Technical Report

for

Greenhouse Monitoring System

Version 1.0 approved

Prepared by Colin Blakley, Kenneth Chen and Princess Hernandez

Humber College – School of Applied Technology

April 25, 2019

# Declaration of Joint Authorship

We, Colin Blakley, Kenneth Chen and Princess Hernandez, confirm that this work submitted for assessment is our own and is expressed in our own words. Any uses made within it of the works of any other author, in any form (core concepts, diagrams and figures, previous technologies, programs and source code), are properly acknowledged at the point of use. A list of the references used is included. Colin handled the developing of the GUI and linking with the database. Kenneth handled the developing of the mobile application and the database. Princess managed integrating all hardware components.

|  |  |  |
| --- | --- | --- |
| Co-authors’ signatures | | |
| Date | Name | Signature |
| April 25,2019 | Colin Blakley |  |
| April 25,2019 | Kenneth Chen |  |
| April 25,2019 | Princess Hernandez |  |

# 

# Approved Proposal

January 17, 2019

***Proposal for the development of Greenhouse Monitoring System***

Prepared by Colin Blakley, Kenneth Chen, and Princess Hernandez  
*Computer Engineering Technology Students*

https://github.com/PrincessHernandez/GreenhouseMonitoringSystem

**Executive Summary**

As a student in the Computer Engineering Technology program, I will be integrating the knowledge and skills I have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the following sensors and actuators AM2315 Humidity/Temp (0x5C), CCS811 VOC (0x5B), SSD1306 OLED (0x3C). The database will store temperature, humidity and gas level data retrieved from the device. The mobile device functionality will include a series readings of temperature and humidity, as well as a variety of organic compound levels found in the air. There will be warnings when the data retrieved is below or above acceptable ranges. and will be further detailed in the mobile application proposal. I will be collaborating with the following company/department Colin Blakley, Kenneth Chen, and Princess Hernandez. In the winter semester I plan to form a group with the following students, who are also building similar hardware this term and working on the mobile application with me. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 2 or 3 student group.

**Background**

The problem solved by this project is temperature, humidity and carbon dioxide (CO2) levels are the key factors in monitoring and manage the growth process of plants. Plants become more prone to diseases when temperature is too low or deteriorate when it is too high. Low relative humidity can attract pests like red spider mites because of little to no moisture in plants. The temperature and humidity maintenance play a big impact in contributing to the generation carbon dioxide level. Carbon dioxide is essential for plant growth as it supplies the nutrients.. A bit of background about this topic is greenhouses are closed environments where conditions are optimized for plant growth. Ideally, in greenhouses, temperature should be between 10 - 20 degrees Celsius, 85% - 95% relative humidity and 300 - 500 ppm CO2 level. Measuring and maintaining an ideal temperature, humidity and carbon dioxide (CO2) in greenhouses decreases the dependency on pesticides. Farmers can rely on real-time data of temperature, humidity and CO2 to determine wether their agriculture is at risk. They are able to use the monitoring system to encourage themselves to increase yields. This is an important need for framers in a world where consumers demand more and corporations demand increased profits and cannot afford substantial loses.These sensors will become a staple in the framer community where greenhouse and any structure that need to have a maintained climate.

Existing products on the market include (Jones, 2013). I have searched for prior art via Humber’s IEEE subscription selecting “My Subscribed Content” (IEEE, 2015) and have found and read (Dan, Yang, Jianqiu, 2016) which provides insight into similar efforts.

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

* Java Docs from CENG 212 Programming Techniques In Java,
* Construction of circuits from CENG 215 Digital And Interfacing Systems,
* Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
* Micro computing from CENG 252 Embedded Systems,
* SQL from CENG 254 Database With Java,
* Web access of databases from CENG 256 Internet Scripting; and,
* Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

**Methodology**

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project:  
 Phase 1 Hardware build.  
 Phase 2 System integration.  
 Phase 3 Demonstration to future employers.

*Phase 1 Hardware build*

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

*Phase 2 System integration*

The system integration will be completed in the fall term.

*Phase 3 Demonstration to future employers*

This project will showcase the knowledge and skills that I have learned to potential employers.

The brief description below provides rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

OLED screen from Ebay, USB powered fan from Amazon

**Concluding remarks**

This proposal presents a plan for providing an IoT solution for the atmospheric factors that can be controlled in greenhouses are temperature, humidity and CO2 levels. High humidity and low temperature allows for plants to grow at an ideal rate. The temperature and humidity sensor will gather data to help manage a greenhouse. If the temperatures were to go out of range the yield of the plants could be effected in a negative way, and in a worst case scenario - kill the plants. By tracking and monitoring the data, adjustments can be made to the greenhouse to ensure maximum yield. Therefore, using humidifiers and circulation fans helps maintain an ideal greenhouse environment where plants can freely grow at the best rate. Better control of temperature and humidity can regulate precise CO2 levels and create a secure environment for growing plants.. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by (Dan, Yang, Jianqiu, 2016). I request approval of this project.

**References**

Jones, J.B. (2013, December 1). *Maintaining Control in the Greenhouse*. Retrieved from

https://www.maximumyield.com/maintaining-control-in-the-greenhouse/2/949

Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library

[Online]. Available: https://ieeexplore.ieee.org/search/advsearch.jsp

Dan, L., Jianmei, S., Yang, Y., & Jianqiu, X. (2016). Precise Agricultural Greenhouses Based on

the IoT and Fuzzy Control. 2016 International Conference on Intelligent Transportation,

Big Data & Smart City (ICITBS). doi:10.1109/icitbs.2016.19

# Abstract

Greenhouses has been an environmental solution to grow crops without involving any form of contaminants. However, due to atmospheric factors it is difficult to maintain plant growth. We have conducted research on integration of 3 different sensors that will carry out a solution that will maintain the growth of plants by monitoring atmospheric factors.

