# *NSEMBED Mini-Project 2*

Section: S12

Group No: 5

Members:

Azevedo, Marquus Joss

Del Castillo, Jose Mari

07/22/2024 – Version 0.2

## Change Control

| Version | Date | Comments |
|---|---|---|
| 0.1 | 07/20/2024 | Initial draft |
| 0.2 | 07/22/2024 | Final version |

## Table of Contents

## I. System Overview

**Vibration Data Collector System**

The created Vibration Data collector is made using an ESP32 microcontroller and an ADXL345 Accelerometer. An accelerometer can help determine how much vibration a certain mechanism can have, it could also determine the structural integrity if objects such as buildings, machines, or vehicles, and are also innovatively used in smart phones for screen rotation purposes or gyroscope features in video games and different smart phone applications. In this project, we will be testing some intentional and unintentional scenarios where we can measure vibrations that will be located inside a home.

A diagram of the system will be based on video in YouTube [1] as seen in Figure 1 adjusted with the size of the breadboard.

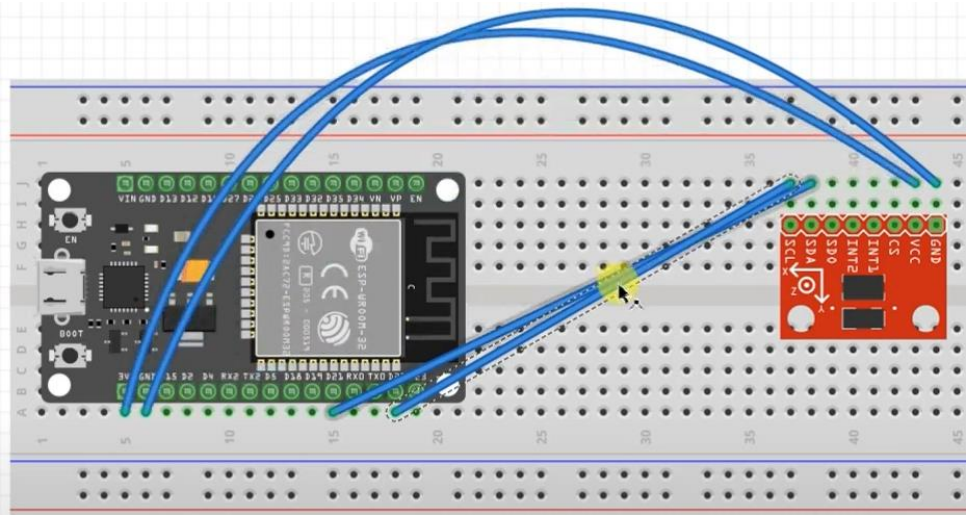


*Figure 1. ESP32 and ADXL345 [1]*

For the Arduino program of the system, in every 100 milliseconds, it would print out a JSON format output of the current time elapsed, X, Y and Z values (in G-force) obtained from the ADXL345 accelerometer. The X, Y and Z values will change depending upon the direction or rotation made on the accelerometer as seen in Figure 2.

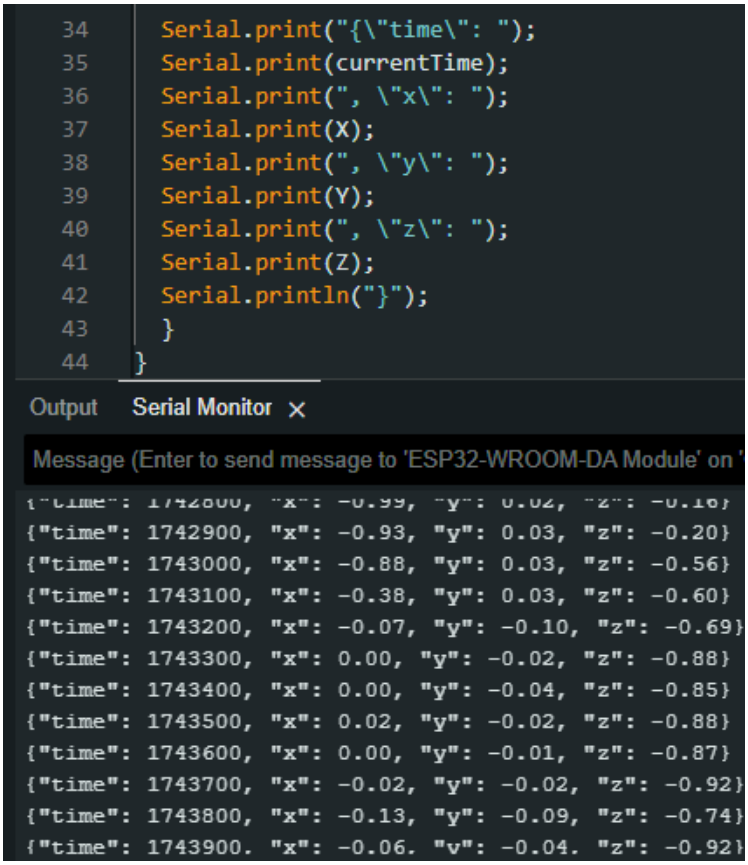


*Figure 2*

To encode the JSON format text to a CSV file, a Python program will be used in which it will also display the current vibration data logs like the serial monitor in the Arduino IDE as seen in Figure 3. Note that the time is normalized to start in 0 milliseconds.

