Vehicle Telemetry System (Working Title)

By: Kyele Haynes, Kogul Balasubramaniam, Samuel Dadet

# Declaration of Joint Authorship

We hereby declare that this technical report contains material that is the result of joint collaboration on the project. All of the team member have done their own research on the hardware and software components utilized in the project. As per the individual sensors used in the project (defined under requirements specifications), each team member has done their own extensive research on the sensor they have been assigned as well as performed data collection, analysis and interpretation on said sensor. Any key ideas involving each individual team members sensor has been written and drafted by themselves with critical revision from both the other team members. Research, primary contributions and experimental designs in regard to the PCB, physical wiring, enclosures, Raspberry Pi and Raspbian operating system will all be conducted, led, written and drafted by Kyele while taking into considerations any suggestions from other team members. The same outlines go for Sam and Kogul in regard to the android development and data linking portion of the project. As per the design and documentation of the database, all team members will help write and draft these sections. We declare that everything written in this technical report is the product of our own ideas and research. Any information that has been used or referenced from another source have been cited under the references section of the report. Finally, we, to the best of our knowledge claim that this report does not defy any copyright laws.

# Approved Proposal

# Abstract

In today’s modern society off road cars have always been a popular method of racing on designated tracks. They provide a thrill to the driver and can be very entertaining when done safe. But since it is like any other mechanical machine it tends to develop problems. There are many problems that could develop throughout the lifespan of a car and our telemetry system is intended to detect certain issues or keep track of certain data that could be beneficial to the cars mechanics or just give any information that the owner might request. The driver would be able to make quick and basic repairs based on the information that our system would provide saving time, energy ando cost. Our system involves four crucial components (described in the requirement specifications). Each sensor will be programmed to collect certain crucial data for the vehicle and will record all of it into a database where it can be accessed by the driver whether it be past or present. The data can be viewed through our mobile application on any android device where the user can log in to get access to there vehicles data. It will also record the data in some type of graph so they can compare past and future information to see whether there was a decline in performance. Thanks to the simplicity of our casing (described in the report) our system will be able to be installed simply to any vehicle. Our design is being set out to make the consumers life a bit easier as well as help the driver know the condition of their vehicle no matter where they are.

# Table of Contents

[Declaration of Joint Authorship 3](#_gjdgxs)

[Approved Proposal 5](#_30j0zll)

[Abstract 7](#_1fob9te)

[Table of Contents 9](#_3znysh7)

[List of Illustrations 11](#_2et92p0)

[Introduction 13](#_3dy6vkm)

[The Technical Issue/Problem 13](#_1t3h5sf)

[Why Solve It? 13](#_4d34og8)

[Scope 13](#_2s8eyo1)

[Objective 13](#_17dp8vu)

[Interesting Issue Discoveries 13](#_3rdcrjn)

[Unique Approaches 13](#_26in1rg)

[Project Description 15](#_lnxbz9)

[Requirements Specifications 15](#_35nkun2)

[Software 15](#_1ksv4uv)

[Hardware 15](#_44sinio)

[Conclusion 17](#_2jxsxqh)

[Recommendations 19](#_z337ya)

[Bibliography 21](#_3j2qqm3)

[Appendices 23](#_1y810tw)

# List of Illustrations

# Introduction

Baja Racing is a design competition hosted by the Society of Automotive Engineers (S.A.E). It's a tournament where various college and universities student come together to compete in a dirt road type of track, building their own vehicle to the specifications of the S.A.E board. Most of the students who participate in this competition are mostly automotive engineering students. Most students are focused on one path only and have very little knowledge of electronics and computer programming which would put them at a disadvantage if there is any issues that develop because of those disadvantages.

The objective of this telemetry system is to make life easier for the buggy racers in detecting fault in their vehicle, it would be a fair and just race if all buggy vehicles had our telemetry system in it. leveling the playing field for all participants. some of the unique approach about the telemetry system is that it can be used in smart vehicles and not just on buggy races even in other means of transportation like ships, airplanes and trains. In buggy races, a lot of drivers are stuck with mechanical, electrical, functional issues that they have no clue where or how to deal with. Some don’t even know what is wrong with their vehicle, it just won’t start. With our telemetry system, all those worries, and woes will not be present as you have exactly what is wrong with your vehicle right there on your app.