The following technical report will be on the software and hardware requirements for our project Greenhouse Monitoring System. It consists of 3 sensors that will be stationed in a greenhouse that works with our mobile application “Greenhouse Monitor.” This report will provide the details of the device we are going to develop by including features of our system and diagrams for the application.

Table of Contents

[Declaration of Joint Authorship iii](#_Toc6924877)

[Approved Proposal v](#_Toc6924878)

[Abstract xi](#_Toc6924879)

[Revision History xiv](#_Toc6924880)

[List of Illustrations xv](#_Toc6924881)

[1. Introduction 1](#_Toc6924882)

[1.1 Purpose 1](#_Toc6924883)

[1.2 Document Conventions 2](#_Toc6924884)

[1.3 Intended Audience and Reading Suggestions 2](#_Toc6924885)

[1.3 Product Scope 2](#_Toc6924886)

[2. Overall Description 3](#_Toc6924887)

[2.1 Problem 3](#_Toc6924888)

[2.2 Solution 3](#_Toc6924889)

[2.3 Product Perspective 3](#_Toc6924890)

[2.4 Product Functions 4](#_Toc6924891)

[2.5 Operating Environment 4](#_Toc6924892)

[2.6 Design and Implementation Constraints 4](#_Toc6924893)

[2.7 User Documentation 5](#_Toc6924894)

[2.8 Assumptions and Dependencies 5](#_Toc6924895)

[3. Project Requirements 6](#_Toc6924896)

[3.1 Database Requirements 6](#_Toc6924897)

[3.2 Software Requirements 6](#_Toc6924898)

[3.3 Hardware Requirements 6](#_Toc6924899)

[3.4 Mobile Application 7](#_Toc6924900)

[4. Primary Variables 11](#_Toc6924901)

[4.1 AM2315 Sensor 11](#_Toc6924902)

[4.1.1 Temperature 11](#_Toc6924903)

[4.2 CCS811 Sensor 11](#_Toc6924904)

[4.2.1 CO2 11](#_Toc6924905)

[5. Secondary Variables 13](#_Toc6924906)

[5.1 AM2315 Sensor 13](#_Toc6924907)

[5.1.1 Performance 13](#_Toc6924908)

[5.2 CCS811 Sensor 13](#_Toc6924909)

[5.2.1 Performance 13](#_Toc6924910)

[6. Benefits of the System 13](#_Toc6924911)

[6.1 Reliability 13](#_Toc6924912)

[6.2 Costs 14](#_Toc6924913)

[7. Build Instructions 15](#_Toc6924914)

[7.1 Introduction 15](#_Toc6924915)

[7.2 System Diagram 17](#_Toc6924916)

[7.3 Material Requirements and Budget 18](#_Toc6924917)

[7.4 Time Commitment 19](#_Toc6924918)

[7.5 Raspberry Pi Configuration 19](#_Toc6924919)

[7.6 Mechanical Assembly 19](#_Toc6924920)

[7.7 Soldering 19](#_Toc6924921)

[7.8 Power Up 24](#_Toc6924922)

[7.9 I2C Detection 24](#_Toc6924923)

[7.10 Unit Testing 25](#_Toc6924924)

[7.11 Production Testing 25](#_Toc6924925)

[8. Problems Encountered 27](#_Toc6924926)

[8.1 VOC Sensor Inconsistent Readings 27](#_Toc6924927)

[8.2 Date-Time Python to Timestamp Java 27](#_Toc6924928)

[8.3 Plotting Data 27](#_Toc6924929)

[8.4 Data Updates 27](#_Toc6924930)

[9. Approaches and Tests 29](#_Toc6924931)

[9.1 After Burn-In Period 29](#_Toc6924932)

[9.2 Converting Time to Seconds 29](#_Toc6924933)

[9.3 Different Graphs 29](#_Toc6924934)

[10. Progress Reports 31](#_Toc6924935)

[10.1 Report 1 31](#_Toc6924936)

[10.2 Report 2 31](#_Toc6924937)

[10.3 Report 3 32](#_Toc6924938)

[10.4 Report 4 33](#_Toc6924939)

[10.5 Report 5 35](#_Toc6924940)

[11. Conclusion 37](#_Toc6924941)

[12. Recommendations and Future Work 39](#_Toc6924942)

[13. Technical References 41](#_Toc6924943)

[14. Appendix A: Glossary 43](#_Toc6924944)

[15. Appendix B: Abbreviations 43](#_Toc6924945)

[16. Android Mobile Application 45](#_Toc6924946)

[16.1 Activities 45](#_Toc6924947)

[16.1.1 MainActivity.java 45](#_Toc6924948)

[16.2 Fragments 51](#_Toc6924949)

[16.2.1 SummaryFragment.java 51](#_Toc6924950)

[16.2.2 TemperatureFragment.java 56](#_Toc6924951)

[16.2.3 HumidityFragment.java 61](#_Toc6924952)

[16.2.4 eCO2Fragment.java 66](#_Toc6924953)

[16.2.5 HistoryFragment.java 71](#_Toc6924954)

[16.3 Layouts 75](#_Toc6924955)

[16.3.1 fragment\_temperature.xml 75](#_Toc6924956)

[16.3.2 fragment\_humidity.xml 76](#_Toc6924957)

[16.3.3 fragment\_e\_co2.xml 77](#_Toc6924958)

[16.3.4 activity\_summary.xml 78](#_Toc6924959)

[16.3.5 activity\_history.xml 80](#_Toc6924960)

[17. Hardware Integrated 83](#_Toc6924961)

