|  |
| --- |
| Ryan McAdie |

R.A.D Technologies Greenhouse Monitoring System

Computer Engineering Technology

# Declaration of Joint Authorship

We, Ryan McAdie, Daniel Bujold, and Aiden Bolos, confirm that this work submitted is the joint work of our group and is expressed our own words. Any uses made within it of the works of any other author, in any form (ideas, equations, figures, texts, tables, programs), are properly acknowledged at the point of use. A list of the references used is included. The work breakdown is as follows: Each of us provided functioning, documented hardware for a sensor or effector. Ryan McAdie provided documentation for the BME680 Gas and Air Quality Sensor. Daniel Bujold provided documentation for the Capacitive Moisture Sensor. Aiden Bolos provided documentation for the DS18B20 Temperature Sensor. In the integration effort Ryan McAdie is the lead for further development of our mobile application, Daniel Bujold is the lead for the Hardware, and Aiden Bolos is the lead for connecting the two via the Database.

# Proposal

We have created a mobile application, worked with databases, completed a software engineering course, and prototyped a small embedded system with a custom PCB as well as an enclosure (3D printed/laser cut). Our Internet of Things (loT) capstone project uses a distributed computing model of a smart phone application, a database accessible via the internet, an enterprise wireless (capable of storing certificates) connected embedded system prototype with a custom PCB as well as an enclosure (3D printed/laser cut), and are documented via this technical report targeting OACETT certification guidelines.

Intended project key component descriptions and part numbers

Development platform: Raspberry Pi (Broadcom Development Platform)

Sensor/Effector 1: BME680

Sensor/Effector 2: Capacitive Moisture Sensor EK1940

Sensor/Effector 3: DS18820 Temperature Sensor

We will continue to develop skills to configure operating systems, networks, and embedded systems using these key components to complete a small-scale system of a greenhouse monitoring system that will be capable of accurately retrieving and displaying real time up to date vital information for the greenhouse environment. Systems like this currently exist in the world today, however looking at Humber's current system we noticed that all changes can only be made from a central computer located in the greenhouse, we would like to incorporate a remote monitoring and allow remote changes to our system for convenience and to always know how the greenhouse is doing. We also plan to incorporate systems that can be accessed remotely from inside the greenhouse using our related mobile application. Such systems we currently have in mind are; an irrigation system that can automatically or manually water the plants as needed, a ventilation system that can be accessed to regulate temperatures inside the greenhouse and a blind system that can be used to block intense light and heat from the sun if needed for the plants.

Our project description/specifications will be reviewed by, Valeria an employee of the Humber Greenhouse, ideally an employer in a position to potentially hire once we graduate. They will also ideally attend the ICT Capstone Expo to see the outcome and be eligible to apply for NSERC funded extension projects. This typically means that they are from a Canadian company that has been revenue generating for a minimum of two years and have a minimum of two full time employees.

The small physical prototypes that we build are to be small and safe enough to be brought to class every week as well as be worked on at home. In alignment with the space below the tray in the Humber North Campus Electronics Parts kit the overall project maximum dimensions are 12 13/16" x 6" x 2 7/8" = 32.5cm x 15.25cm x 7.25cm.

Keeping safety and Z462 in mind, the highest AC voltage that will be used is 16Vrms from a wall adapter from which+/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will not exceed 20 Watts. We are working with prototypes and that prototypes are not to be left powered unattended despite the connectivity that we develop.

# Executive Summary

With our greenhouse system users will be able to monitor real time vital information to their greenhouses such as temperature, humidity and soil moisture. These values will be updated in real time so users will always know the state of the greenhouse and if changes need to be made. We are also going to implement a remote management system for certain features of the greenhouse such as an irrigation/feeding system, ventilation and fan system and a light blocking or blind system. With all these, a user would be able to take manual control of the greenhouse and make remote changes to the system that could potentially be better for the plants then just using the automated system. Our system will be one of the first of its kind to implement all the necessary features of a greenhouse monitoring system as well as go above and beyond to achieve features and requirements that would make managing a greenhouse a little more easier with more peace of mind knowing that your plants are always in your hands.

Contents

[Declaration of Joint Authorship 3](#_Toc33010457)

[Proposal 5](#_Toc33010458)

[Executive Summary 7](#_Toc33010459)

[List of Figures 11](#_Toc33010460)

[1.0 Introduction 13](#_Toc33010461)

[1.1 Scope and Requirements 13](#_Toc33010462)

[2.0 Background 19](#_Toc33010463)

[3.0 Methodology 21](#_Toc33010464)

[3.1 Required Resources 21](#_Toc33010465)

[3.1.1 Parts, Components, Materials 21](#_Toc33010466)

[3.1.2 Manufacturing 24](#_Toc33010467)

[3.1.3 Tools and Facilities 26](#_Toc33010468)

[3.1.4 Shipping, duty, taxes 28](#_Toc33010469)

[3.1.5 Time expenditure 29](#_Toc33010470)

[3.2 Development Platform 30](#_Toc33010471)

[3.2.1 Mobile Application 30](#_Toc33010472)

[3.2.2 Image/firmware 35](#_Toc33010473)

[3.2.3 Breadboard/Independent PCBs 37](#_Toc33010474)

[3.2.4 Printed Circuit Board 47](#_Toc33010475)

[3.2.5 Enclosure 47](#_Toc33010476)

[3.3 Integration 47](#_Toc33010477)

[3.3.1 Enterprise Wireless Connectivity 48](#_Toc33010478)

[3.3.2 Database Configuration 48](#_Toc33010479)

[3.3.3 Security 48](#_Toc33010480)

[3.3.4 Testing 48](#_Toc33010481)

[4.0 Results and Discussions 49](#_Toc33010482)

[5.0 Conclusions 51](#_Toc33010483)

[6.0 References 53](#_Toc33010484)

[7.0 Appendix 55](#_Toc33010485)

[7.1 Firmware code 55](#_Toc33010486)

[7.2 Application code 55](#_Toc33010487)

# List of Figures

[Figure 1 DS18B20 Temperature Sensor. This work is a derivative of: https://components101.com/sensors/ds18b20-temperature-sensor by components101. 23](#_Toc33010488)

[Figure 2 Raspberry Pi 4 This work is a derivative of “https://www.raspberrypi.org/products/raspberry-pi-4-model-b/” by Raspberry Pi 24](#_Toc33010489)

[Figure 3 PCB design This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 25](#_Toc33010490)

[Figure 4 Enclosure case; This work is a derivative of "https://inkscape.org/" by inkscape. 26](#_Toc33010491)

[Figure 5 Enclosure case; This image was taken by Aiden Bolos. 26](#_Toc33010492)

[Figure 6 Android Studio. (01,2020). Screenshot from login page of GreenSense mobile application. Screenshot by Ryan McAdie 32](#_Toc33010493)

[Figure 7 Android Studio. (01,2020). Screenshot from values page of GreenSense mobile application. Screenshot by Ryan McAdie 33](#_Toc33010494)

[Figure 8 Android Studio. (01,2020). Screenshot from devices page of GreenSense mobile application. Screenshot by Ryan McAdie 34](#_Toc33010495)

[Figure 9. Air Quality/Humidity Initial schematic. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 38](#_Toc33010496)

[Figure 10. Temperature Initial schematic. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 39](#_Toc33010497)

[Figure 11. Soil Moisture Initial schematic. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 39](#_Toc33010498)

[Figure 12. Air Quality/Humidity Complete Breadboard. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 40](#_Toc33010499)

[Figure 13. Temperature Complete Breadboard. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 41](#_Toc33010500)

[Figure 14. Soil Moisture Complete Breadboard. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 41](#_Toc33010501)

[Figure 15. Air Quality/Humidity Breadboard Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 42](#_Toc33010502)

[Figure 16. Temperature Breadboard Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 43](#_Toc33010503)

[Figure 17. Soil Moisture Breadboard Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 43](#_Toc33010504)

[Figure 18. Air Quality/Humidity PCB Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 44](#_Toc33010505)

[Figure 19. Temperature PCB Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 44](#_Toc33010506)

[Figure 20. Soil Moisture PCB Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0. 45](#_Toc33010507)

# Introduction

With our project we aim to construct and develop a greenhouse monitoring system. Systems like this are already available to consumers and industry professionals yet most lack certain features that many people would like to see incorporated. With our project we aim to achieve a device that has all the features and specifications to benefit everyone. Users will also be able to have up to date information for key variables inside of a greenhouse environment. Some of these key values that will be interpreted; current temperature, current humidity, gas and air quality, and soil moisture levels. These values will then be added to a database hosted by Amazons’ Firestore, which we will then pick up inside of an Android mobile application that we are currently developing, were users will be able to view and interact with the greenhouse in real time. Things like watering the plants, opening vents, turning on fans, and lowering curtains will be some of the features that we also plan to incorporate into the finished project to allow remote management for the greenhouse. According to our project schedule, we have currently completed the first half of our project requirements by successfully completing our previous semester which has allowed us to move on to our current semester where we will continue working and later finish up with our project. This project is in collaboration with the Humber Arboretum, who is in need of an updated system to better closely monitor the environment and habitat of their plants.