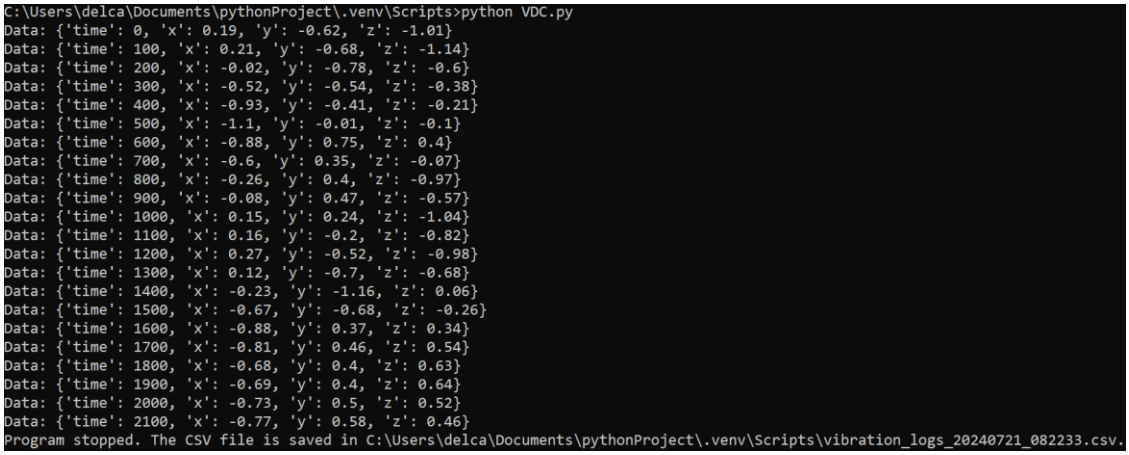


*Figure 3*

As seen in Figure 4, the values from the JSON format are encoded properly in the CSV file Python program output as seen in Figure 3.

| | time | x | y | z |
|---|---|---|---|---|
| 2 | 0 | 0.19 | -0.62 | -1.01 |
| 3 | 100 | 0.21 | -0.68 | -1.14 |
| 4 | 200 | -0.02 | -0.78 | -0.6 |
| 5 | 300 | -0.52 | -0.54 | -0.38 |
| 6 | 400 | -0.93 | -0.41 | -0.21 |
| 7 | 500 | -1.1 | -0.01 | -0.1 |
| 8 | 600 | -0.88 | 0.75 | 0.4 |
| 9 | 700 | -0.6 | 0.35 | -0.07 |
| 10 | 800 | -0.26 | 0.4 | -0.97 |
| 11 | 900 | -0.08 | 0.47 | -0.57 |
| 12 | 1000 | 0.15 | 0.24 | -1.04 |
| 13 | 1100 | 0.16 | -0.2 | -0.82 |
| 14 | 1200 | 0.27 | -0.52 | -0.98 |
| 15 | 1300 | 0.12 | -0.7 | -0.68 |
| 16 | 1400 | -0.23 | -1.16 | 0.06 |
| 17 | 1500 | -0.67 | -0.68 | -0.26 |
| 18 | 1600 | -0.88 | 0.37 | 0.34 |
| 19 | 1700 | -0.81 | 0.46 | 0.54 |
| 20 | 1800 | -0.68 | 0.4 | 0.63 |
| 21 | 1900 | -0.69 | 0.4 | 0.64 |
| 22 | 2000 | -0.73 | 0.5 | 0.52 |
| 23 | 2100 | -0.77 | 0.58 | 0.46 |

*Figure 4*

**Components List, Hardware Setup and Accelerometer Setup**

**The component list will be listed below:**

- ADXL345 Accelerometer
- ESP32-WROOM-32D (DevKitC_V4)
- 4 Male-to-Female Cables
- 1 Breadboard
- Data and Charging Cable

**Accelerometer, ESP32 and Breadboard Setup:**

For the breadboard setup, the ADXL345 accelerometer pins will be connected to the h2 to h9 grid sockets. The male end of the 4 cables will be connected to the GND (Ground – black cable), VCC (Voltage – red cable), SDA (Serial Data Line – yellow cable), and SCL (Serial Clock line – green cable) as seen in Figure 5.

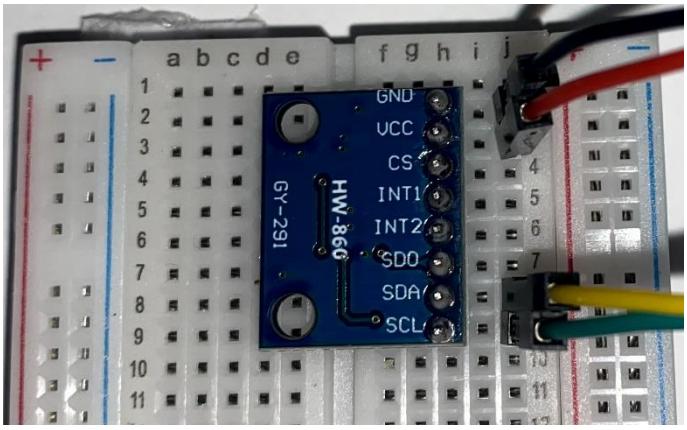


*Figure 5*

For the ESP32 setup, female end of the 4 cables will be connected to the 3V3 (3.3 volts – red cable) for the power required by the accelerometer (VCC), GND (Ground – black cable) for safety in the case of short circuits, pin 21 (yellow cable) for the serial data line to be able to communicate with the ESP32, and pin 22 (green cable) for the serial clock line for synchronization of both the ESP32 and the accelerometer as seen in Figure 6.