With our telemetry system, the driver would be able to detect exactly what is wrong with their vehicle. Our device would provide real time data which is being fed into the driver’s app and would notify the driver of any changes that should be done prior to start of race. Priority is given to the most crucial problem, indicating to the driver what is hindering the total usability of the vehicle. With the temperature sensor, if the engine is overheating, the BME sensor would send data to the phone app indicating to the driver that something has to be done immediately to ensure smooth sailing of the vehicle.

# Project Description

## Requirements Specifications

### Software

There are a few different types of software required for the project. The programming languages we will be using are Java (Android), Python and C. For the operating system on the Raspberry Pi we will be using Raspbian included with programming IDEs compatible with python and C and XRDP for remote access. All the software and connections on the Raspberry Pi will be setup by Kyele. The programming aspect of retrieving data from each of the sensors will be done as follows; Kyele – AMG8833, Kogul – PCF8591, Samuel – BME280. To develop the android application the most recent version of Android Studio is required as well as emulators running multiple different versions of android, Kit-Kat (4.4) or higher. Most of the development work for the application will be completed by Kogul and Samuel. Lastly, a firebase database that is capable of holding records every 5 minutes, 24/7 for up to 1 month is required which will be designed, created, and maintained as a joint effort.

### Hardware

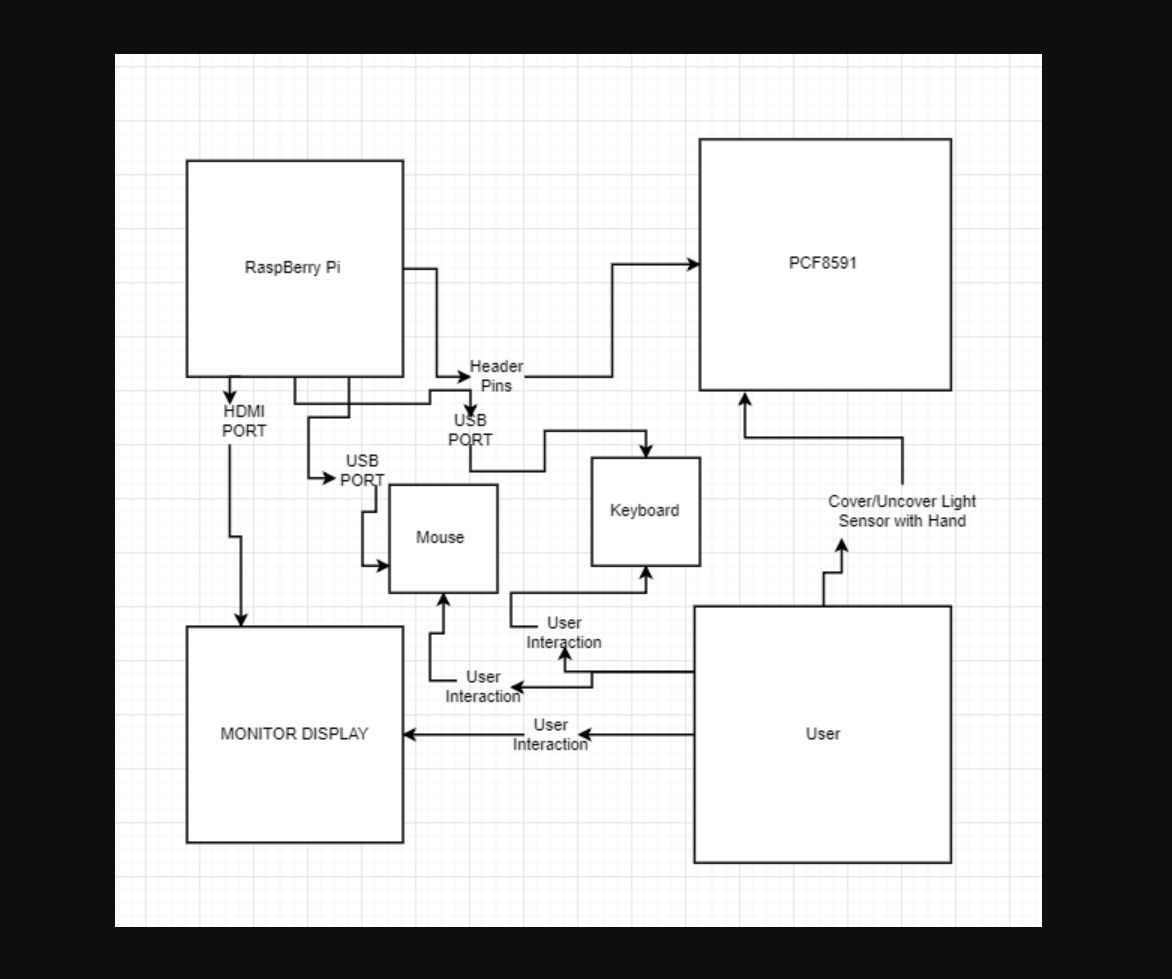
The hardware includes a few main aspects which are; 3 different breakout board sensors, a Raspberry pi 3 (the development platform), smartphone running Android Kit-Kat (4.4) or higher and a 3D printed enclosure. Some other hardware requirements are an 8GB or higher micro SD card, a micro USB power cable for use with Raspberry Pi 3, PCB materials for connecting the sensors to the development platform and wall mounting brackets. As per the hardware work breakdown, each member is responsible for their own sensor; Kyele – AMG8833, Kogul – PCF8591, Samuel – BME280. They each have to completely understand how it functions and how it interacts with the development platform. This includes how it sends data, how it is powered and maximum and minimum specifications it can handle. As per the rest of the hardware, Kyele is responsible for designing the PCB required for connecting the sensors to the development platform with the help Sam and Kogul with their knowledge of the other sensors. He is also responsible for maintaining the Raspbian OS installed on the Raspberry Pi 3, such as installing any newly introduced software and performing updates and security patches. Each team member will have complete access to the Raspbian Operating System as we will all need to work on the software connections to the sensors. Lastly, all team members will work together on designing the final 3D printed enclosure.