[17.1 Python Script 83](#_Toc6924962)

[17.1.1 greenhouse2.py 83](#_Toc6924963)

# Revision History

|  |  |
| --- | --- |
| Date | Version no. |
| April 5 | v1 |
| April 9 | v2 |

# List of Illustrations

[Figure 1. Schematic diagram of monitoring device. 1](#_Toc5696597)

[Figure 2. Diagram for ideal communication between software and hardware. 3](file://localhost/Users/Princess/Downloads/Greenhouse%20Technical%20Report.docx#_Toc5696598)

[Figure 3. Flowchart of monitor system. 8](#_Toc5696599)

[Figure 4. Final look of the device with an enclosure. 9](#_Toc5696600)

[Figure 5. System diagram for components of the system. 11](#_Toc5696601)

[Figure 6. Connection between Raspberry Pi and 3 sensors using a breadboard. 14](#_Toc5696602)

[Figure 7. Schematic view of the connection between Raspberry Pi and 3 sensors. 14](#_Toc5696603)

[Figure 8. PCB view of the connection between Raspberry Pi and 3 sensors. 15](#_Toc5696604)

[Figure 9. Vias soldered on PCB as circled in red. 16](#_Toc5696605)

[Figure 10. Necessary pins soldered on the 40 pin header. 16](#_Toc5696606)

[Figure 11. 4-and-5-pin on PCB as circled in red. 17](#_Toc5696607)

[Figure 12. Complete look of the PCB. 17](#_Toc5696608)

[Figure 13. Display output of 2 sensors detected. 18](#_Toc5696609)

[Figure 14. Display output of 3 sensors detected. 18](#_Toc5696610)

[Figure 15. Sample output terminal. 19](#_Toc5696611)

[Figure 16. Sample output on OLED screen. 20](#_Toc5696612)

# Introduction

## Purpose

This technical report presents the methods and processes of how we achieved our project result. Atmospheric factors can affect plant growth and the result of our device can help farmers control the factors that affect them. The main components of a monitoring system are measurement, data processing and recording. (Refer to Figure 1). Our project will be used as a monitoring system that can be accessed from a mobile device. The system is ideally situated in a greenhouse for best performance and outcome. The application will provide temperature, humidity and CO2 levels in the greenhouse. This device is meant for agricultural workers that need help to maintain their plant growth if they cannot tend to their plants immediately.

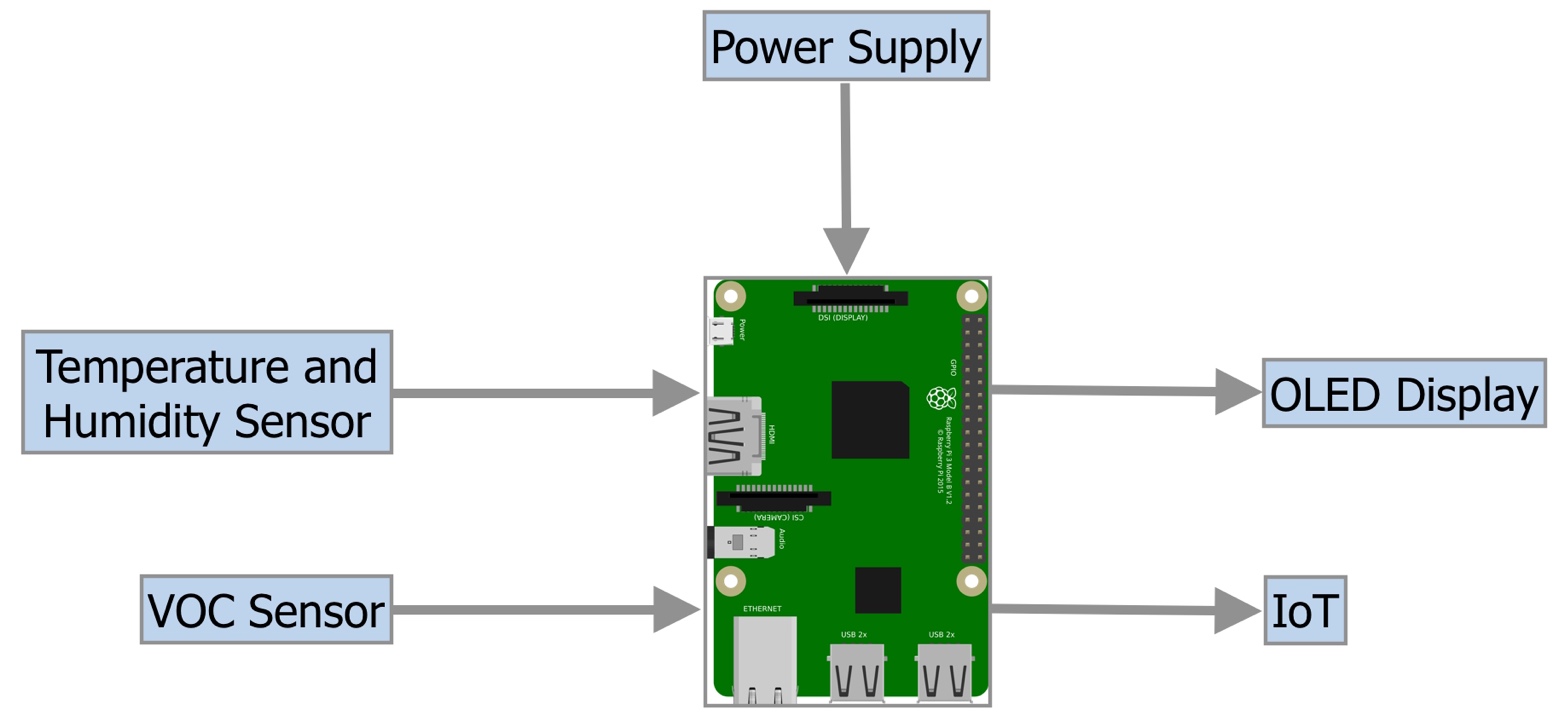


Figure 1. Schematic diagram of monitoring device.