## 1.1 Scope and Requirements

We are creating an Internet of Things (IoT) capstone project that uses a distributed computing model and is documented by an OACETT certification acceptable technical report. This project will consist of a Broadcom (Raspberry Pi 4) development platform with a custom PCB for connecting sensors, which will be encased in a custom enclosure. The Broadcom development platform will connect to the internet through enterprise wireless (capable of storing certificates). This device will be responsible for picking up/processing readings from the sensors and storing them within a database. The device will be capable of reading temperature from the DS18B20, air quality/humidity from the BME680, and soil moisture from the EK1940. The maximum dimensions for this project are 12 13/16" x 6" x 2 7/8" = 32.5cm x 15.25cm x 7.25cm. We will be adhering to CSA Z462, the highest AC voltage that will be used is 16Vrms from a wall adapter from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will not exceed 20 Watts.

This database will be hosted through Google’s Firebase and will be used to store the temperature, air quality/humidity, and soil moisture readings. These stored readings will be accessible for retrieval and display. It will also store login credentials of employees using the mobile application.

The mobile application will be an Android application designed for phones running Marshmallow 6.0 or higher. It will consist of a login screen for greenhouse employee authentication, and a guest login for all others. The application will be able to retrieve/display sensor data in real time, as well as have a refresh button to update readings. It will also have a weather widget displaying the outside weather local to the greenhouse. This application will be programmed in Java using Android Studio.

This project will be able to auto-maintain greenhouse conditions by comparing sensor readings to given parameters, and using built-in algorithms to determine which parts of the environment need adjusted.

This project will not measure certain readings such as sunlight and plant nutrient level. It will not feature its own outdoor weather station, but as stated above, it will rely on local weather data. This device is a prototype and therefore is not CSA approved.

Raspberry Pi 4

- CPU – Broadcom BCM2711, Quad core Cortex-A72 64-bit SoC @ 1.5GHz

- RAM – 4GB LPDDR4-2400 SDRAM

- WiFi – 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE

- Ethernet – Gigabit

- USB – 2 USB 3.0 ports; 2 USB 2.0 ports

- GPIO header – Raspberry Pi standard 40 pin

- HDMI – 2 × micro-HDMI ports

- Storage – Micro-SD card slot for loading operating system and data storage

- OS – Raspbian Buster (Debian Linux based)

DS18B20

- Measures Temperatures from -55°C to +125°C (-67°F to +257°F)

- ±0.5°C Accuracy from -10°C to +85°C

- Programmable Resolution from 9 Bits to 12 Bits

BME 680

- Interface I²C and SPI

- Pressure: 300 - 1100 hPa

- Humidity: 0 - 100%

- Temperature: -40 - 85°C

EK1940

- Interface PH2.54-3P

- Capacitive sensor

- Analog out

Custom PCB

- 2 layers

- 1.6mm thick

- lead-free soldering

- 1oz copper weight

- FR4 standard

- 16 mil min. trace spacing

Custom Enclosure

- 3D printed using PLA filament

- laser cut 3mm acrylic

Database

- Firebase Cloud Firestore

- NoSQL cloud based

- flexible, hierarchical data storage (documents and collections)

- persists data on device for offline use

Mobile Application

- native Android application

- minimum Marshmallow 6.0 (API 23) or higher

- phones and tablets (portrait mode)

- secure login for staff

# 2.0 Background

The real-life problem being solved by this project is Humber’s arboretum (Humber Arboretum, 2020) and several other nurseries lack a proper system to measure temperature, humidity, and soil moisture levels. This monitoring system will help them keep track of everything related to the health of the habitat inside the greenhouse. The device is capable of reading temperature, humidity and soil moisture to be used in plant nurseries. Along with a constructed smartphone mobile application that can be used to access a database to show users real-time information regarding temperature, humidity and soil moisture. The device is focused on solving these particular problems, will automate the greenhouse maintenance operations and monitor the growth conditions inside the greenhouse closely. (Humber Arboretum, 2020)

Humber Arboretum has a system currently installed by a company called Argus Controls (Argus Controls, 2020). The system consists of sensors/effectors, control panels, power panels, and connects to Argus servers. It can then be accessed/controlled locally from the client PC or remotely by Argus. Sensors monitor temperature, humidity, soil moisture, gas levels (CO and CO2), and a weather station monitors light, temperature, wind, rain, and snow outdoors. Effectors include air vents/fans, a mister, roof shades/curtains, evaporative cooling/heating pipes, and an irrigation system (not enabled). The system monitors all the sensors and uses the effectors to control the environment in the greenhouse. Most effectors rely on the readings of multiple sensors (i.e. curtains rely on light, temperature, and humidity). Argus mostly manages the system remotely. The system is lacking a functional irrigation system, nutrient/seeding system, different zones for different types of plants, and adjustable shade/lighting for certain plants.

Our device includes 3 sensors, a PCB board and a CPU (Raspberry Pi, 2020) that will connect to the app through Bluetooth and with multiple sensors connected to the greenhouse, we will be able to monitor all internal and external data and make any changes to the growing environment in the greenhouse in real-time. All of this data using firebase will be collected and mapped so we can control the outcome of a particular instance in the greenhouse. An example would be triggering irrigation when the solar level reaches a certain value and many more. With this type of flexibility, any greenhouse related data could be collected and controlled via automation. The greenhouse staff will be able to read and interact with that data directly through any secure Internet network and connecting their android smartphone through Bluetooth allowing them to view their greenhouse information with any android smartphone and have the capabilities needed to maintain and manually control the environment inside the greenhouse closely.

This proposal presents a plan for providing a solution for the arboretum at Humber College. This is an opportunity to combine the skills and knowledge that we’ve learned throughout our program and create a capstone project demonstrating our ability to create a greenhouse system that will improve the current system and provide the staff at Humber’s arboretum an easier more efficient solution to maintain the greenhouse from anywhere.

# Methodology

We are building our project based on our college course outlined to demonstrate our knowledge and understanding in key engineering concepts. We plan on solving the issues with current greenhouse monitoring systems that do not allow them to be very mobile in the sense that most actions must be given locally at the greenhouse typically at PC or control center located on site. With this we plan to incorporate a secure mobile application capable of reading key values in the habitat as well as being able to make remote changes to the environment as a user sees fit.

## 3.1 Required Resources

In this portion of the document we will be discussing the parts/components/materials we will be using, how we plan on building a custom PCB to incorporate all of our sensors and components as well as a case to house all the electronics, we will go over the tools and facilities we plan to utilize to complete our project, our plan to maximize efficiency and reduce cost in terms of shipping, duty and taxes on our items and lastly, we will discuss our working time over our lead time.

### 3.1.1 Parts, Components, Materials

With this project we plan to utilize plenty of different parts/components to add the needed functionality to the device to take in readings and make changes to the environment of the greenhouse as well as, different materials to construct things like the custom case we will build to enclose the entire project. One of the sensors we will be using is the BME680 (Bosch Sensortec, 2019) which will be responsible for taking in readings such as humidity and air quality (VOC (Volatile organic compound) gases). This sensor is capable of accurately measuring humidity with ±3% accuracy, barometric pressure with ±1 hPa absolute accuracy and temperature to ±1.0°C (Bosch Sensortec, 2019). This sensor will work in conjunction with the other sensors on the board to accurately update vital readings of the greenhouse to the database that will then be used to inform the users through the mobile application. Next, this project includes a Gikfun EK1940 capacitive soil moisture sensor. This sensor will be responsible for reading moisture levels in the soil of various plants. This sensor transmits data through analog signals. It features 3 wires: Vcc, Gnd, and Aout. The sensor has a built-in voltage regulator chip to support 3.3v - 5v. This sensor is very resistant to corrosion and has a long service life due to its use of capacitance. The EK1940 soil moisture sensor features an EK1856 diaphragm pump to go alongside it. This pump will be responsible for dosing a plant with water when its soil moisture value reads too low. This pump has a working voltage of 6v - 12v DC. It has a maximum flow rate of 1.5L/min. The inlet and outlet diameter of the pump are 5mm.

In order to connect/interface the EK1940 with the Broadcom platform, an analog-to-digital converter is required. The ADS1115 will be used due to its low cost, simple connection, and existing library base.

Other parts and materials used for connecting the EK1940 and EK1856 include: 1k resistor, PN2222A transistor, IN4001 diode, 1P3T switch, 2.1mm DC power jack, 3 pin molex connectors/headers, cone spray nozzle, ¼ “ barb to ⅜ “ NPT fitting, ¼ “ ID clear vinyl tube, electrical wire, and heat shrink.

Lastly, the 3rd main sensor we will be using is the DS18B20 sensor that will measure the temperature inside the greenhouse as well as an LED to indicate if the temperature exceeded the threshold.

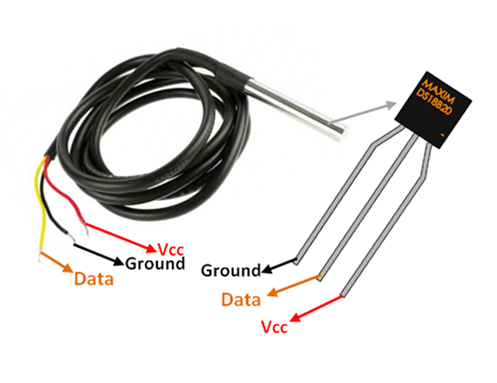


Figure 1 DS18B20 Temperature Sensor. This work is a derivative of: https://components101.com/sensors/ds18b20-temperature-sensor by components101.

