ECE LABORATORY

DREXEL UNIVERSITY

To: **Dr. Peters**

From: **Ehi Simon**

Re: **ECE 304 Lab 1**

PURPOSE:

The purpose of this week's lab is to build and test the circuit we will use for the rest of the term. It focuses on developing the basic circuits that will support my labs for most of the term.

.

Discussion:

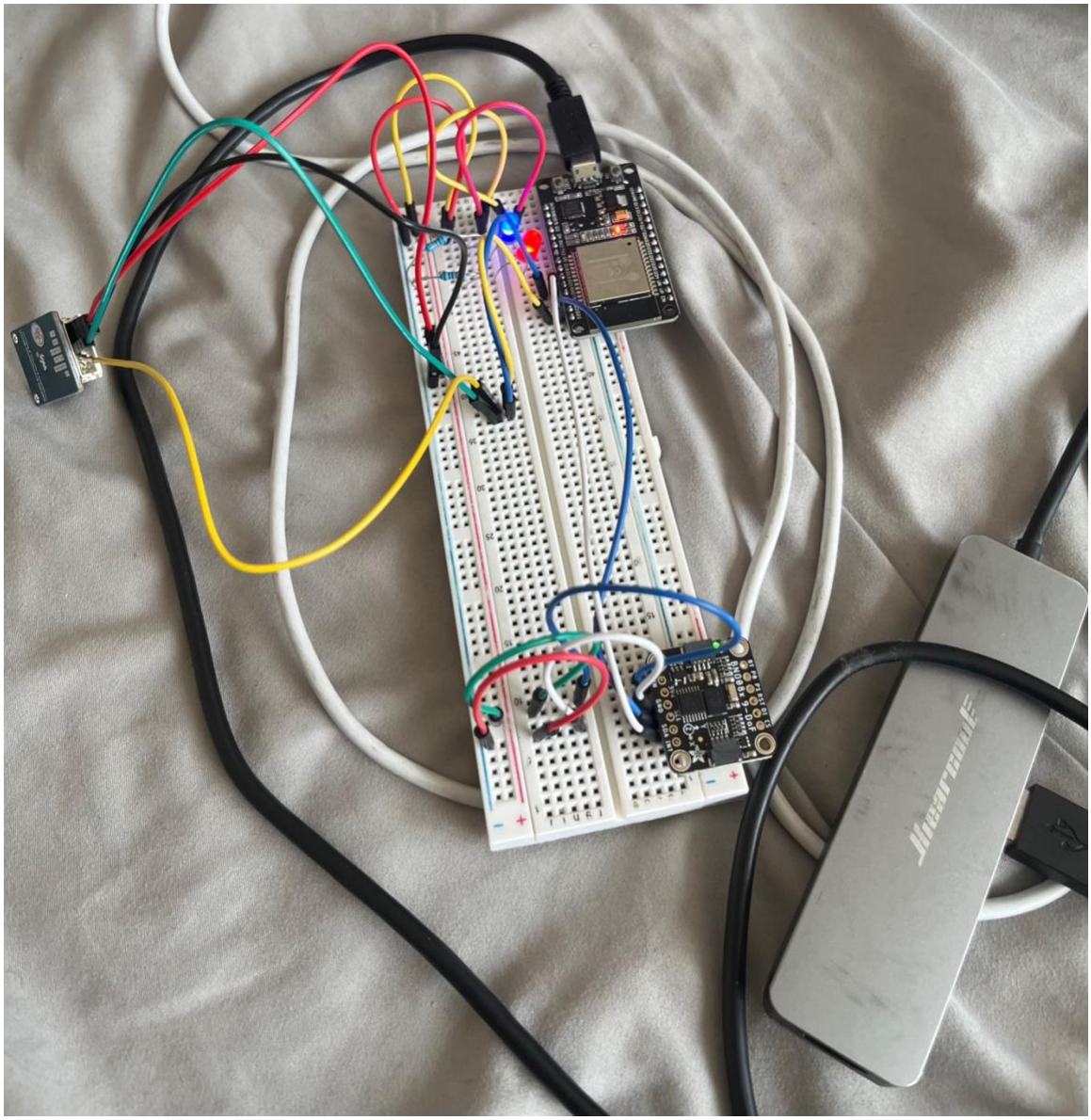

Fig. 1. Circuit Connection for Project 1

The circuit for the lab was built like the one above. It consists of 2 330Ω resistors, a red LED, a blue LED, a BME280 Environmental Sensor, an Adafruit BNO085 IMU, and an ESP32S microcontroller.

To test if the circuit is working, an integer value between 0 and 8191 is entered using the serial monitor. Additionally, the yaw, pitch, and roll readings are gotten from the IMU. The integer value entered by a user represents the functionality of the circuit.

