuntary Breathing Movements with Prolonged Breath Holding

Lillian Tucker

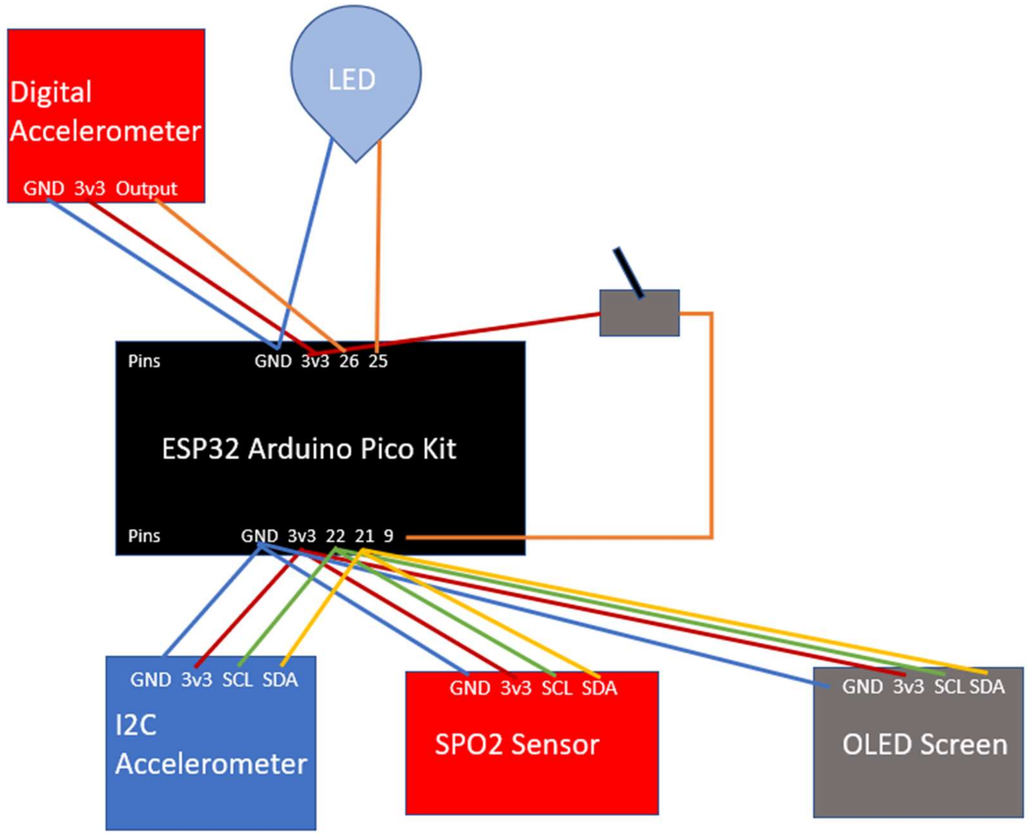
EECE 244

March 2023

Table of Contents

Overview

This paper is a report of an independently constructed device to measure IBM (involuntary breathing movements) with breath holding for extended periods of time. Provided electronics included a programmable circuit board and various electrical components as well as materials to manipulate these components. The goal was to create a more accurate reading for IBM.



Digital Accelerometer (ADXL 337)

The component used to measure the IBM is an ADXL337 digital accelerometer. This sensor is attached to the corresponding GND pin, the 3V3 pin of the ESP32 for power. For the input from the accelerometer to the Arduino, it is connected to pin 26 which is a general GPIO pin.

It is placed flat on the middle of the sternum to measure the IBM using the Z axis of the accelerometer. When analyzing IBM, majority of movements occurs in the chest which is the reason for placement. Another reason for this placement is the sternum area had minimal movement when adjusting the body for comfort.

The raw input of the accelerometer is the voltage. Using an example code provided by Dr. M, the function mapf() can be used to take this input and process it to output the acceleration of the accelerometer. This can then be printed to the serial monitor or serial plotter using Serial.println(). I scaled the acceleration data by a multiple of 100 to be able to clearly see changes in readings.