The sensor works with the method of 1-Wire communication. It requires only the data pin connected to the microcontroller with a pull up resistor and the other two pins are used for power. It is a programmable digital temperature sensor with an operating range from -55°C to +125°C, it is accurate to ±0.5°C.

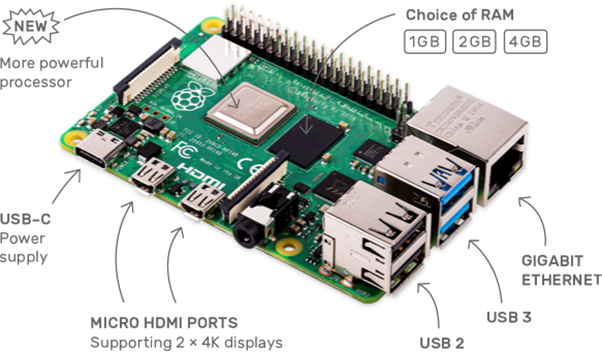
Raspberry Pi 4 CPU provides a wide range of new capabilities that will help the project meet all the requirements needed to complete the project.

Figure 2 Raspberry Pi 4 This work is a derivative of “https://www.raspberrypi.org/products/raspberry-pi-4-model-b/” by Raspberry Pi

Additionally, we plan to incorporate a fan system used to help regulate air quality and temperature inside the greenhouse as well as a shade system to control the amount of sunlight into the greenhouse. These systems will be used in conjunction with one another. For the fan/ventilation system we will use a small 6-12V PC fan as a simulation of a larger more industrial fan that would be used in a more real world application. As for the shade system we will make use of a servo motor to control the up and down motion of the shades. We plan to make use of various materials such as acrylic, steel L-channel ½ inch by ½ inch to make a small working scale greenhouse to demonstrate a working system in action later this semester.

### 3.1.2 Manufacturing

This PCB will be modified to include the BME680 air quality sensor and the EK1940 soil moisture sensor all on the same board. This will require the PCB to be scaled up from its smaller size in CENG 317. Doing so will ensure that all hardware fits on the board, maintains minimum trace spacing, and maintains spacing between connectors/plugs.

The PCB board pictured below (Figure 3) includes the DS18B20 temperature sensor which has three connectors; VDD goes to the 3V3 voltage pin provided by the Raspberry Pi 4 and GND is connected to the Pi’s ground pin. The DQ that provides the data is connected to a 4.7k resistor that’s wired to GIPO4 from one end and to 3V3 from the other end. The PCB board also includes an LED connected to a 220ohms resistor, which indicates that the sensor is working and collecting room temperature in Celsius.

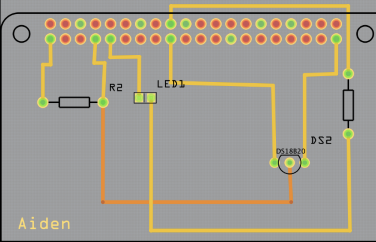


Figure 3 PCB design This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

The device includes an enclosure case to hold the Raspberry Pi and the PCB board. The case has openings for the USB ports, LAN cable, two micro HDMI ports, a display port, an audio port and finally the micro SD card slot. The top shows the project logo and the bottom has the four holes for the Raspberry Pi.

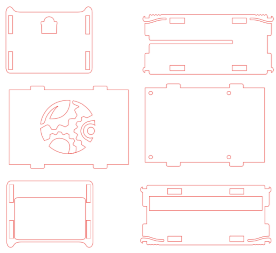


Figure 4 Enclosure case; This work is a derivative of "https://inkscape.org/" by inkscape.



Figure 5 Enclosure case; This image was taken by Aiden Bolos.

### 3.1.3 Tools and Facilities

We plan to use the most out of the resources given to us at Humber College, which will allow us to complete our project in a timely manner. Being students at Humber College we are given access to state of the art facilities and tools to help us advance our project as well as our knowledge and understanding for topics related to computer engineering technology. Tools such as soldering irons, de-soldering irons and a fume ventilation system will allow us to solder our own custom designed PCB that will incorporate all of our sensors and effectors. Other tools we will take advantage off at the college are the lab equipment such as power supplies and digital multi-meters to read accurate values for voltage and resistance when it comes to making our PCB. We are also able to send PCB files to the colleges’ prototype lab to have them make the custom PCB for us. The advantage to using the college for the printing of our PCB gives us the chance to resend new and updated files for new and revised boards if we feel like we need to make changes or if the pervious board didn’t work according to plan. This also gives us the relief that if we were to make a mistake will soldering, we can have another board printed to start fresh. By using a business outside the college, we would have to be sure that our design is what we wanted or that we weren’t going to make a mistake because that would lead to us spending more money for new boards and more time for waiting to receive the boards. The college also offers us the opportunity to use the resources in our prototype lab, things like a laser cutter that we will make use of to cut acrylic for pieces of our custom case. This will allow us to demonstrate our knowledge and understanding with the fundamentals of laser cutting. The prototype lab also allows us access to a 3D printing machine, we will use this machine to fabricate the main components for the custom case. The pieces we create from the laser cutter and 3D printing machine will allow us to fabricate a complete custom case to enclose our development platform, custom PCB and all of our sensors and effectors. Using the college for things like the laser cutter and 3D printing allows us to constantly update or make changes to our case when we need them allowing us to receive the new case within 1-2 days instead of outsourcing to a business outside the school where we would have to spend money and waste time to receive the case which wastes valuable resources in terms to the project. Using the college for most of the main features of the project allows us to operate mostly out of the college by using the resources given to us as students. These are the majority and main tools and facilities we will be using to undertake the preparation and the completion of or project.

### 3.1.4 Shipping, duty, taxes

We plan to optimize our shipping times, duty rates and tax expenses for this project to allow us to spend as little as possible all while still receiving items and materials in a timely and reasonable manor. Some steps we have taken to make sure we are optimizing shipping times are always making sure that websites we use for ordering have a guaranteed shipping date, sites like Amazon have a guaranteed 2 day shipping time for prime members and as students we are privileged with the benefit of getting 6 months free to Amazons’ prime services allowing us to reduce shipping times by a marginal amount. As for duty rates, we are sourcing our products and materials to locally owned businesses or businesses which operate out of Canada or the GTA (Greater Toronto Area). However, regarding the case where we may be required to go and pick up items from stores or mailboxes, we will be factoring in gas and transportation costs. This allows us to skip over things like duties or import fees on electronics or materials we may need saving us money and time when it comes to shipping as well. Lastly for taxes, since tax is a fixed rate on most items these days we can’t do much in terms of trying to combat the amount we are paying in taxes on products or materials however, when ordering items online we are making sure to utilize promotions or discount codes whenever possible to help us save that much more when it comes to the overall budget of the project. Optimizing these few things allows us to spend the least amount of money reducing our overall budget while still receiving our products and materials in a timely manner so we can stay up to date and on track with completing our project.

### 3.1.5 Time expenditure

The time required to complete this project is approximately 50 - 60 hours (continuous work), or 1 - 2 weeks. This varies depending on skill level. We will be dedicating approximately 6 hours per week as a team (2 hours each).

We are approaching this project with mild - moderate experience in electronics and circuits, tools and assembly, and computers. This time calculation is based on that level of experience.

Large amounts of time will go towards research and design of the circuitry/PCB, application programming, hardware programming, technical report writing, and testing/troubleshooting. Smaller amounts of time will go towards parts ordering, image installation, library installs, soldering, hardware assembly and case assembly.

Generally, 2 - 3 hours per week will go towards updating and maintaining the report, 2 - 4 hours will go towards the design/assembly task of the week, and 1 hour will go towards ordering, or planning/discussion.

We will be structuring our time in such a way that it is spent mostly as working time. With the resources we are using, the lead time for ordering components, or having parts made is generally very short (2 - 4 days max). However, lead times will be offset with other work to prevent down time. For example, we will order PCB in mid-February. While we are waiting for it to be made, we will be working on hardware coding and enclosure design. We will order the enclosure in early-March. While we are waiting for that to be made, we will be working on the mobile application and database.

## 3.2 Development Platform

We will be constructing our mobile application with Android Studio, which allows us to develop an application to work on most if not all Android devices. This application will be used to interact with our Broadcom development platform; Raspberry Pi 4. The application will allow the user to take control of specific features or aspects of the greenhouse as well as retrieve up to date and accurate information about the vitals of the greenhouse. These in conjunction with one another will be the backbone of our project.