# Build Instructions

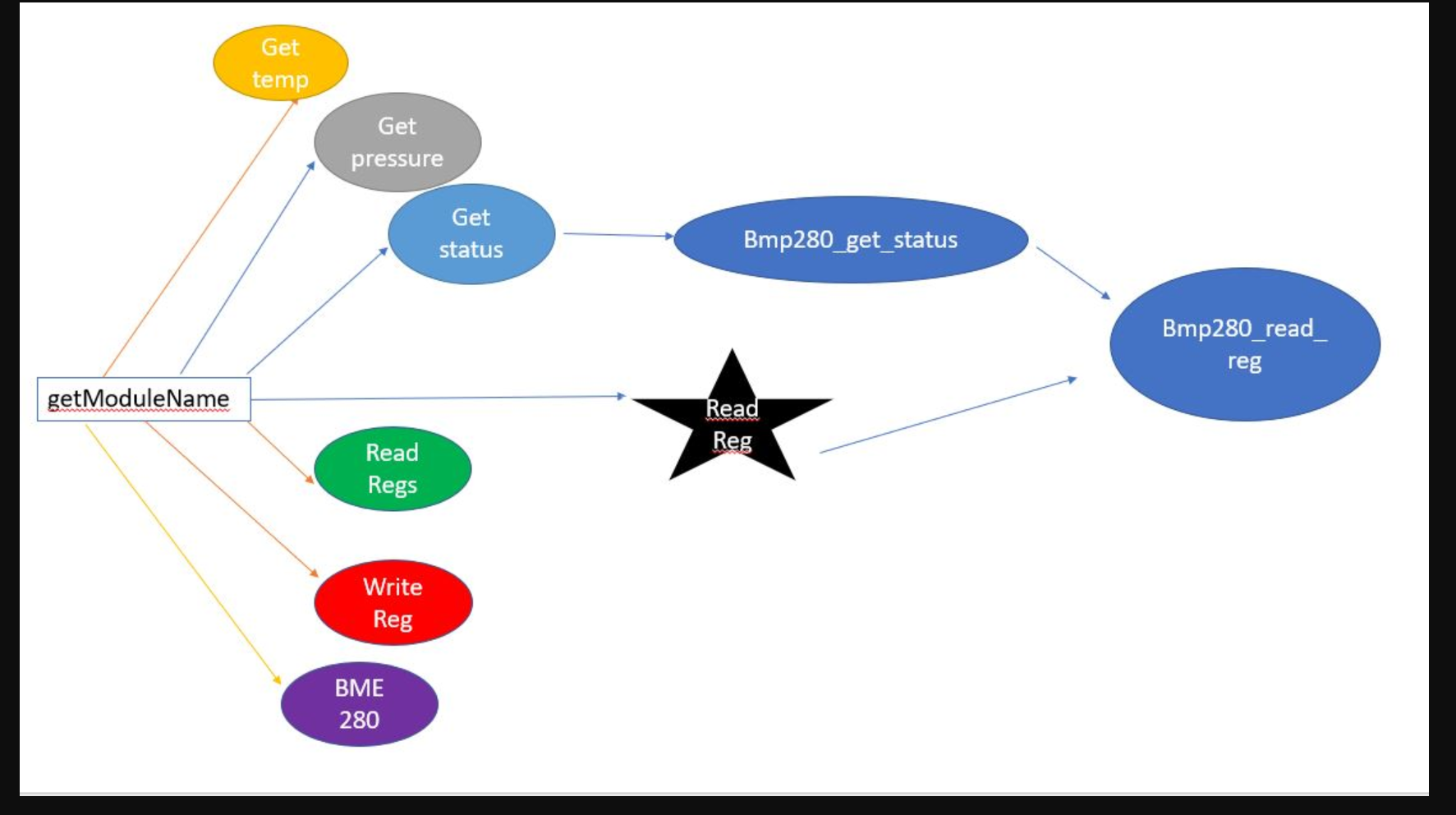
## Introduction

The purpose of this project is to build a telemetry system for a Baja racing vehicle. There are three sensors involved in this project which include a PCF8591, BME280 and AMG8833. This is a collaborative project that involves up to 3 members who each contributed to at least one sensor and the overall dynamics of this project. We put together a [Budget Plan](https://github.com/KogulB/CENG355Project/blob/master/documentation/Budget.xlsx) and a project [schedule](https://github.com/KogulB/CENG355Project/blob/master/documentation/Schedule.mpp) to help us complete the project in a successful manner.