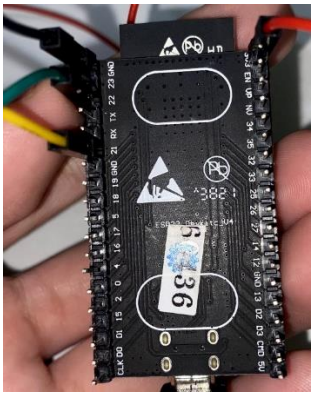

*Figure 6*

The final result of the setup is shown in Figure 7.

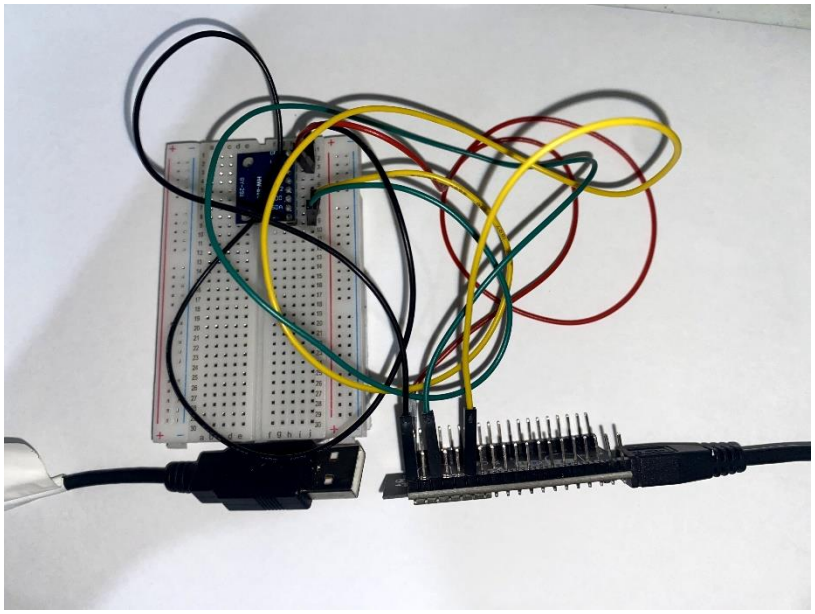

*Figure 7*

**Testing and Data Encoding in Python of Raw Data acquired from the Accelerometer**

The initial testing of the Vibration Data Collector was conducted on the table of one of the students and it was determined that the system is functioning properly as seen in Figure 8, Figure 9, Figure 10, and Figure 11 in which the time, X, Y, and Z values are encoded properly in the CSV file output of the Python program with the time being normalized to 0 at the start to make sure that the last timestamp output will be the total time elapsed in milliseconds.

A video of the short demonstration of the Vibration Data Collector will be seen in: Vibration Data Collector Initial Test.

The outputs for the Python program can be seen in Figure 10, and Figure 11. The CSV file output can be accessed in: vibration_logs_20240721_091811