### 3.2.1 Mobile Application

For the entire duration of semester 5 (September 2019 – December 2019) we have been working to develop an Android application that is capable of connecting with our development platform to allow for remote management of our greenhouse system. During that semester we were tasked with building a basic application that we could use as a starting point for our finished project. The application had to meet certain requirements during the semester that showed our knowledge and understanding in Android Studio and mobile application development. However, some of the requirements we had to meet for this class don’t make sense to keep in our finished application so we will be updating and removing some redundant aspects of the application over the course of this semester to better fit the needs of what we need our finished application to do. Things like, having the application allow users to access a database system hosted on Amazons Firestore database to retrieve up to date information on things like the current temperature inside the greenhouse, the humidity levels, the soil moisture levels of plants, and even the air quality inside the greenhouse to know if the plants are in a safe and optimal growing environment. This information is key and thus will be a main requirement of our finished application. As well as, being able to make remote changes to the environment which could mean, opening or closing a set of blinds to adjust the amount of light inside the greenhouse, turning on a fan or ventilation system to help regulate temperature and air quality, or even taking control of a watering system to manually water the plants as a user sees fit. These features are currently not implemented into the application but will be over the next few months as the project develops further. Right now, we are set at a good spot for our application as we can do most things that we had set out to achieve the semester before. Users are able to login using credentials that are stored securely in a data base(Figure 6), users can access the values that we have stored in the database representing temperature, humidity, and soil moisture levels (Figure 7), which means we have an active connection to our database, we even have a page setup where the users will be able to make the adjustments to the environment remotely(Figure 8). As stated, before this feature is currently not deployed in the application but will be available in future development of the mobile application. As our application stands right now, all activities and features are subject to removal, updates or redesigns as this is just our current working version and where we stand in regards to the development of our mobile application. As time moves on we plan on ironing out bugs, implementing all of our discussed features, removing redundant information or screen clutter, updating UI’s (User Interfaces) for a more appealing look and feel, and finishing up our application with a fully developed, up to date and secure mobile environment to accompany our finished project.

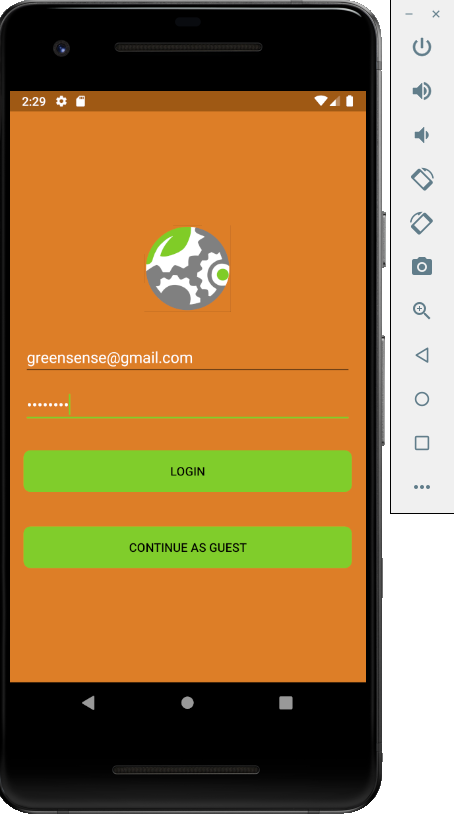


Figure 6 Android Studio. (01,2020). Screenshot from login page of GreenSense mobile application. Screenshot by Ryan McAdie

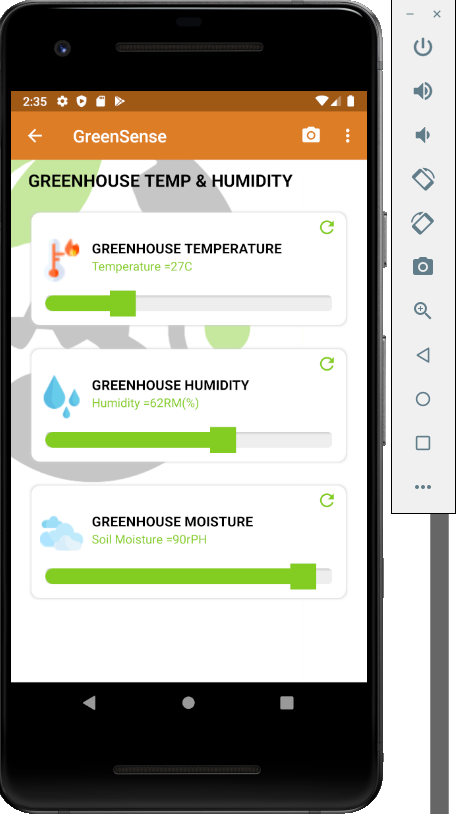


Figure 7 Android Studio. (01,2020). Screenshot from values page of GreenSense mobile application. Screenshot by Ryan McAdie

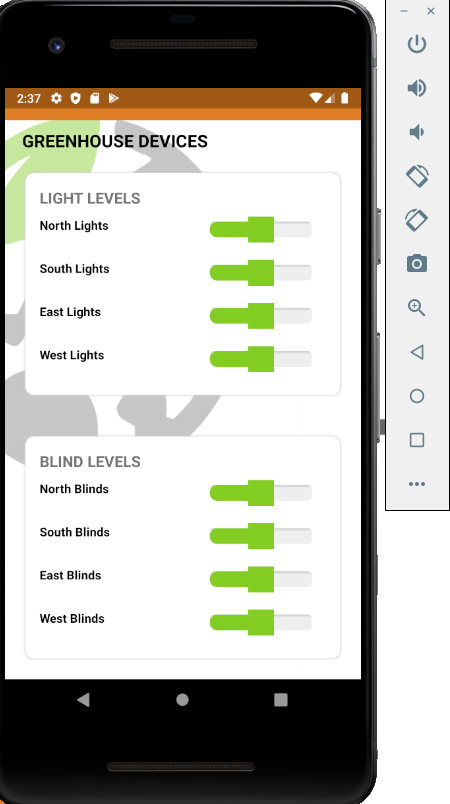


Figure 8 Android Studio. (01,2020). Screenshot from devices page of GreenSense mobile application. Screenshot by Ryan McAdie

### 3.2.2 Image/firmware

In the previous semester we had to decide on which CPU we could use to allow us to remotely manage our greenhouse system. We decided to go with the Raspberry Pi as our CPU to control the sensors remotely. First of all, we installed raspbian on the SD card as a ZIP file using NOOBS software. We got the image from raspberry pi’s download page from their official website, extracted the file using 7-Zip on windows, after that we used the SD Formatter software to format the SD card. After checking that all the files have been deleted from the SD card and it is completely empty, we navigate back to the folder that contains the NOOBS files we downloaded earlier. Selected all the files in the noobs directory and then dragged and dropped the files into the SD card drive which took around 20 minutes to transfer the files. Once complete, we removed the SD card from the computer and placed it into the SD card slot in the Raspberry Pi and then connected the Raspberry Pi to a monitor, keyboard, mouse and finally power. Connecting to the power booted the Raspberry Pi and we were presented with a selection of operating systems to install. We selected raspbian as it was the recommended operating system to install. As this was the first time we booted raspbian, the raspbian config menu appeared where we can set our location details in order to get the correct time and date on the Raspberry Pi. After setting up the Raspberry Pi we configured the Wi-Fi settings in order to get the Raspberry Pi to automatically connect to the internet remotely and control the interface of the Raspberry Pi without the mouse and keyboard using our personal devices. As it is not convenient to work directly on the Raspberry Pi we used VNC (VNC, 2020) stands for Virtual Network Computing which works as a desktop sharing system that allowed us to remotely control the raspberry Pi running VNC server from our computer or mobile device running VNC viewer. VNC viewer transmits the mouse and keyboard touch to VNC server which updates to the user’s screen. After creating an account with VNC and downloading VNC viewer and server on the appropriate devices, we had to setup a password for the connection to prevent unauthorized remote access to the device.

We used python code for all three sensors to measure temperature, humidity and moisture levels. The code used to output the temperature is straightforward. It opens the file w1\_slave, reads the file and prints out the temperature readings coming from the sensors in Celsius and Fahrenheit.

### 3.2.3 Breadboard/Independent PCBs

In the previous course, CENG 317, each group member designed a circuit for their sensor to interface with a Raspberry Pi. Ryan designed a circuit for the BME680 to read air quality, humidity, temperature, and altitude. Aiden design a circuit for the DS18B20 to read precise temperature. Aiden’s circuit also included a warning light when the temperature crossed a certain threshold. Daniel designed a circuit for the EK1940 to read soil moisture levels. In addition, Daniel’s circuit included a motor/pump system for supplying water to a plant. Each group member used previous course knowledge and online resources (datasheets, examples, diagrams) to design their circuit. Circuit schematics were drawn and edited using Fritzing.