## Document Conventions

This document was created based on the OACETT Technology Report Guidelines.

## Intended Audience and Reading Suggestions

This document is intended for agricultural farmers, greenhouse technicians, and workplace management as a reference when they are using the device and application. Also, the developers of the system are able to contribute and revise to this document at any time.

## Product Scope

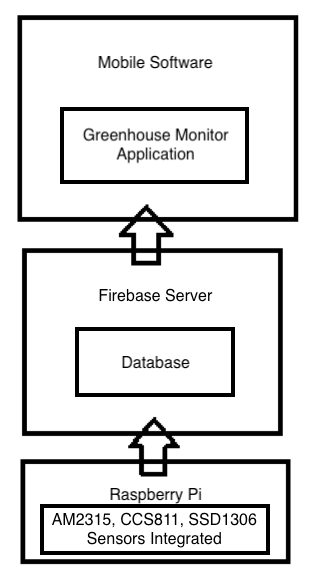
Greenhouse workers must check on their plants frequently and the surrounding factors of plants’ growth. This system will be designed to help raise awareness of the atmosphere factors that affect plants by providing real-time record of temperature, humidity and air quality like CO2.

# Overall Description

## Problem

The environment of plants is consistently monitored throughout the day. Today, there are current devices that measure plant environment but it can take hours to measure temperature, humidity and carbon dioxide respectively.

## Solution

**A system that gathers data of all three (temperature, humidity and carbon dioxide) plant environment is the most ideal solution for this problem. An all-in-one monitoring device allows for easier and faster ways to grow plants at a constant rate.

## Product Perspective

The product has 2 main parts, the mobile app and the integrated sensors. The application is used to view data on Temperature and humidity from AM2315 sensor, and CO2 from CCS811. The OLED (SSD1306) sensor displays the data of all three factors. All sensors will be integrated onto the Raspberry Pi.

The mobile application will need to communicate with the database to get the data from the Raspberry Pi.

Figure 2. Diagram for ideal communication between software and hardware.

The Raspberry Pi is used to communicate with the database by sending data for it to be stored.

Refer to Figure 2.

## Product Functions

The device and application need an internet connection to function, and will connect to a Firebase database server. The product must be connected to the Internet to do the following:

* Read the temperature, and send it to the server so the application can read it
* Read the humidity, and send it to the server so the application can read it
* Read the CO2 levels, and send it to the server so the application can read it

## Operating Environment

The device operating environment is mostly ideal for indoor, but it can also work outdoor. The mobile application can be used anywhere with an Internet connection, software is android lollipop and up.

## Design and Implementation Constraints

The Internet connection is a constraint for the application. It isn’t a huge constraint but the application fetches data from the database once in a while or the static wouldn’t update, it is crucial that there is an Internet connection for the application to function. The Raspberry Pi itself has the same constraint due to the need of it sending data to the database as well.

## User Documentation

All documentation and source code for our firmware and Android Application are available through GitHub as it is open source.

## Assumptions and Dependencies

One assumption about the product will be use on a mobile phone that meets the required specifications. If the phone does not have the hardware resources required for the application, for example other application has already allocated space, the application may not work as intended. The application is developed in Java and some functions are conversion of data from Python to Java. As all the code in this project is open source on Github, they are able to modify it as they please with correct use of Python/Java coding formats.

# 

# Project Requirements

## Database Requirements

We are using Google’s Firebase as a database, which can be access individually between us – the developers. The server-side application requires a Google account to access Firebase. The database will store the readings of the temperature, humidity, CO2, as well as date and time stamp. The application will be connected with the database to show the readings stored on the database to the application. Kenneth will be working on the database, and Colin will link the database to connect to the application to retrieve readings from the Pi.

## Software Requirements

The mobile application, Greenhouse Monitor, requires the target (currently only on Android platforms) to be at API level 21 and/or up to run. Greenhouse Monitor will take data from the database with date and time stamp, and shows information of temperature, humidity and CO2. The most important and main fragment shows all three of the sensors’ information, which is the most recent entry retrieved from the Raspberry Pi to the database. The application will be updated every time it is ran and manually updates the information shown from the database. Kenneth will be working on developing the application and its features, and Colin will develop the GUI (XML files).

## Hardware Requirements

The AM2315 sensor measures the temperature and humidity, and the CCS811 sensor measures the levels of CO2 that are in the air. Both sensors are integrated together with Raspberry Pi. The completed hardware will be able to be connected with the software in order to receive and store data in a remote database. Additionally, OLED will be added as a feature to show readings of temperature, humidity and CO2. Princess will be integrating all these sensors onto her Raspberry Pi.

## Mobile Application

The mobile application will retrieve data from the database, as well as the date and time, to display on the main fragment. The date and time stamp are converted into readable date and time format as Python’s time format is different from the time format of Java’s. One other fragment in the application shows previous data readings plotted onto a graph. The user is able to view data history so that they are able to compare and observe plant behaviours in different times throughout the day.