*Figure 8*

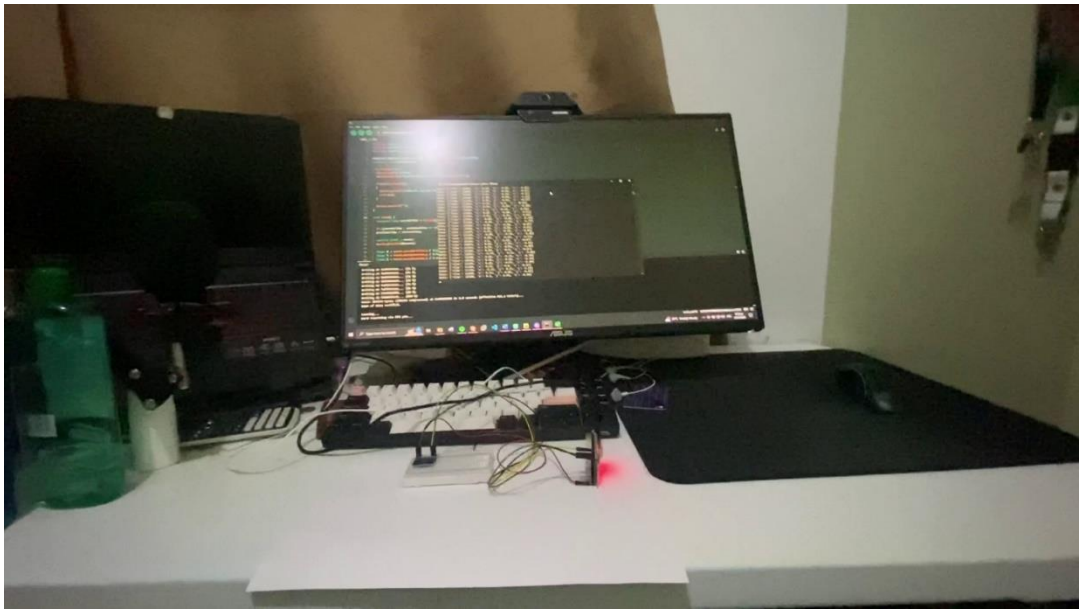

*Figure 9. Table is bumped intentionally*

```
Data: {'time': 4200, 'x': 0.08, 'y': 0.05, 'z': -0.87}
Data: {'time': 4300, 'x': 0.07, 'y': -0.03, 'z': -0.95}
Data: {'time': 4400, 'x': 0.07, 'y': -0.01, 'z': -0.86}
Data: {'time': 4500, 'x': 0.08, 'y': -0.01, 'z': -1.03}
Data: {'time': 4600, 'x': 0.14, 'y': -0.17, 'z': -1.13}
Data: {'time': 4700, 'x': -0.02, 'y': -0.05, 'z': -0.96}
Data: {'time': 4800, 'x': -0.02, 'y': -0.05, 'z': -0.84}
Data: {'time': 4900, 'x': 0.04, 'y': -0.17, 'z': -0.86}
Data: {'time': 5000, 'x': 0.06, 'y': -0.25, 'z': -0.74}
Data: {'time': 5100, 'x': 0.07, 'y': -0.05, 'z': -0.8}
Data: {'time': 5200, 'x': 0.02, 'y': -0.14, 'z': -0.78}
Data: {'time': 5300, 'x': 0.07, 'y': -0.04, 'z': -0.74}
Data: {'time': 5400, 'x': 0.04, 'y': -0.1, 'z': -1.0}
Data: {'time': 5500, 'x': 0.0, 'y': -0.12, 'z': -0.82}
Data: {'time': 5600, 'x': 0.04, 'y': -0.06, 'z': -0.91}
Data: {'time': 5700, 'x': 0.07, 'y': -0.1, 'z': -0.9}
Data: {'time': 5800, 'x': 0.04, 'y': -0.03, 'z': -0.92}
Data: {'time': 5900, 'x': 0.0, 'y': -0.03, 'z': -0.9}
Data: {'time': 6000, 'x': -0.02, 'y': -0.09, 'z': -0.89}
Data: {'time': 6100, 'x': -0.03, 'y': -0.08, 'z': -0.86}
Data: {'time': 6200, 'x': -0.02, 'y': -0.12, 'z': -0.87}
Data: {'time': 6300, 'x': -0.0, 'y': -0.03, 'z': -0.86}
Data: {'time': 6400, 'x': -0.04, 'y': 0.01, 'z': -0.85}
Data: {'time': 6500, 'x': -0.07, 'y': 0.2, 'z': -0.97}
Data: {'time': 6600, 'x': 0.11, 'y': 0.55, 'z': -0.85}
Data: {'time': 6700, 'x': 0.19, 'y': 0.91, 'z': -0.7}
Data: {'time': 6800, 'x': 0.15, 'y': 1.06, 'z': -0.55}
Data: {'time': 6900, 'x': 0.16, 'y': 1.12, 'z': -0.34}
Data: {'time': 7000, 'x': 0.02, 'y': 1.02, 'z': 0.1}
```

*Figure 10*

| | | | | |
|---|---|---|---|---|
| 44 | 4200 | 0.08 | 0.05 | -0.87 |
| 45 | 4300 | 0.07 | -0.03 | -0.95 |
| 46 | 4400 | 0.07 | -0.01 | -0.86 |
| 47 | 4500 | 0.08 | -0.01 | -1.03 |
| 48 | 4600 | 0.14 | -0.17 | -1.13 |
| 49 | 4700 | -0.02 | -0.05 | -0.96 |
| 50 | 4800 | -0.02 | -0.05 | -0.84 |
| 51 | 4900 | 0.04 | -0.17 | -0.86 |
| 52 | 5000 | 0.06 | -0.25 | -0.74 |
| 53 | 5100 | 0.07 | -0.05 | -0.8 |
| 54 | 5200 | 0.02 | -0.14 | -0.78 |
| 55 | 5300 | 0.07 | -0.04 | -0.74 |
| 56 | 5400 | 0.04 | -0.1 | -1 |
| 57 | 5500 | 0 | -0.12 | -0.82 |
| 58 | 5600 | 0.04 | -0.06 | -0.91 |
| 59 | 5700 | 0.07 | -0.1 | -0.9 |
| 60 | 5800 | 0.04 | -0.03 | -0.92 |
| 61 | 5900 | 0 | -0.03 | -0.9 |
| 62 | 6000 | -0.02 | -0.09 | -0.89 |
| 63 | 6100 | -0.03 | -0.08 | -0.86 |
| 64 | 6200 | -0.02 | -0.12 | -0.87 |
| 65 | 6300 | 0 | -0.03 | -0.86 |
| 66 | 6400 | -0.04 | 0.01 | -0.85 |
| 67 | 6500 | -0.07 | 0.2 | -0.97 |
| 68 | 6600 | 0.11 | 0.55 | -0.85 |
| 69 | 6700 | 0.19 | 0.91 | -0.7 |
| 70 | 6800 | 0.15 | 1.06 | -0.55 |
| 71 | 6900 | 0.16 | 1.12 | -0.34 |
| 72 | 7000 | 0.02 | 1.02 | 0.1 |

*Figure 11*

The final testing will be conducted in the same environment in a scenario where one of the students is playing a first-person shooter video game (Figure 12) to collect the vibration data on his table to test on how much it is wobbling when playing as a person can create drastic movements when aiming or shooting when controlling the spray pattern of a certain gun as seen in Figure 13.