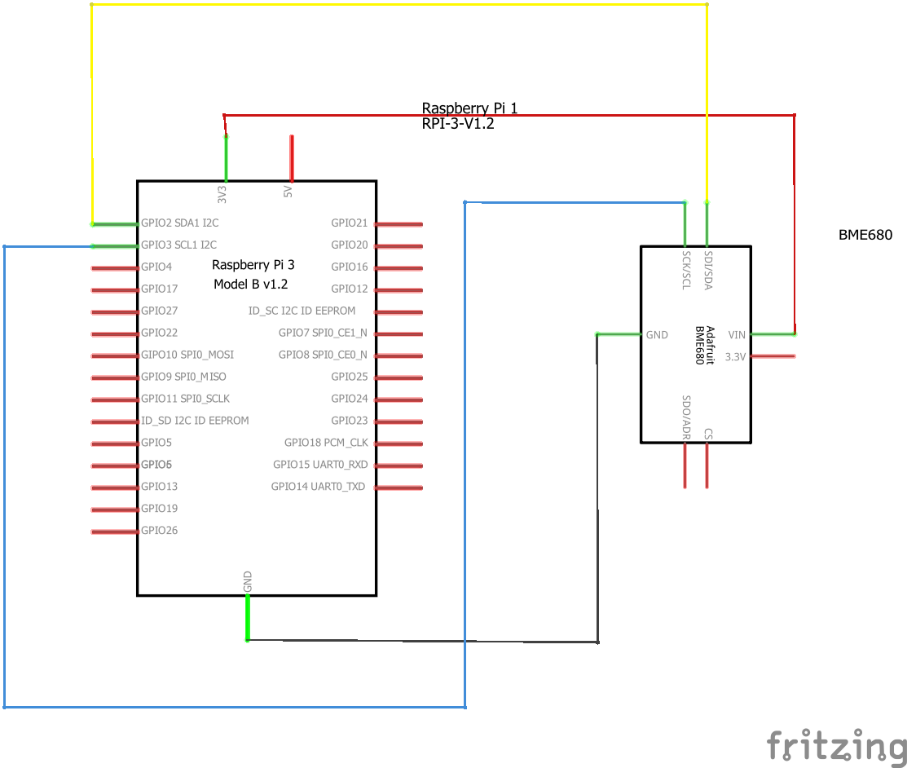


Figure 9. Air Quality/Humidity Initial schematic. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

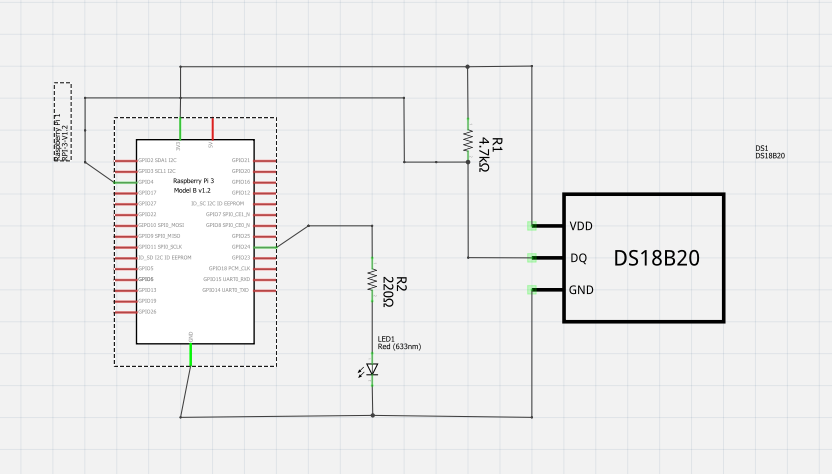


Figure 10. Temperature Initial schematic. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

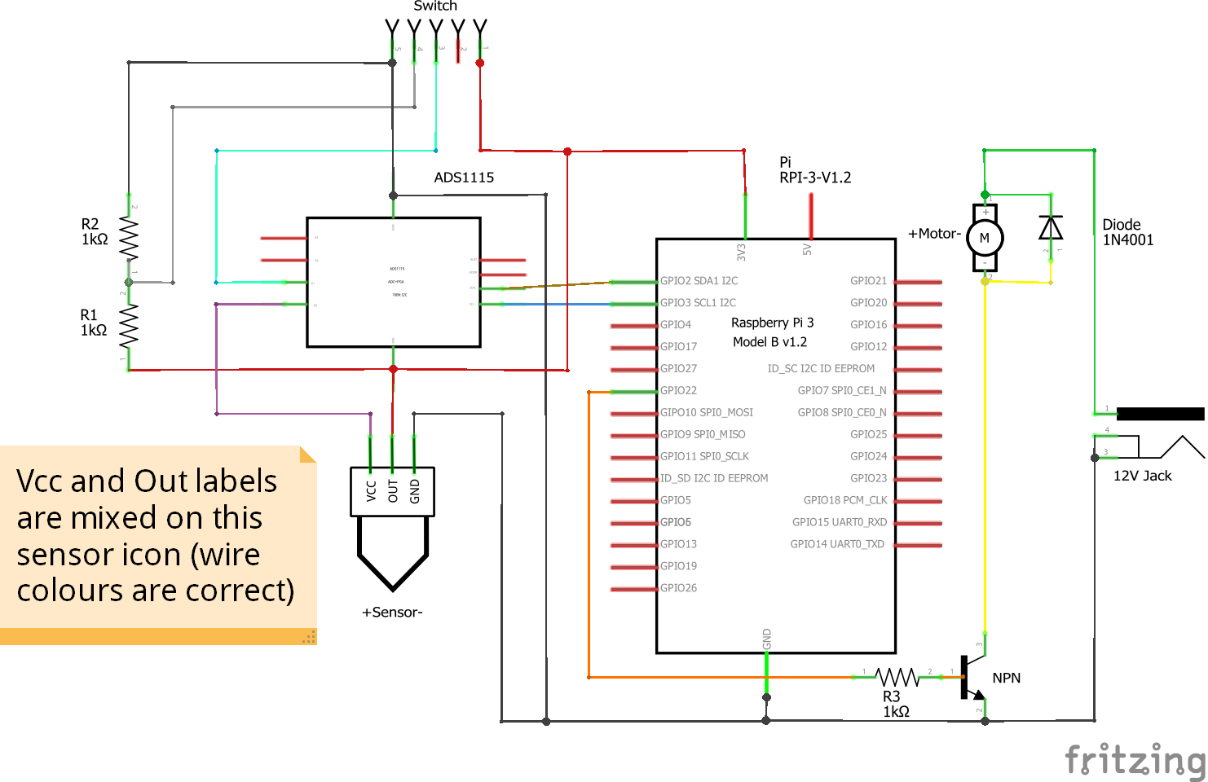


Figure 11. Soil Moisture Initial schematic. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

After the schematics were created and checked for correctness, prototype circuits were built using breadboarding. Each group member installed their components onto a breadboard and connected the components via jumper wires. These connected components/subsystems were then connected to the Raspberry Pi pins using jumper wires. Before powering on, connections were double-checked and tested with a multimeter. This was done as a precaution to prevent potential damage to the Pi or the components. The Raspberry Pi was then powered on and sample code was used to test the circuit for connectivity and proper functionality.

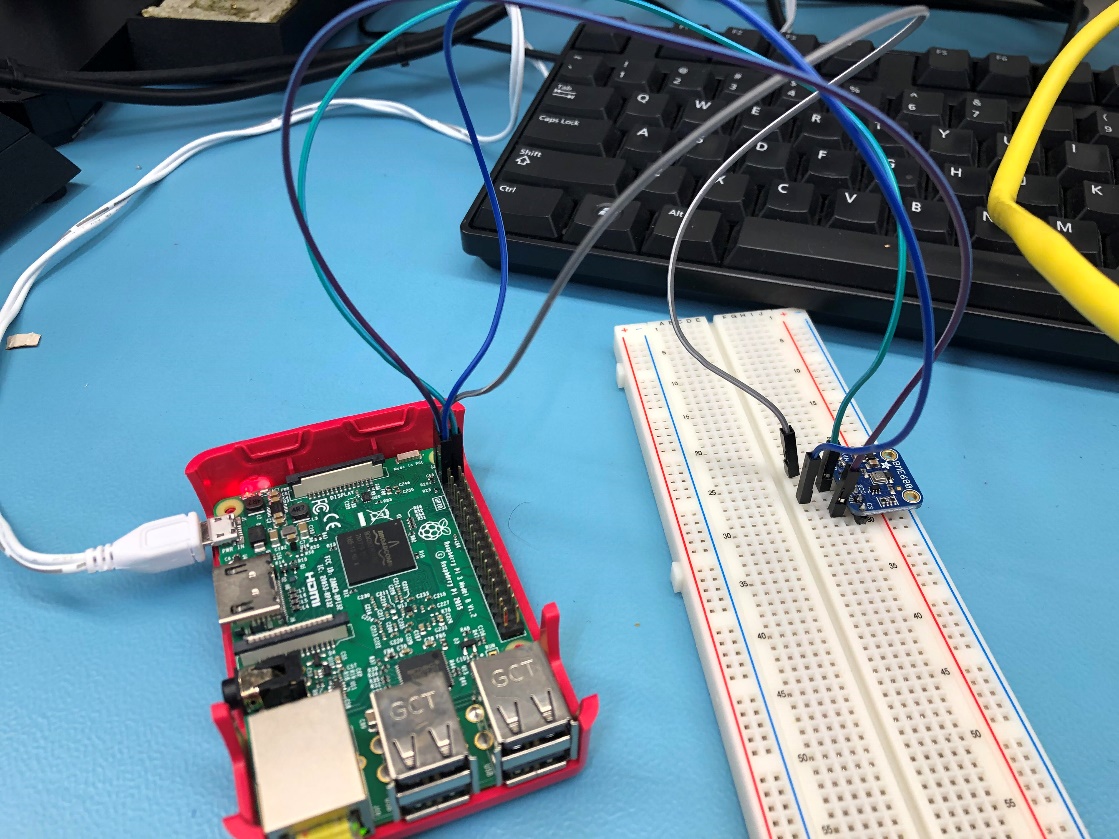


Figure 12. Air Quality/Humidity Complete Breadboard. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