I2C Accelerometer (MPU 6050)

The component used to filter out movements other than IBM is an MPU 6050 I2C accelerometer. It is attached to the blank circuit board that the ESP32 Arduino is wired on. This sensor is attached to the corresponding GND and 3V3 pins of the ESP32 for power. Since this sensor is I2C, an SCL and SDA pin are required to operate. On the ESP32, pin 22 is used for SCL and pin 21 is used for SDA so the sensor is connected accordingly.

The smaller circuit board has the Arduino and I2C accelerometer connected directly and is placed flat on the hips round the hip bone area. On the waist is where most movement occurs for bodily adjustments for comfort and minimal movement during IBM. Like the ADXL337, the raw input from the MPU 6050 is voltage readings. Using a sample code downloaded from the MPU 6050 library, the voltage readings can be processed to acceleration. One unique component of the MPU 6050, is that in addition to being able to measure acceleration in the XYZ plane, it can also measure acceleration gyroscopically. This gyroscopic feature is used to measure hip movements in filtering out data. In the Arduino code, an if statement is used to determine when to plot the data from the digital accelerometer. If the gyroscopic data from the I2C accelerometer read less than 0.7, it would plot the data read by the ADXL337. Otherwise, no data would plot and there would be a 500-millisecond delay to allow time between the start of bodily adjustments and the start of measuring steady chest movements.

SPO2 Sensor

The component used to measure heartrate and blood oxygen levels is an I2C SPO2 sensor which is placed on the padding of the middle finger on the right hand for optimal reading. This sensor is attached to the corresponding GND and 3V3 pins of the ESP32 for power. Since this sensor is I2C, an SCL and SDA pin are required to operate. On the ESP32, pin 22 is used for SCL and pin 21 is used for SDA so the sensor is connected accordingly.

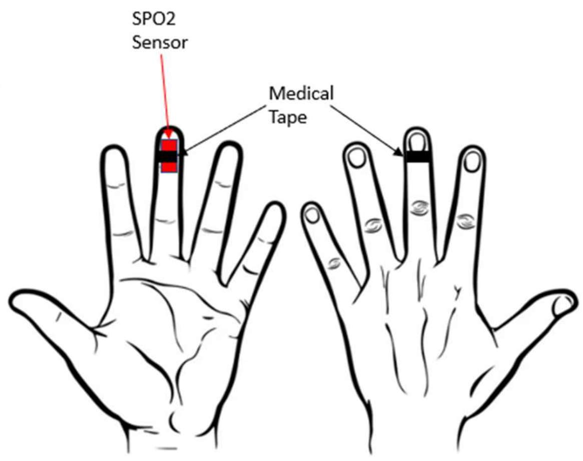
This component is connected directly on the larger circuit board with long wires extending the range so the sensor can be placed on the finger while the board is on the hip. There is an example library and code that can be downloaded and used to detect heart rate, blood oxygen levels, temperature, and reading status. There are 3 different reading statuses for the SPO2 monitor: 1 means there is no finger detected, 2 means there is a finger detected but isn't being properly detected, and 3 means there is a finger being properly detected to display heartrate and O2 levels. If the status reads 3, power to pin 25 will be turned on to light an LED to signal the SPO2 is properly detecting and measuring. Otherwise, the LED is turned off. The data is sampled at the same rate as both accelerometers.

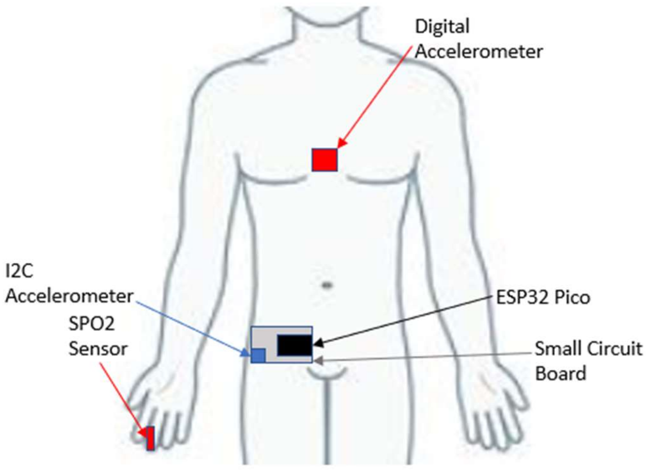
OLED Display Screen

The component used to display heart rate and blood oxygen levels as well as displaying when the system is measuring data. This sensor is attached to the corresponding GND and 3V3 pins of the ESP32 for power. Since this sensor is I2C, an SCL and SDA pin are required to operate. On the ESP32, pin 22 is used for SCL and pin 21 is used for SDA so the sensor is connected accordingly.