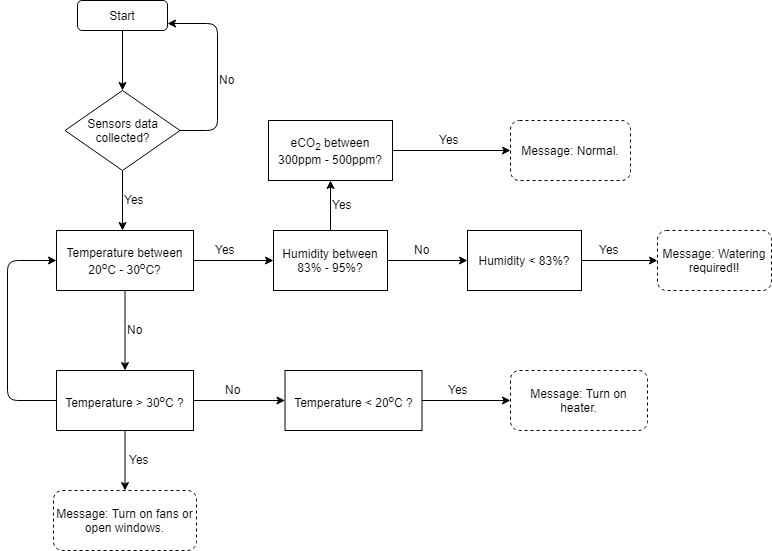


Figure 3. Flowchart of application’s monitor system.

Figure 3 shows a flowchart of the application’s control system. As the database is updated dynamically, the application fetches the updated data. The temperature, relative humidity and equivalent carbon dioxide are dependent on the greenhouse’s environment. Based on the data collected, temperature, humidity and carbon dioxide must meet a desirable level in order for plant growth to be successful at a constant rate. The user will receive a message to water the plants when the humidity conditions are not met, e.g. humidity is less than 83%. When the temperature conditions are not met, e.g. temperature is greater than 30 degrees Celsius or less than 20 degrees Celsius, the user will get a message to turn on fans or turn on heater respectively. Considering the temperature to go about freezing or cooling point, greenhouses are meant to trap heat inside. Thus the temperature that is less than 20 degrees Celsius is not an ideal environment in a greenhouse.

# Primary Variables

We can consider the primary variables in which will directly affect the design results due to assumptions the developers made.

## AM2315 Sensor

### Temperature

Every greenhouse must have a maintained and ideal temperature inside. The thermistor temperature sensor incorporated with humidity sensor in one are integrated with a microcontroller that does the readings and outputs data.

## CCS811 Sensor

### CO2

CO2 sensor provides very inaccurate readings because the sensor is unstable. Some readings start out from 9000 ppm and stays within the 9000s area. It will only come back to normal when we test run the sensor multiple times.

# Secondary Variables

We can consider the primary variables in which will directly affect the design results outside the control of the developers.

## AM2315 Sensor

### Performance

The accuracy of the sensor, according to the datasheet, is ±1 degrees Celsius. However, there are moments where the temperature sensor drops from 25.0 degrees Celsius to 0.0 degrees Celsius. The datasheet mentions the possibility of electrical characteristics, such as high, low, input and output voltage, being its cause of range jump. It is depended on the power given to the Raspberry Pi.

## CCS811 Sensor

### Performance

The datasheet recommends a 48-hour burn in time and 20-min warm up time. This must be done every time we are testing the system.

# Benefits of the System

## Reliability

To improve reliability, we tried to keep the system as simple as possible. This will help the agricultural industry to rely on the system more without having to do most work. To those who are unable to maintain their greenhouse, whether they are at home or away from home, the device enables them to keep an eye on their greenhouse through the application. That way, they do not have to constantly check the greenhouse very so often.

## Costs

With a simple system comes with costs. We tried to keep our system at a low cost as it is a small scale. For one device, we wish to sell in the market for $39.99. Ideally, multiple devices should be stationed in a greenhouse. Depending on the size of the greenhouse, there should be 3 - 4 devices for 6.56 ft. x 8.2 ft.

6.3 Opportunities for Energy Conservation

Energy usage is one of the factors that can help maintain a greenhouse. By maintaining the normal range for temperature, humidity and CO2 farmers will not have to leave their fans or water running for too long. As soon as the range becomes normal, fans and water can be turned off thus saving energy and water.

# Build Instructions

Figure 4. Final look of the device with an enclosure.

## Introduction

The Greenhouse Monitoring System is a device that allows greenhouse technicians to monitor atmospheric factors that may affect plant growth. The temp/humid sensor measures temperature and humidity respectively, while the VOC sensor can measure equivalent CO2. These three factors can have an impact in plants whether they are growing at a normal or slow pace. To increase awareness, our monitoring system serves as a tool for greenhouse technicians to maintain and control the temperature, humidity and CO2 levels in a greenhouse environment.

## System Diagram

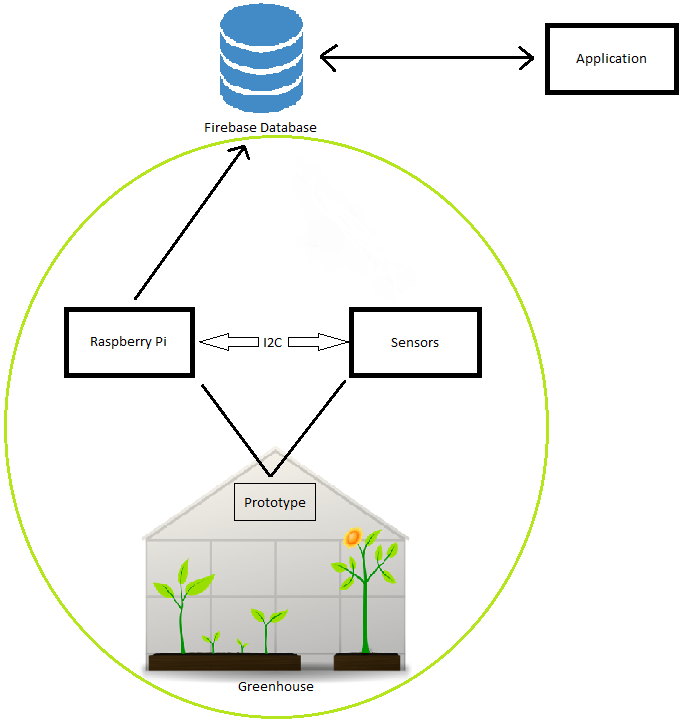


Figure 5. System diagram for components of the system.