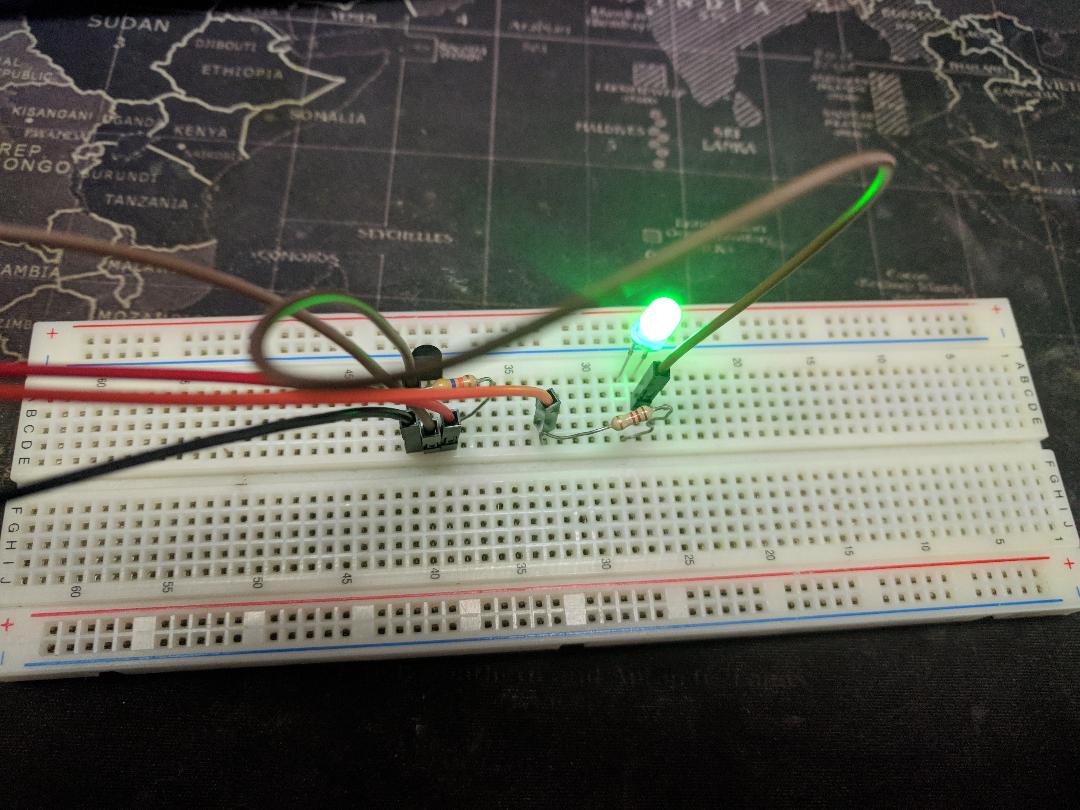


Figure 13. Temperature Complete Breadboard. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

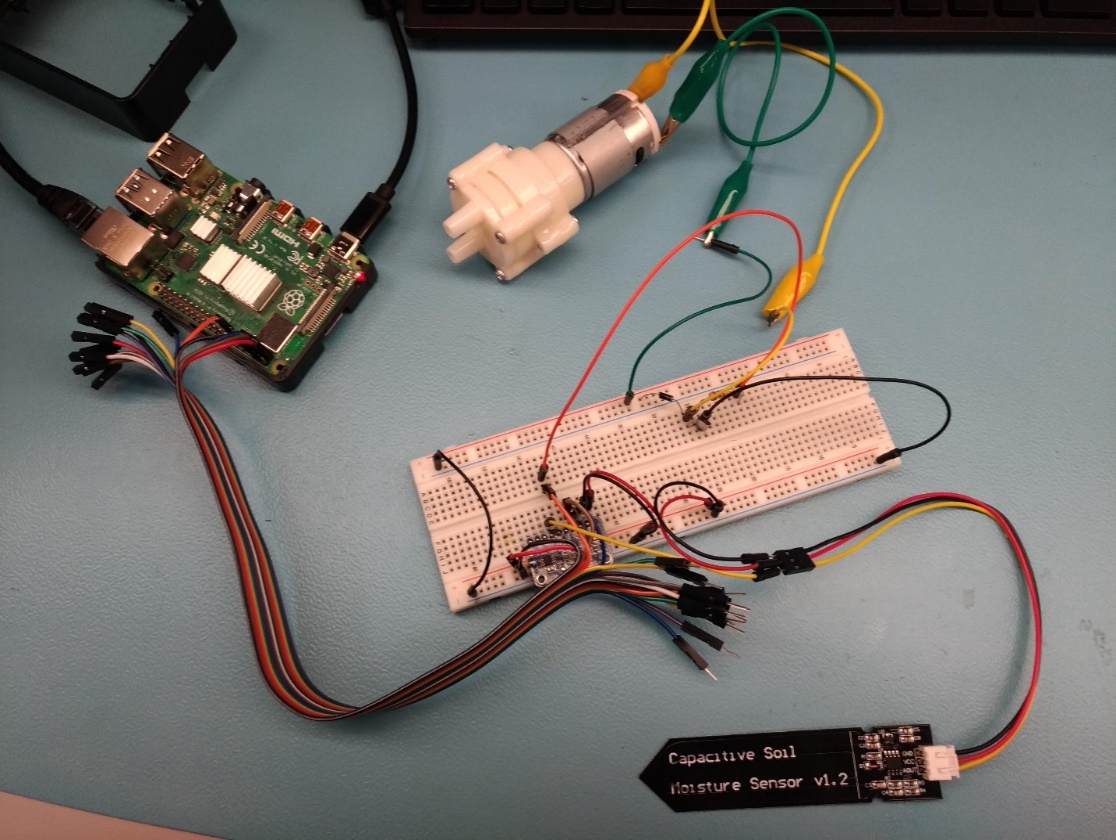


Figure 14. Soil Moisture Complete Breadboard. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

Circuit designs were finalized once functionality was verified through repeated tests. The breadboard and Raspberry Pi connections were then drawn in Fritzing.

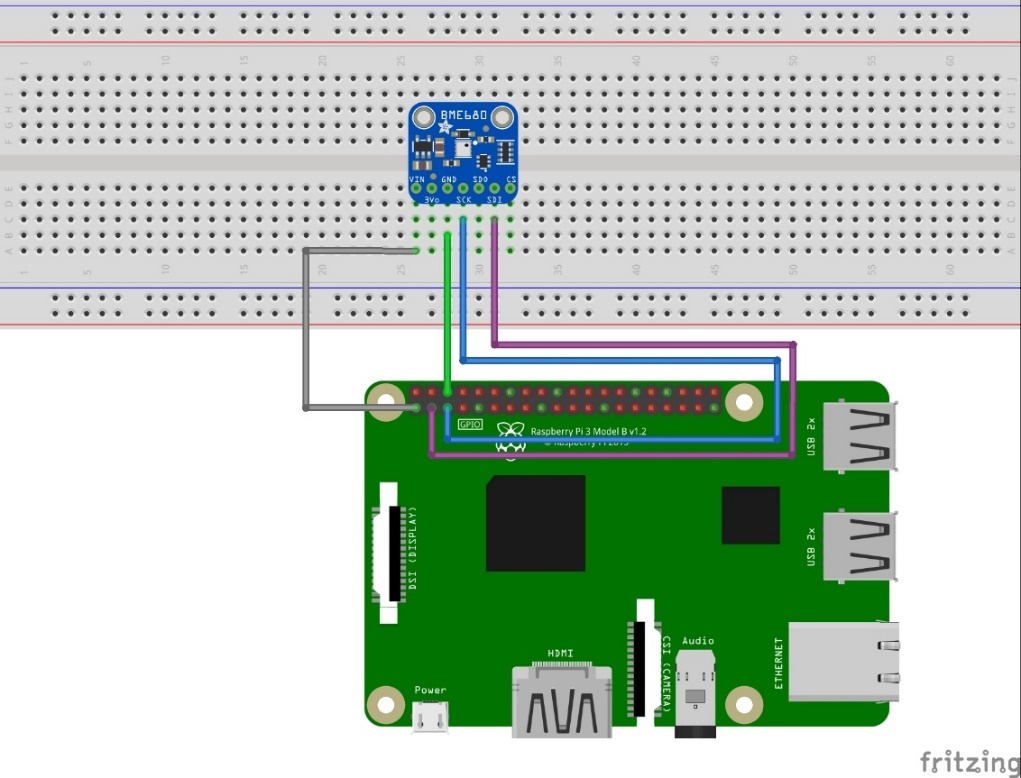


Figure 15. Air Quality/Humidity Breadboard Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