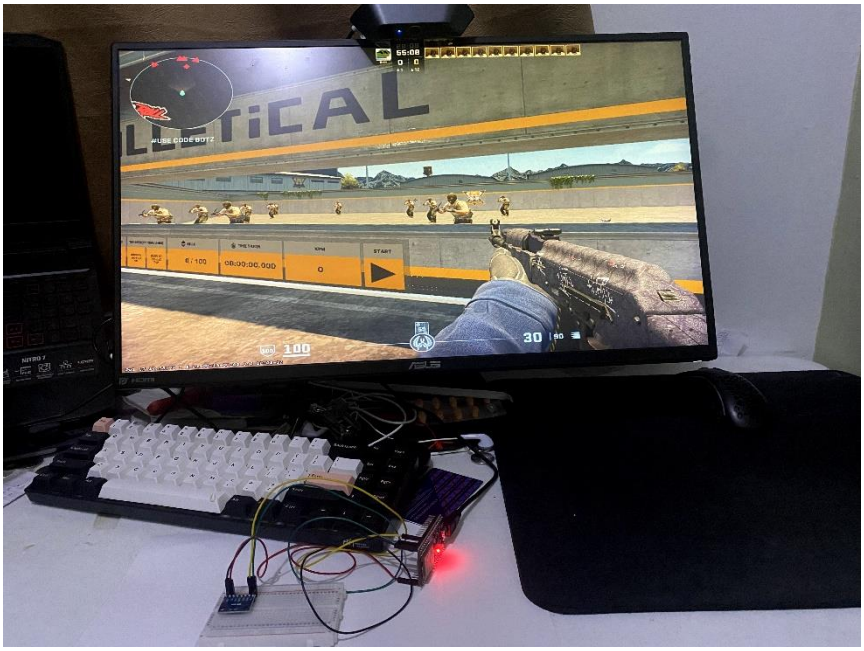

*Figure 12*

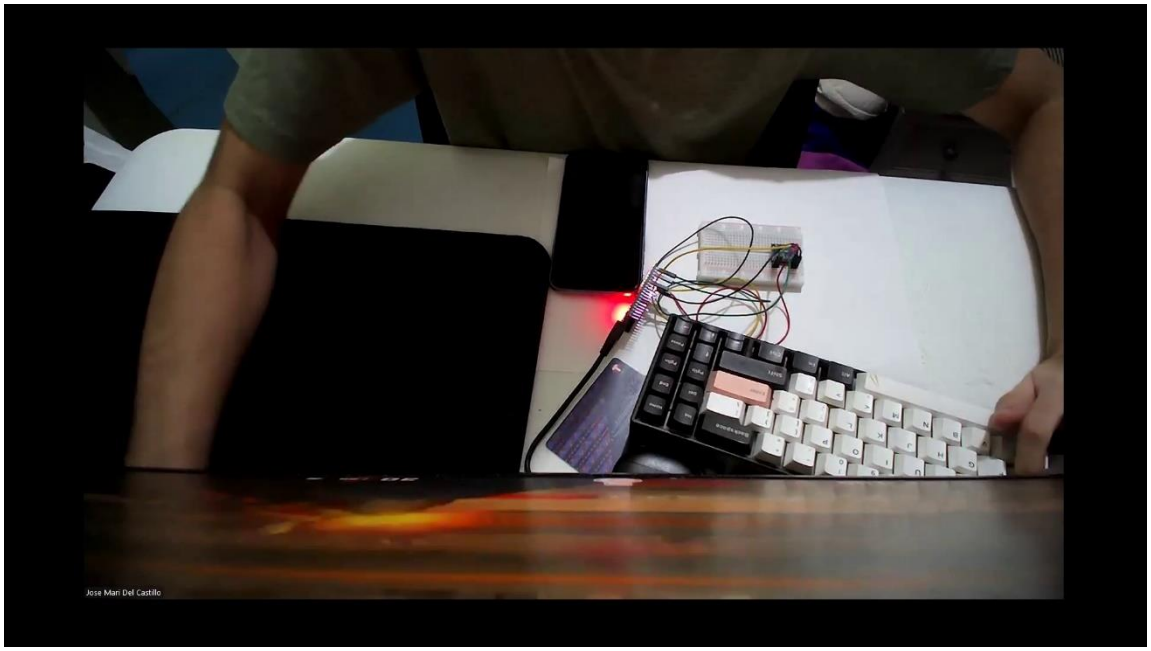


*Figure 13*

As seen in Figure 14, the Python program was able to record the data from the accelerometer for the whole test. The data collected is also the same in the CSV output.