## Material Requirements and Budget

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Source | CAD Price | USD Price | Link |
| CanaKit Raspberry Pi 3 B+ Starter Kit | Amazon | $98.99 |  | https://www.amazon.ca/CanaKit-Raspberry-Starter-Premium-Black/dp/B07BCC8PK7/ref=sr\_1\_1\_sspa?crid=3FN3IC415XF1B&keywords=raspberry+pi+3&qid=1551450371&s=electronics&sprefix=raspberry+%2Celectronics%2C153&sr=1-1-spons&psc=1 |
| OLED Display for Arduino 128x64 Pixel I2C, 0.96 inch, SSD1306, Library, 3-5V | Amazon | $6.29 |  | https://www.amazon.ca/Display-Arduino-128x64-SSD1306-Library/dp/B077D4RQG1/ref=sr\_1\_1\_sspa?hvadid=324955435621&hvdev=c&hvlocphy=9000993&hvnetw=g&hvpos=1t1&hvqmt=b&hvrand=7609421822030419687&hvtargid=kwd-45652301614&keywords=oled+ssd1306&qid=1551449756&s=electronics&sr=1-1-spons&tag=googcana-20&psc=1 |
| 20x2-pin Header (Female) | Amazon | $9.99 |  | https://www.amazon.ca/2x20-pin-Female-Stacking-Header-Raspberry/dp/B071FT161B/ref=sr\_1\_3?crid=SKA71RFLSHC5&keywords=20+pin+header+female&qid=1551450588&s=electronics&sprefix=20+pin+header%2Celectronics%2C149&sr=1-3 |
| 5-pin Header (Female) | Creatron | $0.43 |  | https://www.creatroninc.com/product/5-pin-receptacle-socket/?search\_query=4+pin+stackable+header&results=48 |
| 4-pin Header (Female) | Creatron | $0.42 |  | https://www.creatroninc.com/product/4-pin-receptacle-socket/ |
| AM2315 - Encased I2C Temperature/Humidity Sensor | Adafruit |  | $29.95 | https://www.adafruit.com/product/1293 |
| SparkFun Air Quality Breakout - CCS811 | Sparkfun |  | $20.95 | https://www.sparkfun.com/products/14193?\_ga=2.97662492.2095878335.1537831851 |

Table 1. List of materials and their respective prices needed to build the system.

\*Prices does not include tax and shipping.

## Time Commitment

This project can be completed in approximately 7 days if you followed the mechanical assembly and diagrams. Desired to do so, you must order the materials 2 - 3 weeks before working on this project as shipping takes about 1 - 2 weeks. Once you receive the materials, you can start configuring your Raspberry Pi which will take around 3 hours. Connecting the parts to the Raspberry Pi will take 30 minutes. And then to set up the code on the Raspberry Pi and testing the code will take around a 1 hour. It is recommended that you take 3 - 4 hours working on this project daily.

## Raspberry Pi Configuration

1. Create your Raspberry Pi's [image](https://github.com/six0four/StudentSenseHat/blob/master/cribpisdcard.md) for your project.
2. Then test your LED to [blink](https://github.com/six0four/StudentSenseHat/blob/master/README.md#student-raspberry-pi-image-creation-and-test-code).

## Mechanical Assembly

1. Set your sensors onto your breadboard and connect your sensors to the appropriate GPIO pinout of the Raspberry Pi. The wiring should look something like this:
2. Boot your Raspberry Pi and open terminal.
3. To check if your sensors are functioning, click [here](#i2c-detection).

## Soldering

1. You can design your own PCB using [Fritzing software](http://fritzing.org/download/) for free or our version of the [fritzing file](https://github.com/PrincessHernandez/GreenhouseMonitoringSystem/blob/master/documentation/Fritzing/Greenhouse.fzz). You can refer to the images below:

Breadboard view

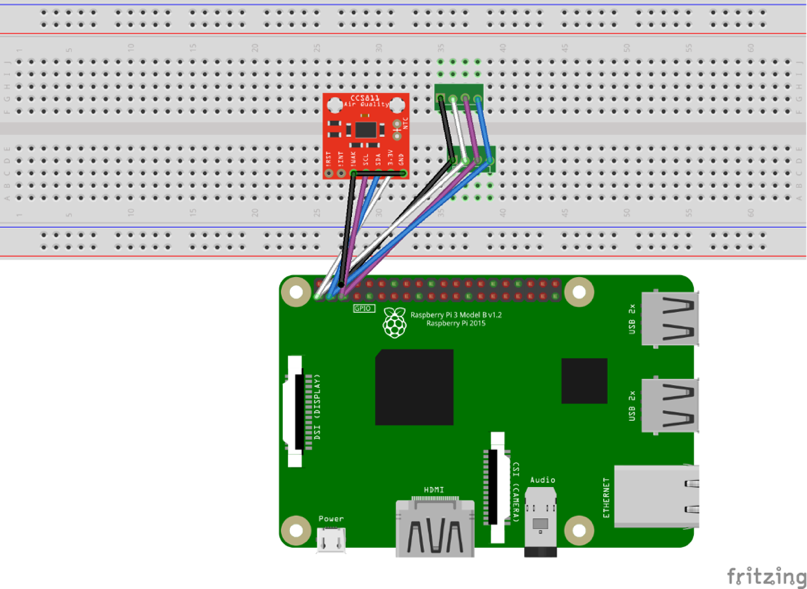


Figure 6. Connection between Raspberry Pi and 3 sensors using a breadboard.