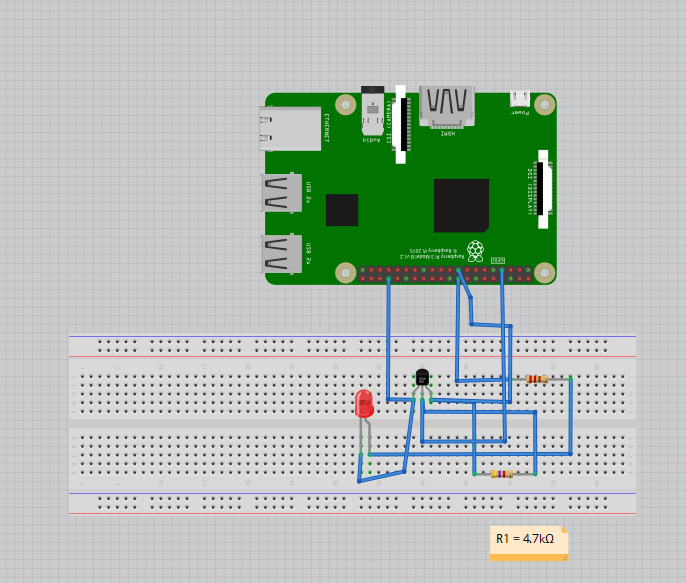


Figure 16. Temperature Breadboard Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

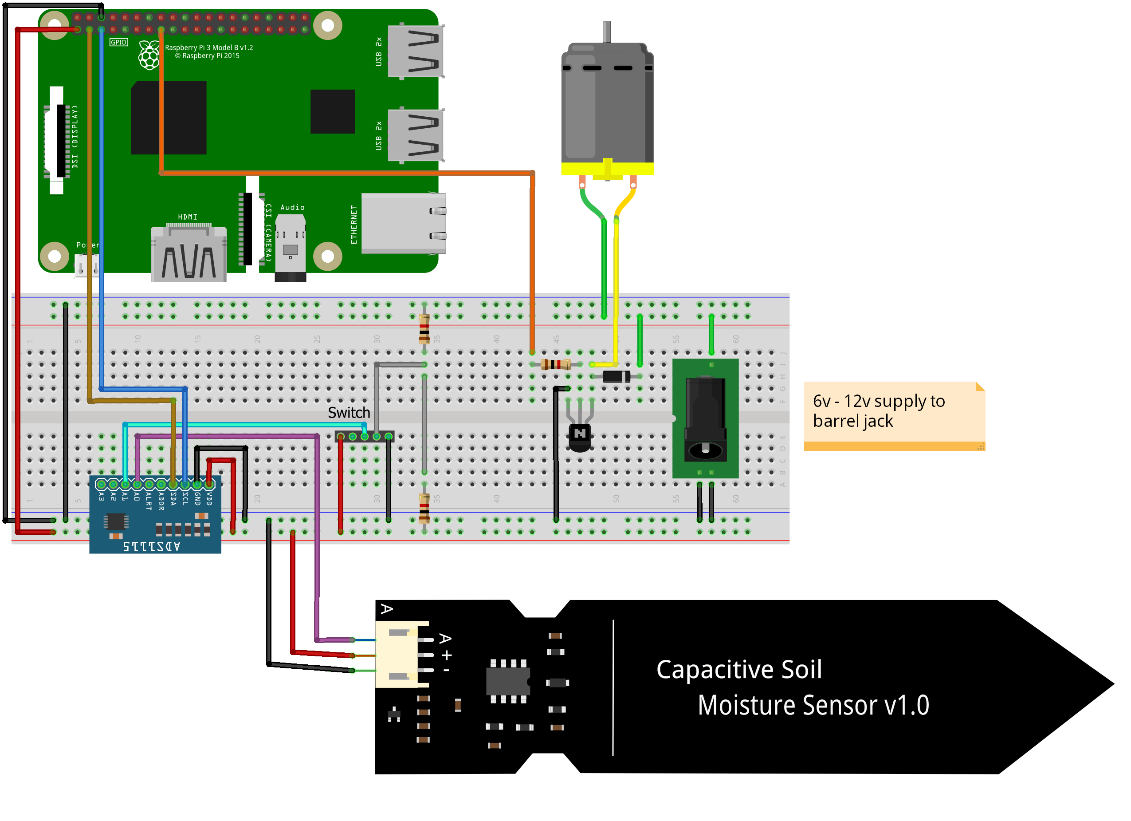


Figure 17. Soil Moisture Breadboard Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

At this point, the circuit needed to be transferred from the breadboard to a more permanent/production-level solution. The answer to this was to design a custom PCB. Each group member would design and edit a custom PCB using Fritzing. This was done by selecting board dimensions, drawing traces, inserting vias, and placing holes for headers/components.

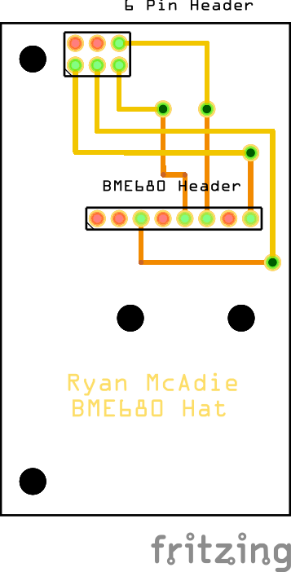


Figure 18. Air Quality/Humidity PCB Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

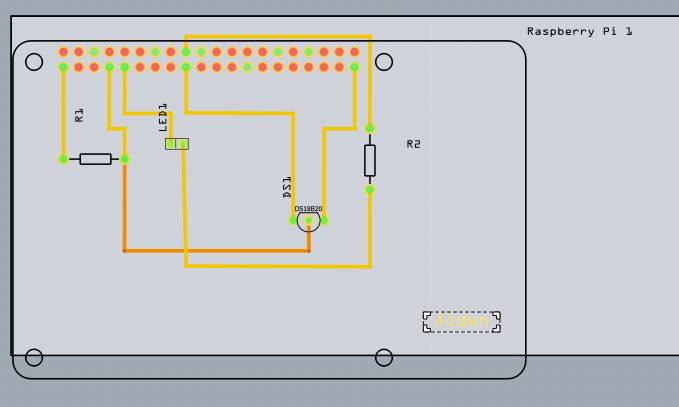


Figure 19. Temperature PCB Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

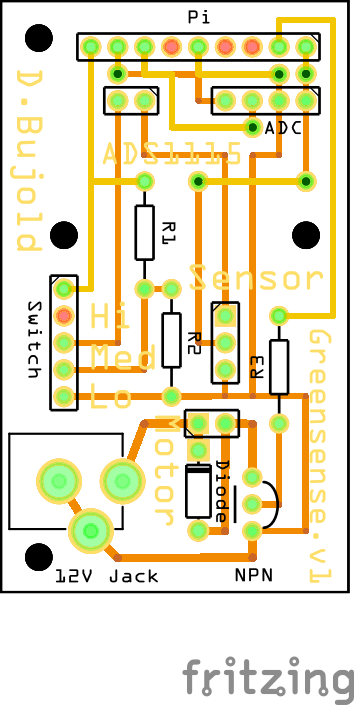


Figure 20. Soil Moisture PCB Design. This work is a derivative of "http://fritzing.org/parts/" by Fritzing, used under CC:BY-SA 3.0.

Once the PCB was designed and approved by the Fritzing software, the design was exported as a Gerber file. This file was then sent to the Humber Prototype Lab for the PCB to be made. The Prototype Lab uses the LPKF ProtoLaser ST and the LPKF ProtoMat S103 to create these custom PCBs. Following a 2-3 day lead time the PCB would is ready for pickup. After receiving the PCB, we fitted and soldered all the necessary components in their respective places.

With the PCB fully assembled, we double-checked all solder joints and connections, as well as tested the circuit with a multimeter. When everything checked out, we installed the PCB onto the Raspberry Pi for the power up test. Just as we did with the breadboarding, the Pi was turned on and sample code was used to test for functionality. We continued by modifying the code with repeating tests until the desired behaviour was achieved. Ryan tested air quality, humidity, and temperature readings by exposing the unit to different environments. He also tested altitude by moving the unit up and down. Aiden tested temperature by exposing his sensor to different temperatures, and tested the warning light by exposing his sensor to heat. Daniel tested soil moisture by placing his sensor in various soils and pure water.

Now that this process has been completed, we have three individual fully functional sensors/PCBs. Our end goal is to join our three subsystems together as one complete system. The next step is to breadboard all of our sensors together with the Raspberry Pi to ensure group functionality. We will be doing this during the week of February 17th. In addition, Ryan will be adding a fan/shutter effector to complement his air quality readings, and Aiden will be adding an overhead shade effector to complement his temperature readings. Daniel has already implemented his soil watering effector. These extra systems are currently being designed, and schematics are being drawn in Fritzing. Once the circuit designs for these effectors are completed, we will breadboard them alongside the three sensors. This way we will be able to ensure the complete system is functional as a unit before proceeding to the next step.

After we have breadboarded the complete system, checked it over, and are confident with its performance; we will design a custom PCB for all of these systems. This PCB will provide a means of interfacing the three sensors and three effectors with the Raspberry Pi and a separate power circuit. If we stay on schedule (which we currently are) the PCB design in Fritzing will be completed and sent off to be made by the week of February 24th.

### 3.2.4 Printed Circuit Board

Demo

/1 Hardware present?

/1 PCB Complete and correct

/1 PCB Soldered wire visible but trim, no holes or vacancies

/1 PCB Tested with multimeter

/1 PCB Powered up

How did you build your Prototype: PCB?

### 3.2.5 Enclosure

Demo

/1 Hardware present?

/1 Case encloses development platform and custom PCB.

/1 Appropriate parts securely attached.