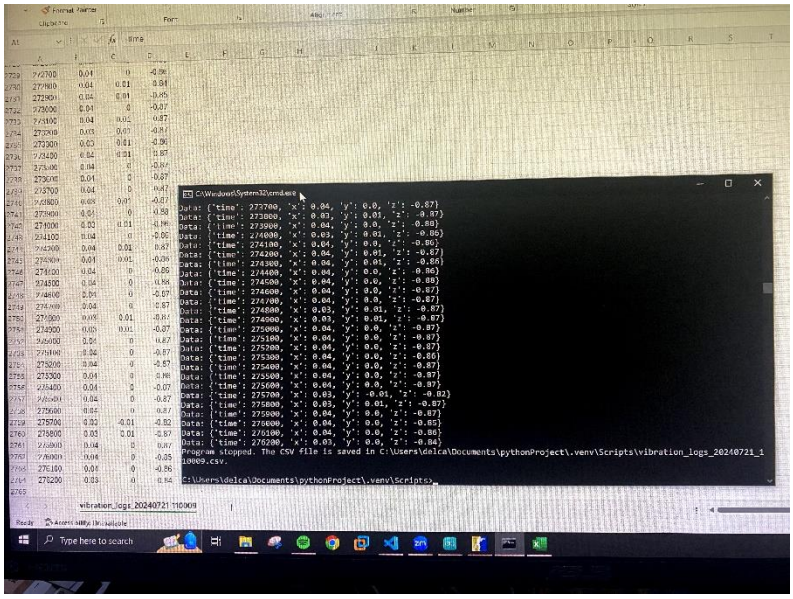

*Figure 14*

The video recording for the verification of the data gathered from the Vibration Data Collector and the CSV output for the test will be seen in: System Video Recording and Log.

## II. Sensor Discussion

### ADXL345

The sensor that will be used is an ADXL345 which will be acted upon as an Accelerometer. The Accelerometer or ADXL345 is a circuit that measures acceleration in 3 axes such as the x, y and z axes [2]. The sensor consists of micro-machined structure on a silicon wafer. The structure is suspended by polysilicon springs that allows smooth deflection in any direction when it is subjected to acceleration in X, Y and/or Z axis [3]. The acceleration force of the Accelerometer is in *g* units wherein it has the acceleration due to gravity equal to $9.8\ m/s^2$ [2]. For it to be used in Arduino, the library "Adafruit ADXL345" must be installed. Its features contain an ultralow power of 23 μA in measurement mode and 0.1 μA in standby mode at 2.5V, a supply voltage range of 2.0V to 3.6V, temperature ranges from −40°C to +85°C and can detect single/double tapping and free-fall [4].

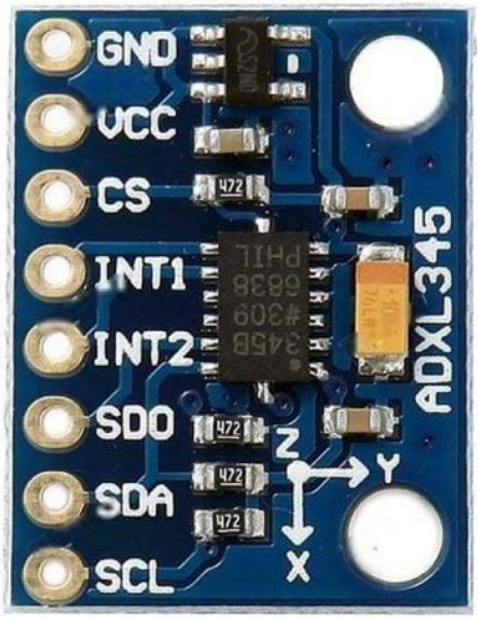


*Figure 15*

Figure 15 shows the pins that are present in the sensor are GND, VCC, CS, INT1, INT2, SDO, SDA, and SCL. GND is the ground pin. VCC is the power supply pin which has an input voltage from 3V to 6V. CS is the chip select pin, but it also determines whether the board will boot up into I2C or SPI mode. INT1 and INT2 pins are the interrupt 1 output pin and interrupt 2 output pin on the sensor. The SDO pin is the serial data output pin which can be used as MISO in SPI mode. The SDA pin is the serial data input and output pin which is the data line on the I2C bus. Lastly, the SCL pin is the serial communication clock pin which is the clock line on the I2C bus [3] [5] .

## III. Results and Analysis

For the testing of the Vibration Data Collector Python program's CSV output, the simplest way to determine if the wobble of the table is acceptable when the student is playing first-person shooter games is to determine if there are outliers or points that differ from most of the points per column in the dataset to depict if there are significant amount of wobbliness of the table when playing. In which, we will determine if there is an acceptable number of outliers or wobbliness for the table. This will also help determine if the table must be replaced since the student is also playing a competitive shooter which can also be applied in an actual competitive environment.

As we can see in Figure 16, Figure 17, and Figure 18, it could probably be observed that the X and Y values give the lesser number of outliers but are more compressed together. While in Z, there are more outliers, but they are more spread out. However, as the wobble

of the table moves in the left and right direction, it could be said that the value of Z should be of X, as the orientation of the accelerometer is faced down, but for now it would still be indicated as Z.