Main.cpp

In my main.cpp file, I initialized the multiple libraries that were needed for the sensors to work and provide readings. The main function consisted mostly of the setup and loop functions.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(blue_LED_pin, OUTPUT);
  pinMode(red_LED_pin, OUTPUT);
  ledcSetup(ledChannel, freq, resolution);
  ledcAttachPin(red_LED_pin, ledChannel);

  bool status;
  status = bme.begin(0x76);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
  while (1);
  }
    while (!Serial) delay(10);     // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit BNO08x test!");

  // Try to initialize!
  if (!bno08x.begin_I2C()) {
    Serial.println("Failed to find BNO08x chip");
    while (1) { delay(10); }
  }
  Serial.println("BNO08x Found!");


  setReports(reportType, reportIntervalUs);

  Serial.println("Reading events");
  delay(100);
  Serial.println("Enter A Number between 0 and 8191: ");
}
```

Fig. 2. Setup of the ESP32 Code

The figure above is concerned with the setup of the ESP32. The “serial.begin” line starts serial communications with a 115200-bps baud rate. The next two lines set the pins for the LEDs for OUTPUT. The two lines below those are specially for the ESP32. They are used to set up the LED behavior for the red LED.

The next few lines start the BME280, and check if the BME280 is found. They also start the BNO085 chip, and check if it is found. After this in the setup function, we also print out a prompt for the user to enter an integer between 0 and 8191.

```
void loop() {
  // Read command from serial monitor
  if (Serial.available() >0){
    int command = Serial.parseInt();
    while (Serial.available()>0){
      char t;
      t = Serial.read();
    }
    Serial.print("Command Entered: ");
    Serial.println(command);
    if ((command < 0) || (command> 8191)){
      Serial.println("ENTER A NUMBER FROM 0 TO 8191");
    }
    else{
      if(bitRead(command,0)){
        Serial.print("Temperature = ");
        Serial.print(bme.readTemperature());
        Serial.println(" *C");
      }

      if(bitRead(command,1)){
        Serial.print("Humidity = ");
        Serial.print(bme.readHumidity());
        Serial.println(" %");
      }

      if(bitRead(command,2)){
        Serial.print("Pressure = ");
        Serial.print(bme.readPressure());
        Serial.println(" hPa");
      }

      if(bitRead(command,3)){
```

Fig. 3. Section of the Loop Function

The figure above shows the main section of the loop function. It starts with the line “int command = Serial.parseInt()” reading a sequence of characters from the serial monitor, converting them into an integer value, and assigning it to the variable command. Then, it prompts the user to enter a number between 0 and 8191. The rest of this section is comprised of if statements reading the command and giving an output based on the command. The output is then printed. The output received can be found in the table below:

Bit	Description
0 (LSB)	Temperature: (0/1 for no measurement/measurement)
1	Humidity: (0/1 for no measurement/measurement)
2	Pressure: (0/1 for no measurement/measurement)
3	Altitude: (0/1 for no measurement/measurement)
4	Blue LED: (0/1 for OFF/ON)
5-12 (MSB)	Blue LED intensity (Leftmost 8 bits for 0-255)

Table 1. Table Showing Command Bits and Their Descriptions

The figure below shows the next main section of the loop function. This section prints the yaw, pitch, and roll as well as the time and accuracy of the sensor values.

```
if (bno08x.getSensorEvent(&sensorValue)) {
  quaternionToEulerRV(&sensorValue.un.arvrStabilizedRV, &ypr, true);

  static long last = 0;
  long now = micros();
  Serial.print(now - last);             Serial.print("\t");
  last = now;
  Serial.print(sensorValue.status);     Serial.print("\t");  // This is accuracy in the range of 0 to 3
  Serial.print(ypr.yaw);                Serial.print("\t");
  Serial.print(ypr.pitch);              Serial.print("\t");
  Serial.println(ypr.roll);
}
  }
```

Fig. 4. Section of the Loop Function

In the end, the output comes out like this:

```
Enter A Number between 0 and 8191:
66
Command Entered: 66
Humidity = 48.18 %
Blue LED Value: 0
Red LED Value: 2

29757583        3       -15.36  -0.94   -2.27
Enter A Number between 0 and 8191:
78
Command Entered: 78
Humidity = 48.17 %
Pressure = 100391.16 hPa
Altitude = 78.04 m
Blue LED Value: 0
Red LED Value: 2

3945220 3       -15.36  -0.94   -2.27
```

Fig. 5. Figure Showing Output After Entering Command

Conclusion

This lab helped me understand how to integrate and interface with sensors like the BME280 to measure temperature, humidity, pressure, and altitude. It was also my first time working with the BNO085 chip, and it showed me how to work with the IMU.