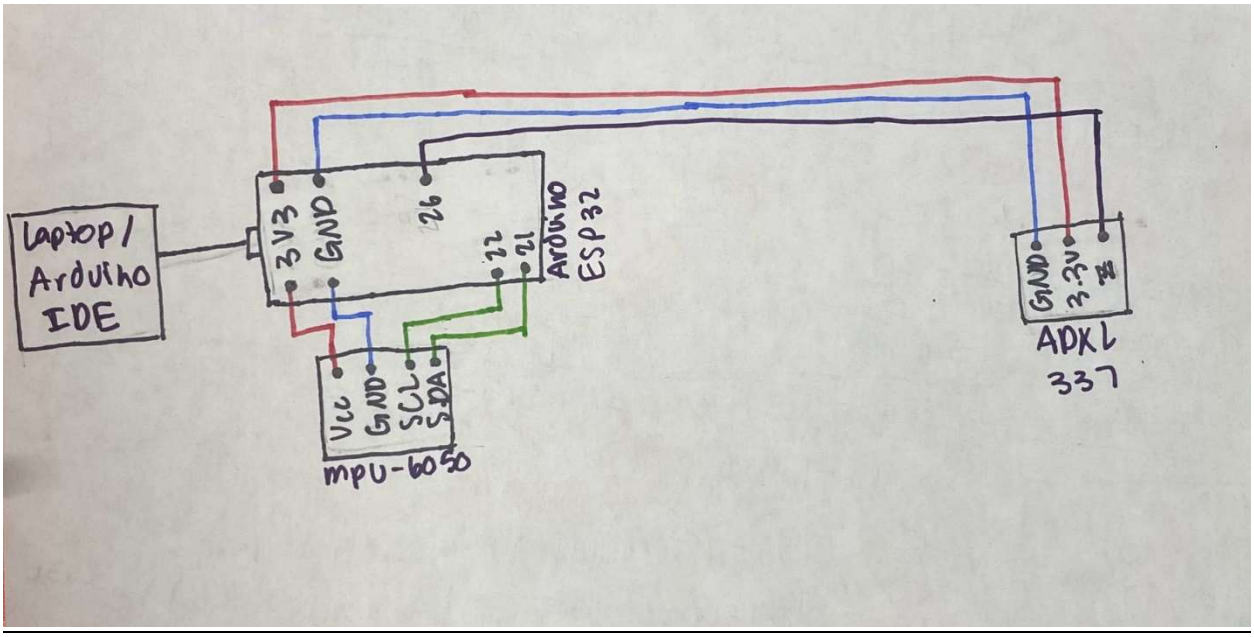
Measuring Involuntary Breathing Movements with Prolonged Breath Holding

Lillian Tucker

EECE 244

February 2023

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.



(Figure 1: Diagram of ESP32, ADXL337 and MPU6050)

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 watching the video of Jordan performing breath holding to demonstrate the IBM, I noticed that with any IBM, there was consistently movement in the sternum area. Another reason for this is the sternum area had minimal movement when Jordan was adjusting his 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. When watching the IBM demonstration of Jordan, there was consistent movement in the hip area when he adjusted his body 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. I used this to measure hip movements in filtering out data. In the Arduino code, I used an if statement 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 any finger. 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 also connected directly on the smaller circuit board with long wires extending the range so the sensor can be places 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.

MicroSD Card Adapter (HW-125)

The component used to write data to a file is the HW-125 MicroSD Card Adapter. The ESP32 has a library for writing to and SD card and that example was used here. This component is attached to the corresponding GND and 3V3 pins of the ESP32 for power. The microSD card Adapter uses SPI to communicate so there 4 pins in addition to VCC and GND: CS (chip select/slave select) connected to pin 5, SCK (serial clock) connected to pin 10, MOSI (master out slave in) connected to pin 23, and MISO (master in slave out) connected to pin 19.

This component is connected on a larger circuit board separate from the one the ESP32 is attached to. Wires connect the larger circuit board to the smaller circuit board with the ESP32. This is because I did not want to have the interactable components on the hip with the ESP32 as interacting with the components could make the subject uncomfortable.

There are pre coded conditions for if the SD card cannot be read, if the file cannot be read, if the file cannot be open/closed, and more. The code I added is to find/create the data file and append to the file in

the loop. The data written to this file is milliseconds, heartrate, oxygen levels, and acceleration of the digital accelerometer. If the button is turned on, SPO2 data is written to the SD card if there is a valid reading. If the I2C accelerometer is within its acceptable gyroscopic range, that data will also be written to the SD card. Once the button is turned off, data stops writing to the SD card. The sampling rate of writing data is the same as the sampling rate of capturing data.

The calibration of the if-else statement was find using trial and error. The gyroscopic threshold was originally set to < 0 but I found that it was too sensitive and would exclude more drastic chest movements. The delay in the else statement was originally set to 100-milliseconds. I found this delay to be too short as data would resume plotting while bodily adjustments were still active. I increased this delay by 100-milliseconds until I was satisfied by the delay.