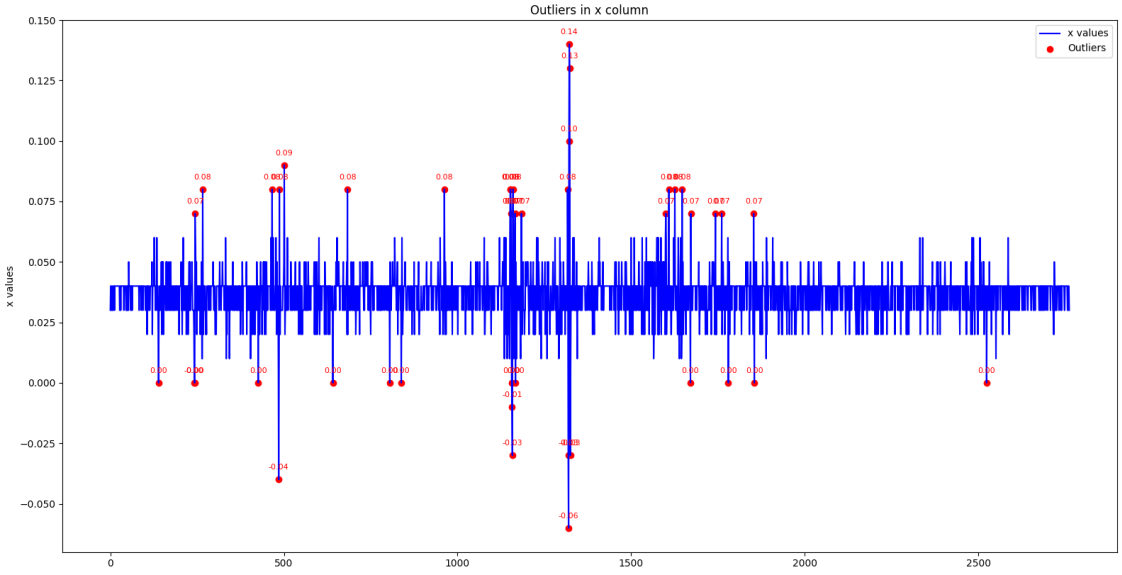


*Figure 16*

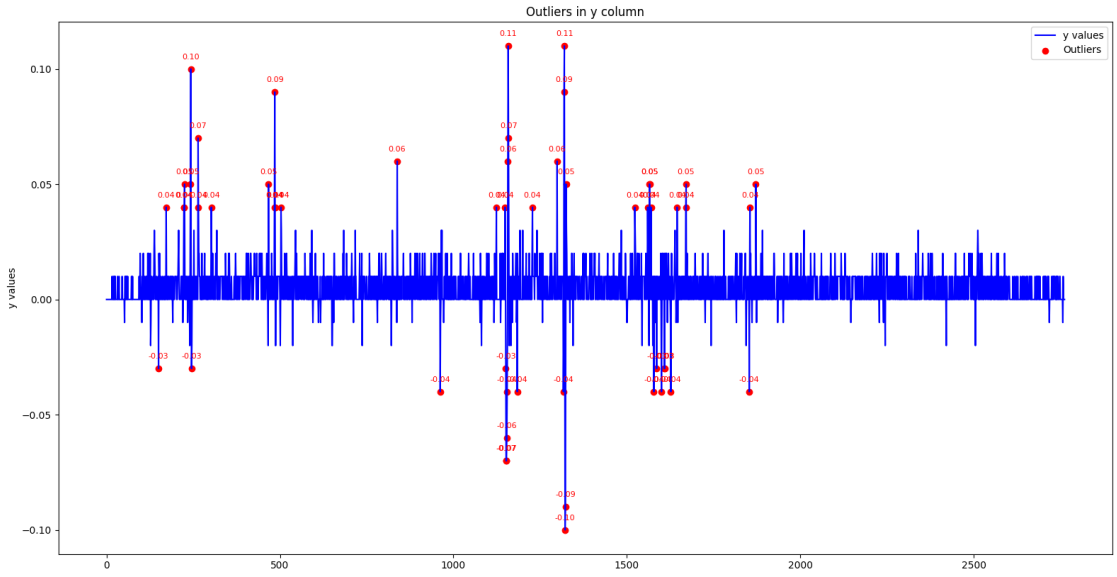


*Figure 17*

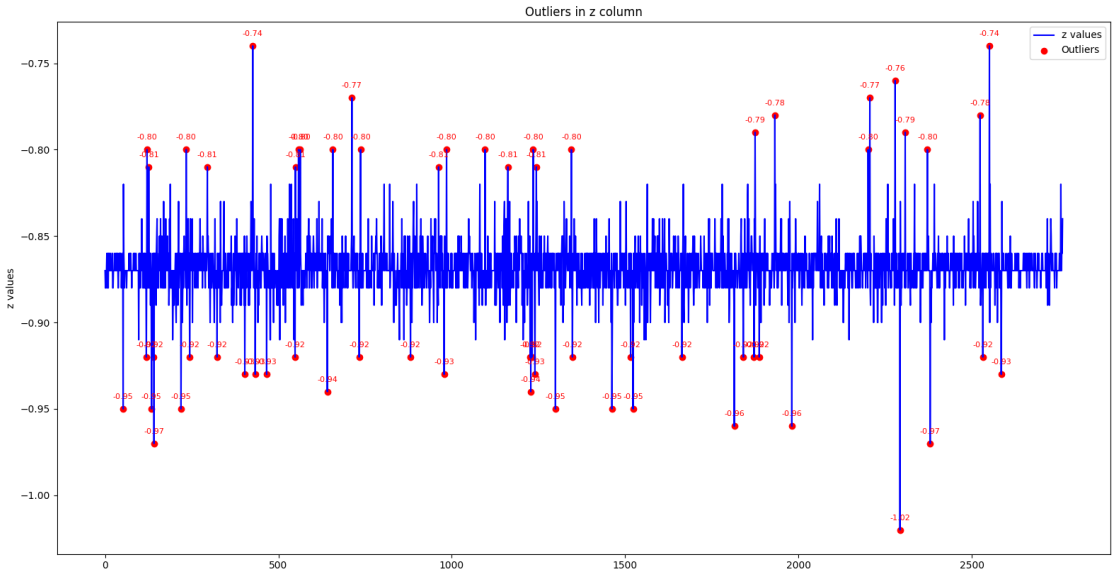


*Figure 18*

To confirm if the observations are correct based on the number of outliers, we will get the total outliers per column divided by the number of values per column. In which, we will see if there are more than 5% of outliers per column as the student will not want noticeable movements of the table when playing as it could affect his performance in playing competitive video games.

It can be seen in Figure 19 that the total amount of outliers per X, Y, and Z values are all less than 5% and less than 2.5% which is more than acceptable for the student as he will not need to replace anything in his setup such as his table's wobbliness is unnoticeable especially when playing. Note that when the student was playing, he had his chest near the table which could have helped with the table's stability. However, in higher competitive environments like the semi-professional or professional level, anything miniscule as this can affect performance of users as players in the higher competitive levels play in well set-up systems where tables that are used are tougher and remain stable even when players create drastic movements with their mouse on the table as seen in Figure 20 for them to be fully focused on the game only, maximizing performance.

```
The percentage of outliers in x is within the acceptable range: 1.63%
The percentage of outliers in y is within the acceptable range: 1.95%
The percentage of outliers in z is within the acceptable range: 2.24%
```

*Figure 19*


*Figure 20*

## IV. Conclusion

In conclusion, the students were able to successfully make a functionable and useful Vibration Data Collector system that can collect vibration data in multiple scenarios such as to the structural integrity of an object or in this case, the students tested the wobbliness of table when playing a competitive first-person shooter. The students were also able to make an Arduino program that prints out the raw data gathered from the ADXL345 accelerometer in JSON format in which a Python program can parse the printed outputs and save the data into a CSV file. In which, the students have learned the importance of accelerometers and how they can be used in different fields and environments such as Science and Engineering or even personal setups or scenarios to test the stability of something.

## V. Appendix

### A. Arduino Code

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>

Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12346);
unsigned long previousTime = 0;

void setup(void) {