Schematic view

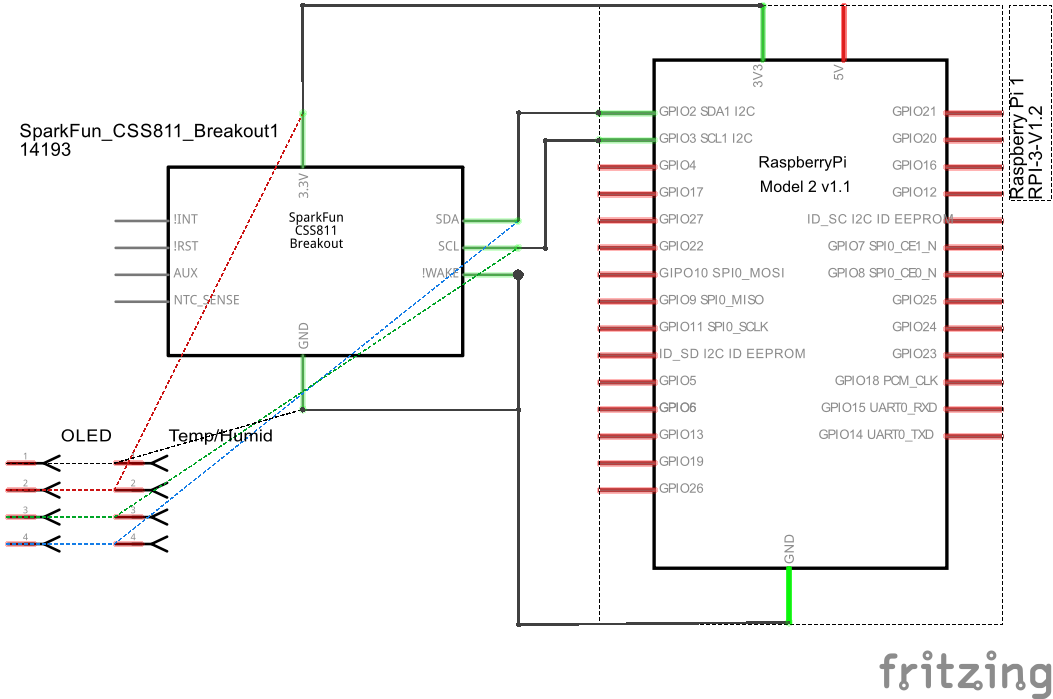


Figure 7. Schematic view of the connection between Raspberry Pi and 3 sensors.

PCB view

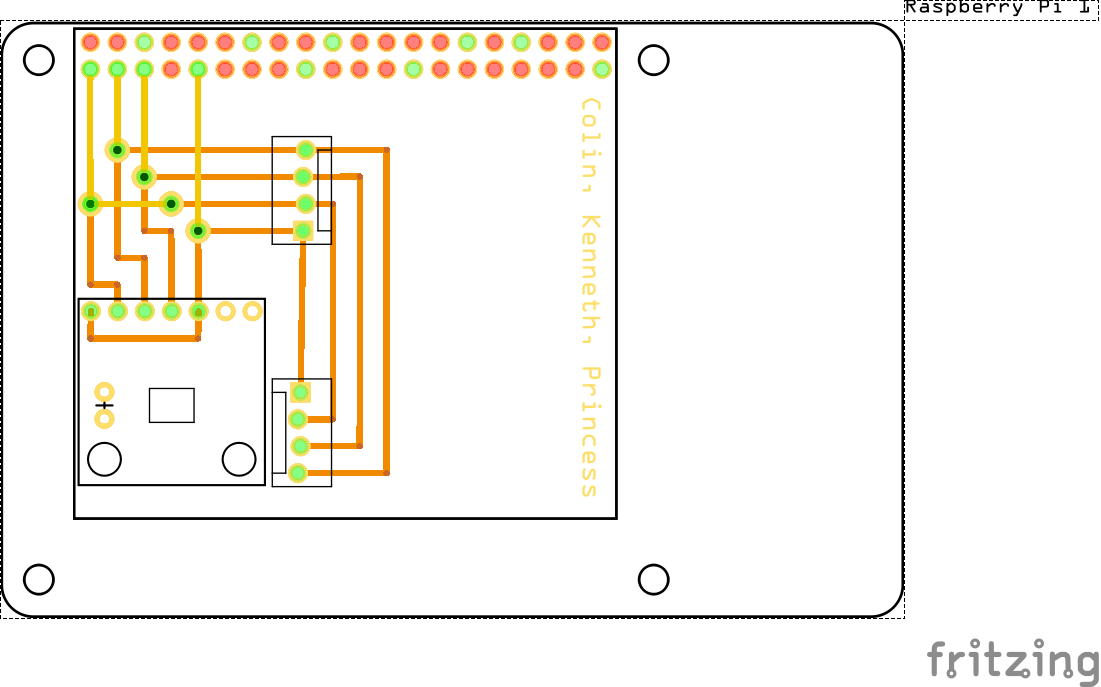


Figure 8. PCB view of the connection between Raspberry Pi and 3 sensors.

Here are the following pins that you should know for this project:

* *Power Pins*
* Vin - power pin
  + Since the sensor uses 3.3V, give it the same power as the logic level of you Raspberry Pi.
* GND - common ground for power and logic
* *Logic Pins*
* SCL - i2c clock pin
  + Connect to your Raspberry Pi i2c clock line.
* SDA - i2c data pin
  + Connect to your Raspberry Pi i2c data line.
* WAKE\* - wakeup pin for the sensor
  + Please make sure that the WAKE pin is connected to GND. Otherwise, you will not get the address for CCS811 sensor.