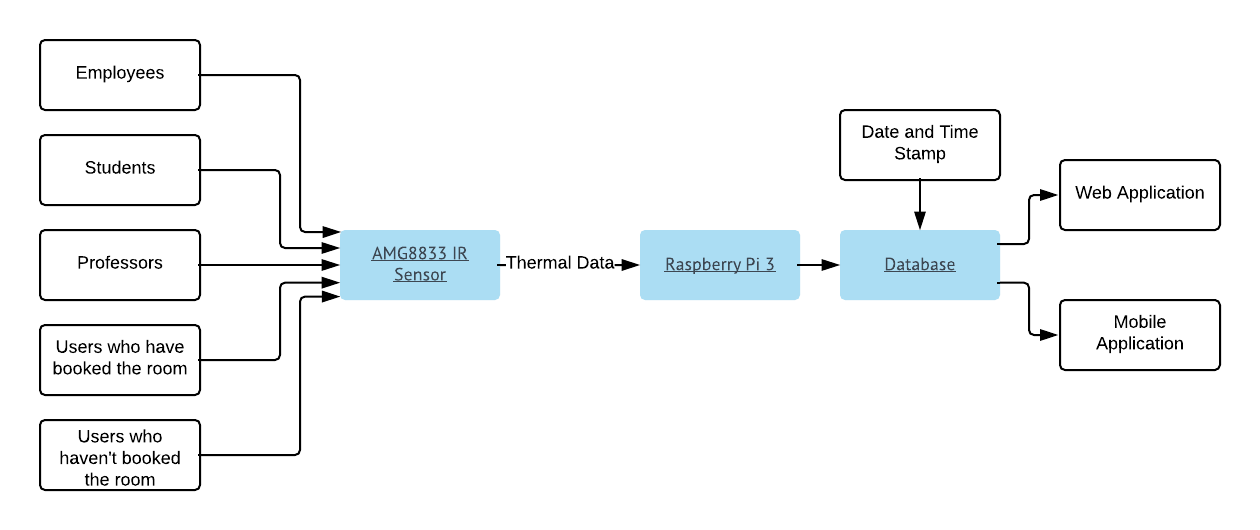
## System Diagrams



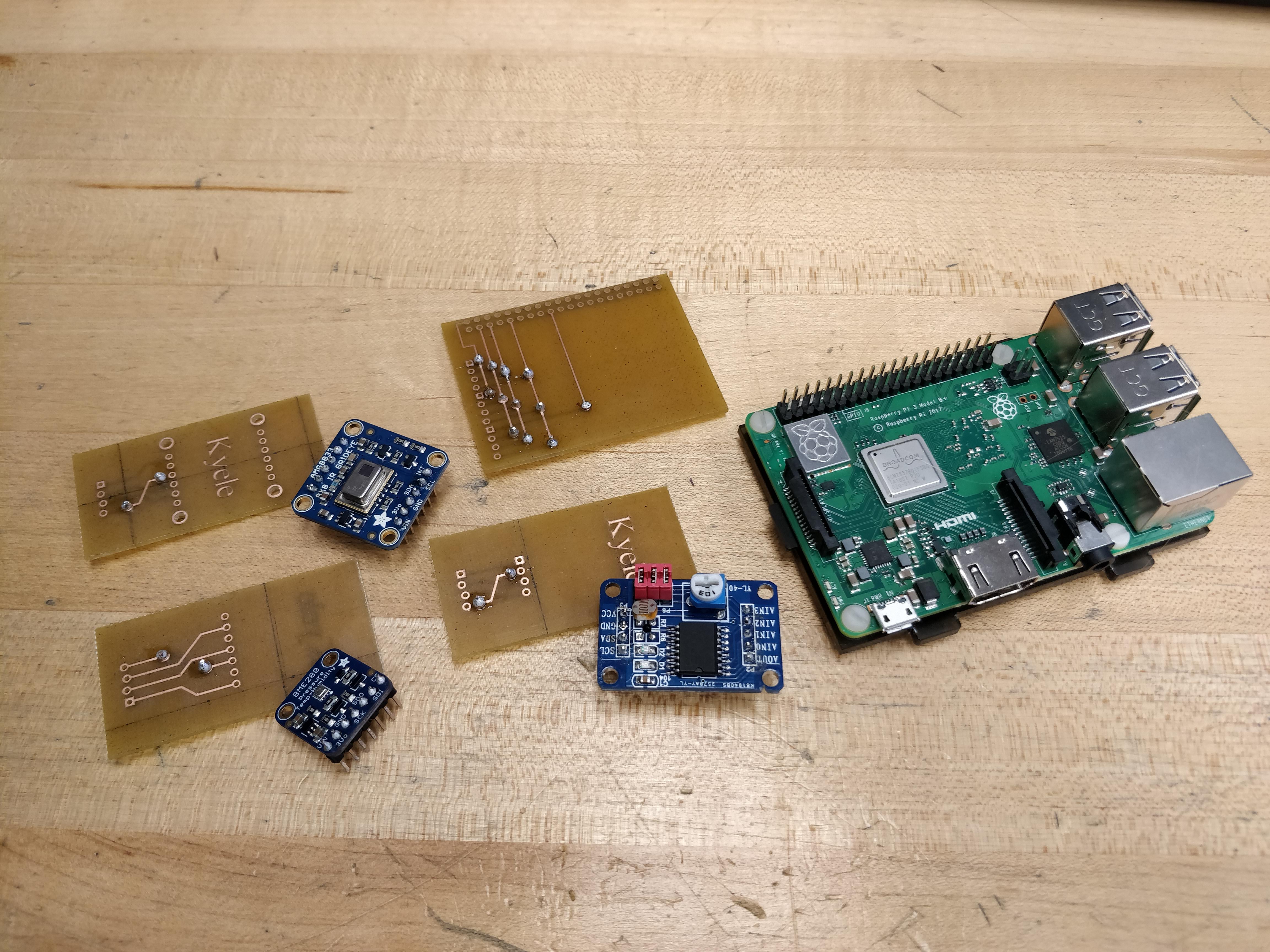
System Diagram 1



System Diagram 2



System Diagram 3



Raspberry Pi Enclosure

## Budget Plan/Bill

The main components of the project were a Raspberry Pi 3 Kit which included a power cable and a default case. Memory card would have to be purchased separately but for our case we had one already. Other purchases made were the sensors and devices needed to run everything: PCF8591, BME280, AMG8833, Ethernet cable and adapter. An HDMI cable for initial setup and a breadboard for testing will also need to be purchased but in our case, we already had one of each. There are however some extra parts that were purchased for testing but were never user used. The link to the budget plan can be found [here](https://github.com/KogulB/CENG355Project/blob/master/documentation/Budget.xlsx).