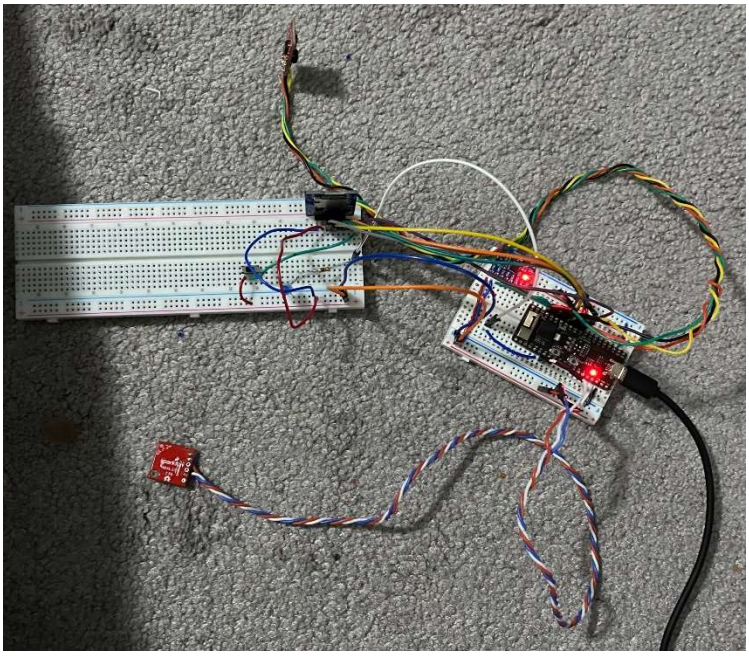


Figure 2: Real setup

Figure 3: Real setup on person

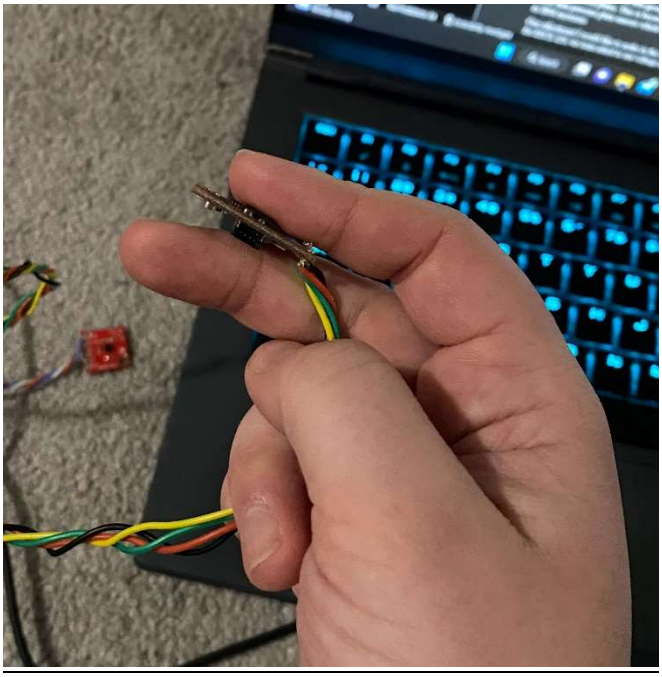


Figure 4: SPO2 sensor on finger

Results

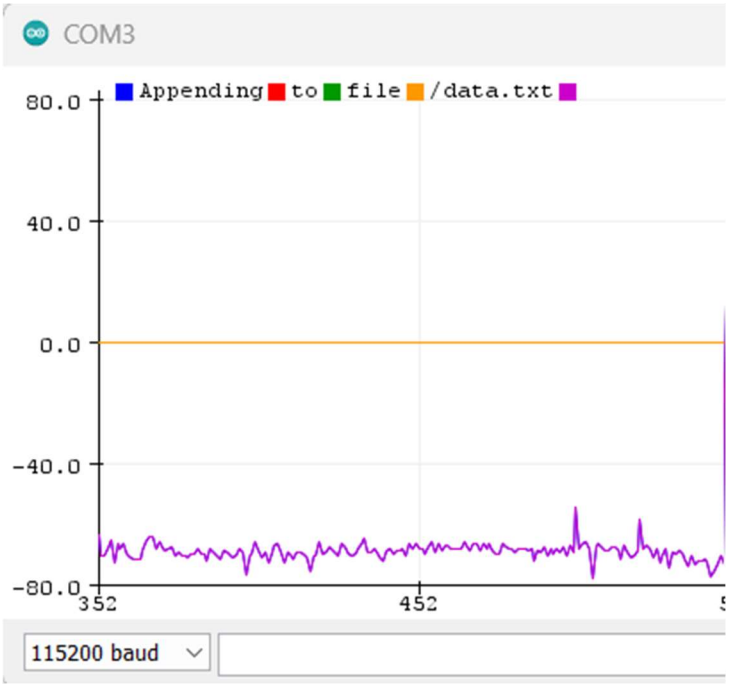


Figure 5: Serial Plot of Data from Arduino (Video includes trial for this data plot)

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.

One adjustment I would like to make in the future is filtering for external electrical components. In a lab for EECE 226, we were asked to see voltage and frequency interference various objects and electronics can produce. One device I decided to observe is my personal insulin which sits on the side of my hip around the hip bone area. When measuring voltage interference of my pumped, I measured 60mVpp interference while the pump was on and touching the oscilloscope. While the oscilloscope was approximately 2 inches away from the insulin pump there was a measured 20 mVpp while the pump was on. In the future I want to keep in mind the various medical devices individuals may have on them day to day.