/1 Appropriate parts accessible.

/1 Design file in repository, photo in report.

How did you build your Prototype: Case?

## 3.3 Integration

Demo

/1 Hardware present?

/1 Data sent by hardware

/1 Data retrieved by mobile application

/1 Action initiated by mobile application

/1 Action recieved by hardware

Report

/1 Enterprise wireless connectivity (250)

/1 Database configuration (250 words)

/1 Security considerations (500 words)

/1 Unit testing (900 words)

/1 Production testing (100 words)

### 3.3.1 Enterprise Wireless Connectivity

How did you make a Database accessible by both your Prototype and Mobile Application?

### 3.3.2 Database Configuration

### 3.3.3 Security

### 3.3.4 Testing

Unit testing and Production testing.

# 4.0 Results and Discussions

Is your prototype perfect? What did you learn?

# 5.0 Conclusions

If you were making 1000 of these.

Report

/1 Hardware present?

/1 Checklist truthful

/1 Valid Comments

/1 Results and Discussion (500 words)

/1 Conclusion

# 6.0 References

*Argus Controls*. (2020). Retrieved from Argus is a member of the Conviron Group of Companies : https://www.arguscontrols.com/

Bosch Sensortec. (2019, July). *BME680 - Datasheet.* Retrieved from Robert Bosch GmbH: https://ae-bst.resource.bosch.com/media/\_tech/media/datasheets/BST-BME680-DS001.pdf

*Humber Arboretum*. (2020). Retrieved from Humber Arboretum and centre for urban ecology: https://humber.ca/arboretum/

*Raspberry Pi*. (2020). Retrieved from Raspberry Pi 4 Tech Specs: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

VNC. (2020). Retrieved from realvnc: https://www.realvnc.com/en/connect/download/viewer/

# 7.0 Appendix

## 7.1 Firmware code

import time

import board

from busio import I2C

import adafruit\_bme680

import RPi.GPIO as GPIO

import Adafruit\_ADS1x15

import os

import glob

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

# Create an ADS1115 ADC (16-bit) instance.

adc = Adafruit\_ADS1x15.ADS1115()

GAIN = 1

# Create library object using our Bus I2C port

i2c = I2C(board.SCL, board.SDA)

bme680 = adafruit\_bme680.Adafruit\_BME680\_I2C(i2c, debug=False)

# change this to match the location's pressure (hPa) at sea level

bme680.sea\_level\_pressure = 1013.25

os.system('modprobe w1-gpio')

os.system('modprobe w1-therm')

base\_dir = '/sys/bus/w1/devices/'

device\_folder = glob.glob(base\_dir + '28\*')[0]

device\_file = device\_folder + '/w1\_slave'

def read\_temp\_raw():

f = open(device\_file, 'r')

lines = f.readlines()

f.close()

return lines

def read\_temp():

lines = read\_temp\_raw()

while lines[0].strip()[-3:] != 'YES':

time.sleep(0.2)

lines = read\_temp\_raw()

equals\_pos = lines[1].find('t=')

if equals\_pos != -1:

temp\_string = lines[1][equals\_pos+2:]

temp\_c = float(temp\_string) / 1000.0

temp\_f = temp\_c \* 9.0 / 5.0 + 32.0

return temp\_c, temp\_f

while True:

print("Gas: %d ohm" % bme680.gas)

print("Humidity: %0.1f %%" % bme680.humidity)

print("Soil = %d" %adc.read\_adc(0, gain=GAIN))

print("Temp in C,F:")

print(read\_temp())

print("")

time.sleep(5)

## 7.2 Application code

Code for retrieving information from the database:

greenhouse1\_frag.java

private void getTempStat() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 int temp = ((Long) Objects.*requireNonNull*(document.get("Temp"))).intValue();  
 if(temp >= 26 && temp <= 30){  
 tvTempStat.setText("GREAT");  
 tvTempStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*green*));  
 }else if(temp >= 21 && temp <= 25){  
 tvTempStat.setText("OKAY");  
 tvTempStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*yellow*));  
 }else if(temp <= 20){  
 tvTempStat.setText("NEEDS ATTENTION (TOO LOW)");  
 tvTempStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*red*));  
 }else if (temp > 32){  
 tvTempStat.setText("NEEDS ATTENTION (TOO HIGH)");  
 tvTempStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*red*));  
 }  
 }  
 } else {  
 tvTempStat.setText(getString(R.string.*docErr*) + task.getException());  
 }  
 });  
}  
private void getHumStat() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 int hum = ((Long) Objects.*requireNonNull*(document.get("Humidity"))).intValue();  
 if(hum >= 50 && hum <= 70){  
 tvHumStat.setText("GREAT");  
 tvHumStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*green*));  
 }else if(hum >= 71 && hum <= 80){  
 tvHumStat.setText("OKAY");  
 tvHumStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*yellow*));  
 }else if(hum <= 49){  
 tvHumStat.setText("NEEDS ATTENTION (TOO LOW)");  
 tvHumStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*red*));  
 }else if (hum > 81){  
 tvHumStat.setText("NEEDS ATTENTION (TOO HIGH)");  
 tvHumStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*red*));  
 }  
 }  
 } else {  
 tvHumStat.setText(getString(R.string.*docErr*) + task.getException());  
 }  
 });  
}  
private void getWaterStat() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 int soil = ((Long) Objects.*requireNonNull*(document.get("Soil"))).intValue();  
 if(soil >= 88 && soil <= 100){  
 tvWaterStat.setText("GREAT");  
 tvWaterStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*green*));  
 }else if(soil >= 70 && soil <= 87){  
 tvWaterStat.setText("OKAY");  
 tvWaterStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*yellow*));  
 }else if(soil <= 69){  
 tvWaterStat.setText("NEEDS ATTENTION (TOO LOW)");  
 tvWaterStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*red*));  
 }else if (soil > 101){  
 tvWaterStat.setText("NEEDS ATTENTION (TOO HIGH)");  
 tvWaterStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*red*));  
 }  
 }  
 } else {  
 tvWaterStat.setText(getString(R.string.*docErr*) + task.getException());  
 }  
 });  
}  
private void getGasStat() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 int airq = ((Long) Objects.*requireNonNull*(document.get("AirQ"))).intValue();  
 if(airq >= 0 && airq <= 50){  
 tvGasStat.setText("GREAT");  
 tvGasStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*green*));  
 }else if(airq >= 51 && airq <= 100){  
 tvGasStat.setText("OKAY");  
 tvGasStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*yellow*));  
 }else if(airq <= 101){  
 tvGasStat.setText("NEEDS ATTENTION (UNHEALTHY ENVIRONMENT)");  
 tvGasStat.setTextColor(ContextCompat.*getColor*(getContext(), R.color.*red*));  
 }  
 }  
 } else {  
 tvGasStat.setText(getString(R.string.*docErr*) + task.getException());  
 }  
 });  
}

TempAndHumidityActivity.java

private void SetRandomTempValue() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 tvGreenHouseTemp.setText(getString(R.string.*tempEq*) + document.get("Temp") + getString(R.string.*cels*));  
 int temp = ((Long) Objects.*requireNonNull*(document.get("Temp"))).intValue();  
 sbTemp.setProgress(temp);  
 }  
 } else {  
 tvGreenHouseTemp.setText(getString(R.string.*docErr*) + task.getException());  
 }  
 });  
}

private void SetRandomHumidityValue() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 tvGreenHouseHumidity.setText(getString(R.string.*humEq*)+ document.get("Humidity") + getString(R.string.*humVal*));  
 int hum = ((Long) Objects.*requireNonNull*(document.get("Humidity"))).intValue();  
 sbHumidity.setProgress(hum);  
 }  
 } else {  
 tvGreenHouseHumidity.setText(getString(R.string.*docErr*) + task.getException());  
 }  
 });  
}

private void SetMoistureLevelValue() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 tvMoistureLevel.setText(getString(R.string.*soilmoisEq*) + document.get("Soil") + getString(R.string.*moisVal*));  
 int soil = ((Long) Objects.*requireNonNull*(document.get("Soil"))).intValue();  
 spMoistureLevel.setProgress(soil);  
 }  
 } else {  
 tvMoistureLevel.setText(getString(R.string.*docErr*)+ task.getException());  
 }  
 });  
}

private void SetGasValue() {  
 db.collection("Readings").document("Values").collection("Data")  
 .get()  
 .addOnCompleteListener(task -> {  
 if (task.isSuccessful()) {  
 for (QueryDocumentSnapshot document : Objects.*requireNonNull*(task.getResult())) {  
 tvGasLevel.setText("Air Quality =" + document.get("AirQ") + "AQI");  
 int airq = ((Long) Objects.*requireNonNull*(document.get("AirQ"))).intValue();  
 airq = airq / 5;  
 sbGasLevel.setProgress(airq);  
 }  
 } else {  
 tvGasLevel.setText(getString(R.string.*docErr*) + task.getException());  
 }  
 });  
}

Demo

/1 Hardware present?

/1 Memo by student A

/1 Login activity

/1 Data visualization activity

/1 Action control activity

Report

/1 Login activity

/1 Data visualization activity

/1 Action control activity

/1 Modified Code Files in Appendix

/1 Link to Complete Code in Repository