## Time Commitment

It took about two hours per sensor for us to complete the research portion of our project because we each did our own research on a different sensor to find all the components and code needed to get the ICs up and running. So, in total research took about 6 hours (2 x 3 sensors).

The materials could take anywhere from 2-5 days to ship if they are ordered from Amazon and depending on the shipping method used. The sensors were however purchased from different websites (Links in the Budget) and took about 1 week to arrive.

After getting all the components listed in the budget, the next step is to test the connection from each sensor to the Raspberry Pi using a breadboard. Make sure to test each sensor separately to not short circuit the sensors. This should take roughly 2 hours.

After figuring out the connections, it took Kyele one hour creating the connections on Fritizing. He then spent approximately 2 more hours redesigning and moving around connections on the PCB by adding vias, making sure the connections don’t overlap and making the PCB overall more efficient. Luckily for anyone else attempting this project the Fritzing files are already created and are included in the Git repository.

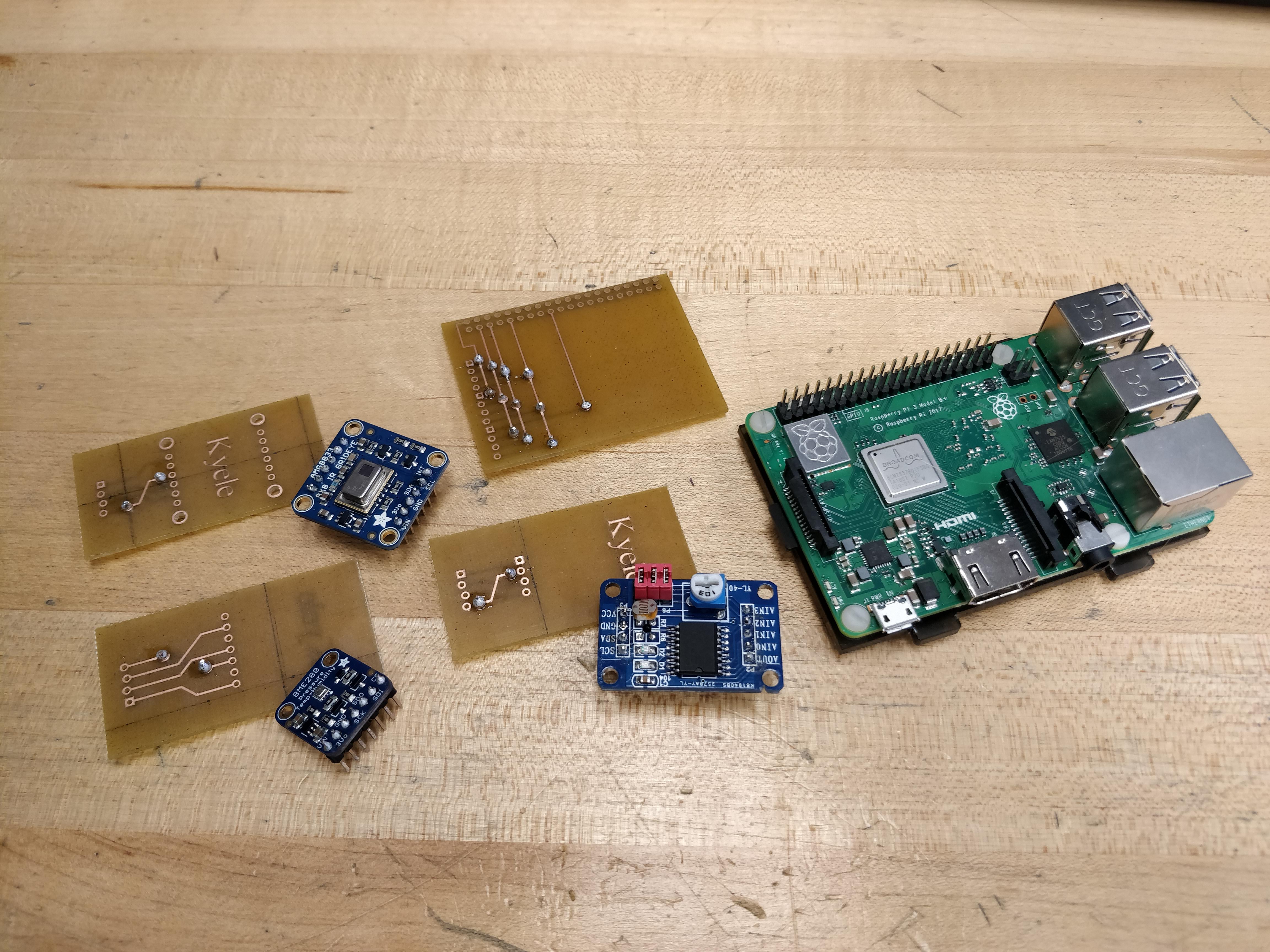
Setting up the code for testing is not hard at all. For the PCF8591 sensor, code can be pulled from the StudentSenseHat Git repository. Since the PCF8591 was used in that project, testing took about 30 minutes. For the BME280 sensor, the test code was found online from the following [website](https://github.com/adafruit/Adafruit_CircuitPython_BME280/blob/master/examples/bme280_simpletest.py). The AMG8833 test code was also found online at this [link](https://github.com/adafruit/Adafruit_CircuitPython_AMG88xx/blob/master/examples/amg88xx_simpletest.py).