This component is connected directly on the larger circuit board next to the switch so the system can be monitored and turned on and off. There is a built-in library that can be downloaded for the OLED screen as well as an example code that can be used and manipulated to fit the needs of the screen and objective.

Set Up Instructions





<u>Small Circuit Board</u>

The smaller circuit board with the ESP32, digital accelerometer and I2C accelerometer should be placed on the right side of the hip over the hip bone. The I2C accelerometer should be oriented below the ESP32 in the direction of the feet of the subject while the ESP32 should be oriented above the I2C accelerometer in the direction of the head of the subject. Be sure the smaller circuit board is placed flat and secured against the hip.

<u>Digital Accelerometer</u>

The digital accelerometer, which is connect to the smaller circuit board, should be placed and taped on the sternum of the subject right above the xiphoid. Be sure the accelerometer is flat with the wires pointing towards the smaller circuit board.

<u>Large Circuit Board</u>

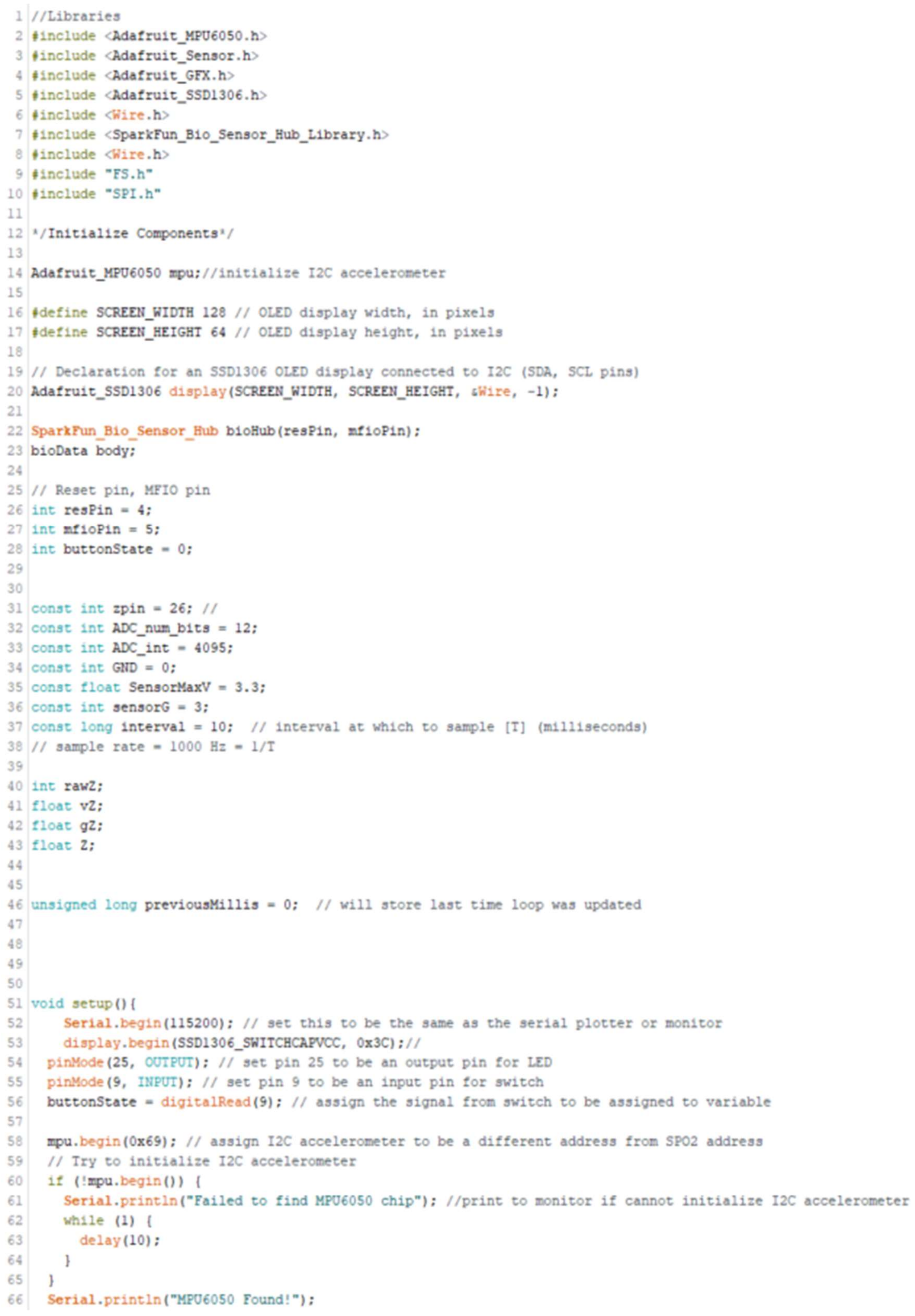
The larger circuit board with the OLED screen, SPO2 sensor, switch, and LED should be placed on the ground on the right side of the subject adjacent to the smaller circuit board. Ensure the larger circuit board is oriented so wires will remain connected.