  Serial.begin(115200);
  Serial.println("Accelerometer Test"); Serial.println("");
```

```
  if(!accel.begin())
  {
    Serial.println("ADXL345 was not detected ... please check wiring");
    while(1);
  }

  Serial.println("");
}

void loop() {
  unsigned long currentTime = millis();

  if (currentTime - previousTime == 100) {
  previousTime = currentTime;

  sensors_event_t event;
  accel.getEvent(&event);

  float X = event.acceleration.x / 9.80665;
  float Y = event.acceleration.y / 9.80665;
  float Z = event.acceleration.z / 9.80665;

  Serial.print("{\"time\": ");
  Serial.print(currentTime);
  Serial.print(", \"x\": ");
  Serial.print(X);
  Serial.print(", \"y\": ");
  Serial.print(Y);
  Serial.print(", \"z\": ");
  Serial.print(Z);
  Serial.println("}");
  }
}
```

### B. Python Code

```
import serial, csv, json, os
from datetime import datetime

filename = f'vibration_logs_{datetime.now().strftime("%Y%m%d_%H%M%S")}.csv'

serport = serial.Serial('COM8', 115200)

firsttime = None

with open(filename, 'w', newline='') as csvfile:
    fieldnames = ['time', 'x', 'y', 'z']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

    try:
        while True:
            buffer = serport.readline().decode().strip()
            try:
                data = json.loads(buffer)

                if firsttime is None:
                    firsttime = data['time']

                adjusted_time = data['time'] - firsttime
                data['time'] = adjusted_time

                print(f"Data: {data}")
```

```
                writer.writerow(data)
            except json.JSONDecodeError as e:
                continue

    except KeyboardInterrupt:
        pass
    finally:
        serport.close()

print(f"Program stopped. The CSV file is saved in {os.path.abspath(filename)}.")
```

## VI. References

[1] v. l. R. &. Automation, "13 Accelerometer (ADXL345) with ESP32," YouTube, 5 March 2020. [Online]. Available: https://www.youtube.com/watch?v=s5Ne0hIaJws. [Accessed 21 July 2024].

[2] R. Pelayo, "How to Use ADXL345 Accelerometer with Arduino," TeachMeMicro, June 11 2019. [Online]. Available: https://www.teachmemicro.com/use-adxl345-arduino/. [Accessed 20 July 2024].

[3] B. Earl, "ADXL345 Digital Accelerometer," Adafruit Learning System, [Online]. Available: https://learn.adafruit.com/adxl345-digital-accelerometer/overview. [Accessed 20 July 2024].

[4] "ADXL345," Analog Devices, 2023. [Online]. Available: https://www.analog.com/en/products/adxl345.html#part-details. [Accessed 20 July 2024].

[5] "ADXL345 Accelerometer Module," Components 101, [Online]. Available: https://components101.com/modules/adxl345-accelerometer-module. [Accessed 20 July 2024].