Total Commitment: 11 hours (excluding shipping)

## PCB/Soldering

Cut off the pins located on the sensors and solder new header pins on both sides so connection to a breadboard becomes possible.

solder new header pins on to PCB so connections to Rpi and sensor is possible.



Soldered Parts

## Mechanical Assembly

To be added later as all three sensors have not been connected together on one board.

## Power Up

To be added when all parts are working on one enclosure

## Production Testing

After powering on the Raspberry Pi, make sure you test if the addresses of the sensors are correct;

1. Connect the Pi to a display using an HDMI cable/adapter
2. Turn on the Pi (wait for it to boot up)
3. Open a terminal window
4. Run the command sudo raspi config
5. Enable I2C Address
6. Exit out
7. Run sudo reboot
8. After reboot open another terminal window
9. Run the command sudo I2Cdetect -y 1
10. It should show that the addresses that are connected are 0x48, 0x69 and 0x77

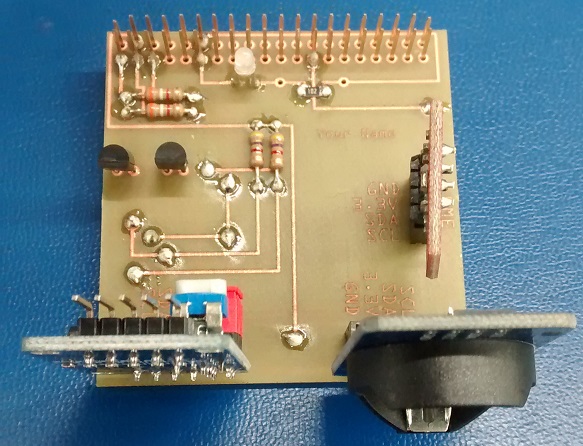
[Example](https://raw.githubusercontent.com/KogulB/KogulBCENG317Project/master/Images/Assigned.PNG)

## Unit Testing

To be added when testing is done roughly in about two weeks

## 9 Mass Production

If there was to be a mass production of this project, we believe that soldering the sensors directly to the PCBs would be much more efficient. This way, the case would be smaller allowing for a cheaper overall cost and smaller footprint when in use. Smaller connectors for the wiring harness between all of the PCBs would also allow for a smaller footprint.



StudentSenseHat

Source (https://raw.githubusercontent.com/six0four/StudentSenseHat/master/images/39.jpg)

# Conclusion

# Recommendations

# Bibliography

# Appendices