<u>SPO2 Sensor</u>

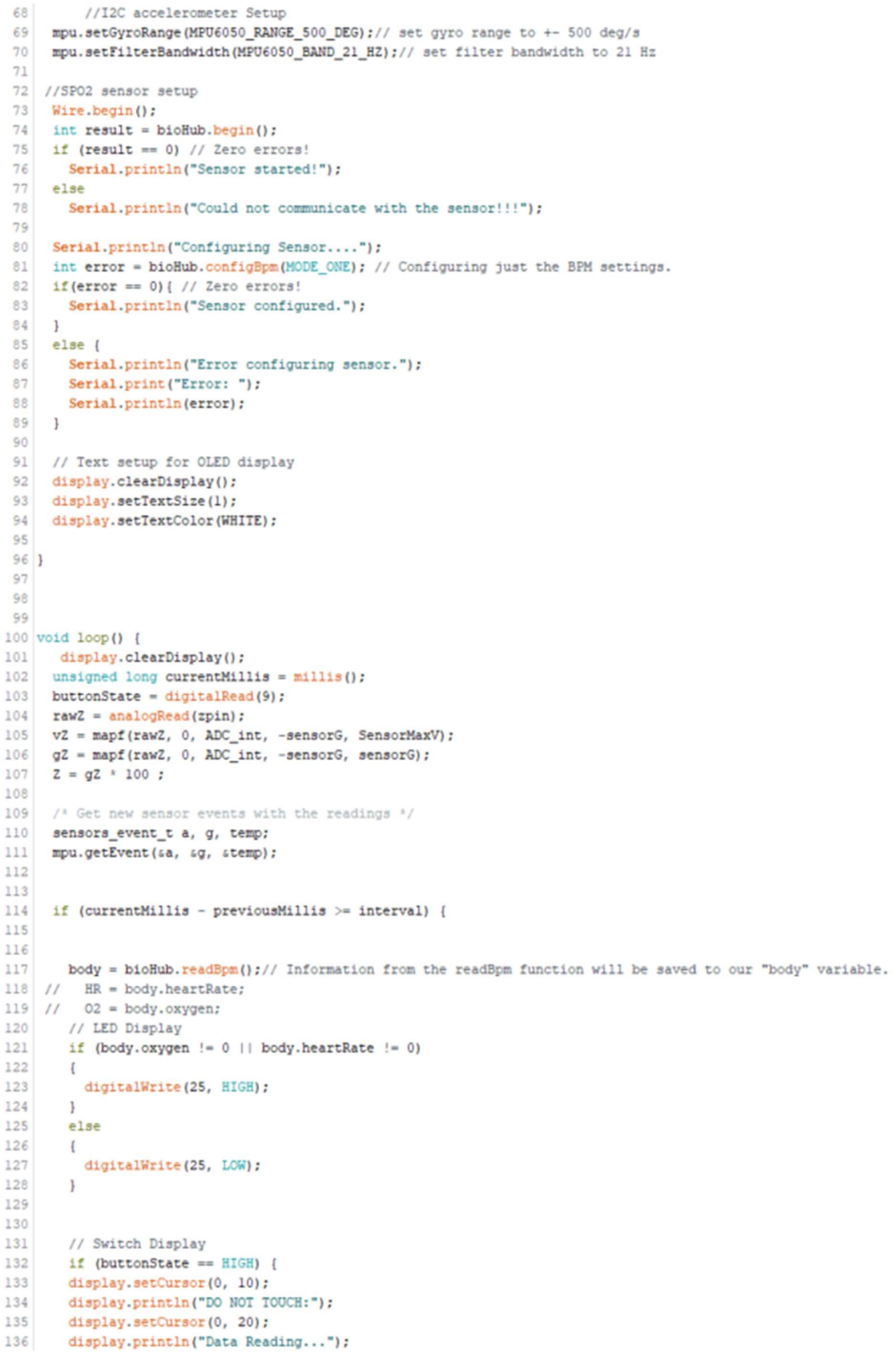
The SPO2 sensor should be placed and secured with medical tape on the right hand of the middle finger for optimal readings. The side that reads “Spark fun” should be against the padding of the finger. Ensure the SPO2 sensor is securely on the finger while keeping minimal pressure between the sensor and finger.

NOTE: Ensure the voltage pins are connected to voltage and ground pins are connected to ground. Displacing the two may result in irreversible damage to electrical components.

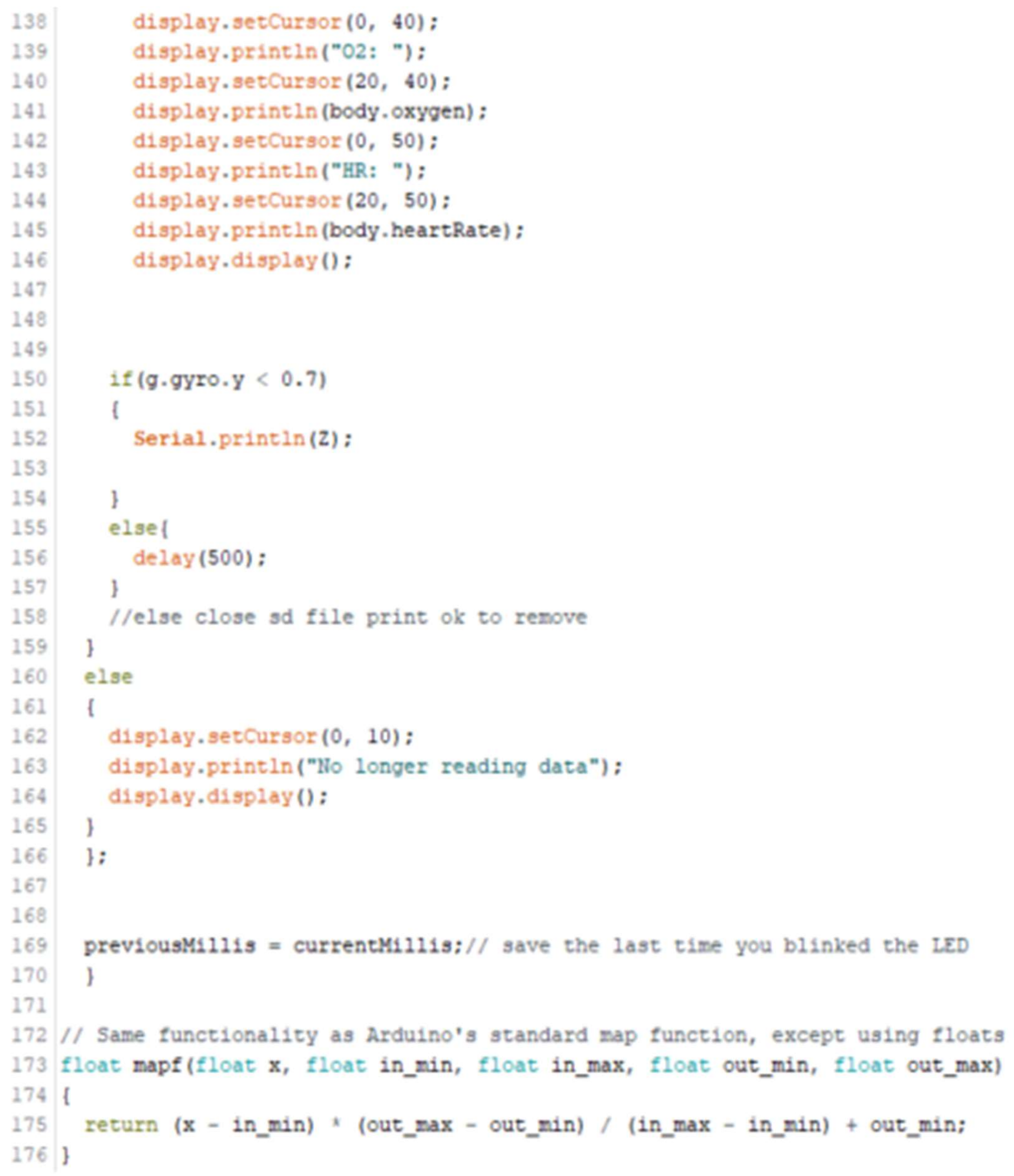
Code



```
//Libraries
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Wire.h>
#include <SparkFun_Bio_Sensor_Hub_Library.h>
#include <Wire.h>
#include "FS.h"
#include "SPI.h"

*/Initialize Components*/

Adafruit_MPU6050 mpu;//initialize I2C accelerometer

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 OLED display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

SparkFun_Bio_Sensor_Hub bioHub(resPin, mfioPin);
bioData body;

// Reset pin, MFIO pin
int resPin = 4;
int mfioPin = 5;
int buttonState = 0;


const int zpin = 26; //
const int ADC_num_bits = 12;
const int ADC_int = 4095;
const int GND = 0;
const float SensorMaxV = 3.3;
const int sensorG = 3;
const long interval = 10;  // interval at which to sample [T] (milliseconds)
// sample rate = 1000 Hz = 1/T

int rawZ;
float vZ;
float gZ;
float Z;


unsigned long previousMillis = 0;  // will store last time loop was updated




void setup(){
    Serial.begin(115200); // set this to be the same as the serial plotter or monitor
    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);//
  pinMode(25, OUTPUT); // set pin 25 to be an output pin for LED
  pinMode(9, INPUT); // set pin 9 to be an input pin for switch
  buttonState = digitalRead(9); // assign the signal from switch to be assigned to variable

  mpu.begin(0x69); // assign I2C accelerometer to be a different address from SPO2 address
  // Try to initialize I2C accelerometer
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip"); //print to monitor if cannot initialize I2C accelerometer
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");
```



```
      //I2C accelerometer Setup
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);// set gyro range to +- 500 deg/s
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);// set filter bandwidth to 21 Hz

 //SPO2 sensor setup
  Wire.begin();
  int result = bioHub.begin();
  if (result == 0) // Zero errors!
    Serial.println("Sensor started!");
  else
    Serial.println("Could not communicate with the sensor!!!");

  Serial.println("Configuring Sensor....");
  int error = bioHub.configBpm(MODE_ONE); // Configuring just the BPM settings.
  if(error == 0){ // Zero errors!
    Serial.println("Sensor configured.");
  }
  else {
    Serial.println("Error configuring sensor.");
    Serial.print("Error: ");
    Serial.println(error);
  }

  // Text setup for OLED display
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);

}



void loop() {
   display.clearDisplay();
  unsigned long currentMillis = millis();
  buttonState = digitalRead(9);
  rawZ = analogRead(zpin);
  vZ = mapf(rawZ, 0, ADC_int, -sensorG, SensorMaxV);
  gZ = mapf(rawZ, 0, ADC_int, -sensorG, sensorG);
  Z = gZ * 100 ;

  /* Get new sensor events with the readings */
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);


  if (currentMillis - previousMillis >= interval) {


    body = bioHub.readBpm();// Information from the readBpm function will be saved to our "body" variable.
 //   HR = body.heartRate;
 //   O2 = body.oxygen;
    // LED Display
    if (body.oxygen != 0 || body.heartRate != 0)
    {
      digitalWrite(25, HIGH);
    }
    else
    {
      digitalWrite(25, LOW);
    }


    // Switch Display
    if (buttonState == HIGH) {
    display.setCursor(0, 10);
    display.println("DO NOT TOUCH:");
    display.setCursor(0, 20);
    display.println("Data Reading...");
```



```
      display.setCursor(0, 40);
      display.println("O2: ");
      display.setCursor(20, 40);
      display.println(body.oxygen);
      display.setCursor(0, 50);
      display.println("HR: ");
      display.setCursor(20, 50);
      display.println(body.heartRate);
      display.display();



    if(g.gyro.y < 0.7)
    {
      Serial.println(Z);

    }
    else{
      delay(500);
    }
    //else close sd file print ok to remove
  }
  else
  {
    display.setCursor(0, 10);
    display.println("No longer reading data");
    display.display();
  }
  };


  previousMillis = currentMillis;// save the last time you blinked the LED
  }

// Same functionality as Arduino's standard map function, except using floats
float mapf(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Note: Requested sampling rate is 1000 Hz. 1000 Hz = 1/T. To achieve this, the loop function must run once every 10 milliseconds.

Results

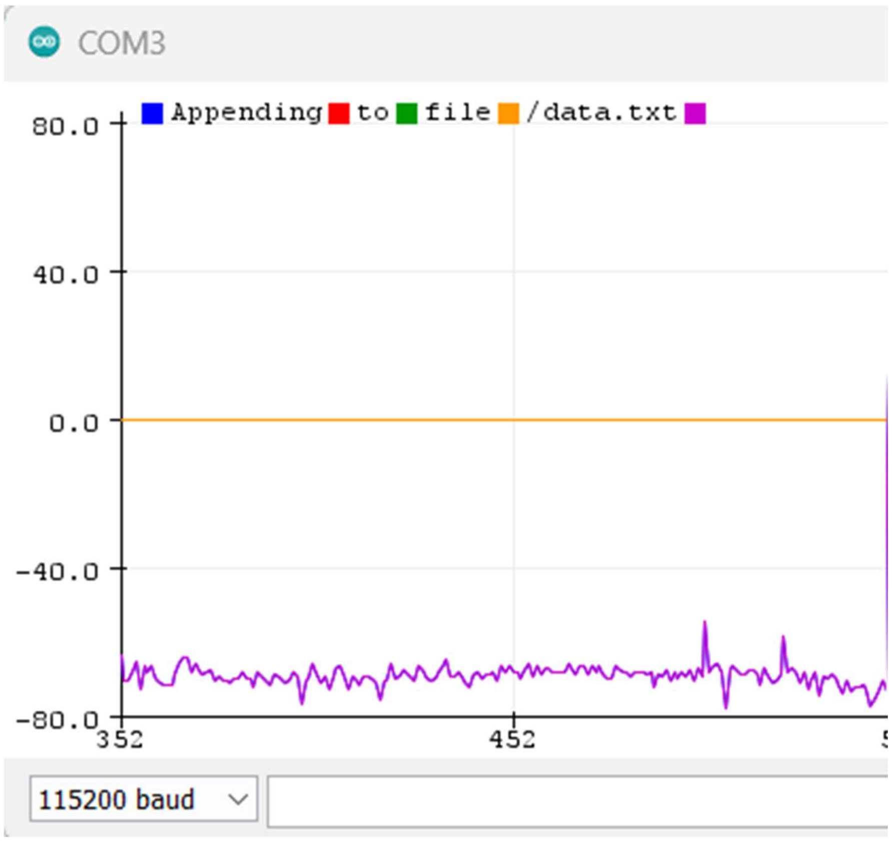


Figure 5: Serial Plot of Data from Arduino (Video includes trial for this data plot)

The numbers seen along the y axis of the plot was the acceleration of the digital accelerometer to a scale of 100, and the x axis is a fixed 500-point plot. This means the numbers on the bottom represent the number of cycles has been executed. Between 352 and 380 was a breath in, 380-410 was small manual chest movements, 410-440 was a breath out and in, 450-480 was a breath hold, 480-540 was large manual chest movements, and the last dip at the end was another breath out.

Although not shown on the plot, the plot did stop collecting data when there was noticeable movement measured on the I2C accelerometer. Although there were still some data differences while the chest movements were negligible, the difference in data shown while I performed chest movements and manual breathing was still notable.

With the accelerometer, the data grouping stays consistent rather than gradually increasing which can happen with force plates. This is because the instantaneous acceleration is independent to each data point rather than pressure plate sensors which can consistently gain pressure as the chest consistently expands as IBM increases.