1. Once you have obtained your PCB board solder the following:

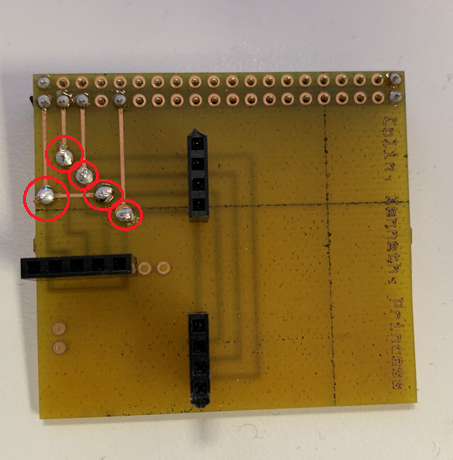
* Vias\*
* 

Figure 9. Vias soldered on PCB as circled in red.

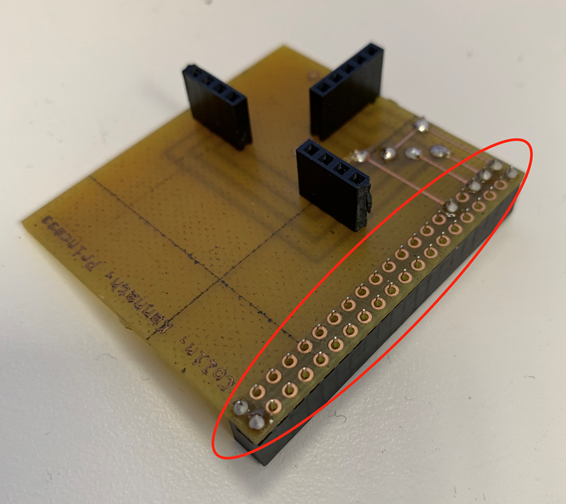
* \*Note: You must thread a single strand of wire through the holes, solder it, and then cut the remaining wires off.
* 20x2-pin socket
* 

Figure 10. Necessary pins soldered on the 40 pin header.

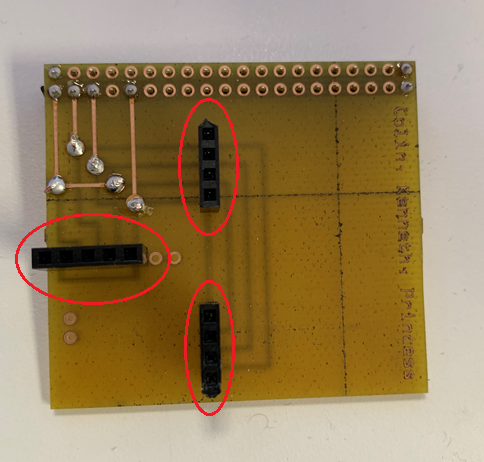
* 4 and 5-pin sockets
* 

Figure 11. 4-and-5-pin on PCB as circled in red.

Once you have finished soldering, your board should look like this along with the sensor:

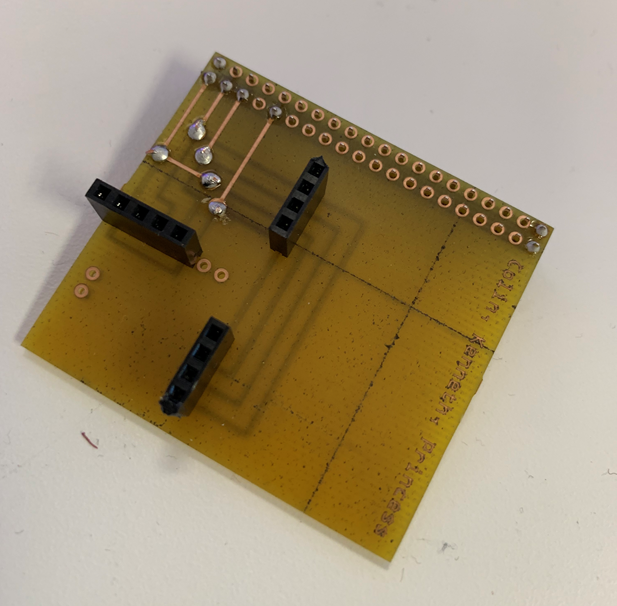


Figure 12. Complete look of the PCB.

## Power Up

1. Boot your Raspberry Pi and open terminal.
2. To check if your sensors are functioning, click [here](#i2c-detection). If they are not being detected, there might be a problem with the solder or the PCB itself.
3. Download the following codes here to setup your sensors: [CCS811](https://github.com/PrincessHernandez/GreenhouseMonitoringSystem/blob/master/python%20script/classes/CCS811_RPi.py) and [SSD1306](https://github.com/PrincessHernandez/GreenhouseMonitoringSystem/blob/master/python%20script/classes/SSD1306.py).

## I2C Detection

To check if the board is functioning and detecting the sensor, open terminal and type

sudo i2cdetect -y 1

This will display an output of the sensor's address - 0x3C, 0x5B. See sample output below:

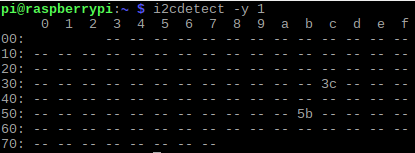


Figure 13. Display output of 2 sensors detected.

To wake up AM2315 Temp/Humid sensor, run the command(quickly) again. It will show all addresses - 0x3C, 0x5B, 0x5C.

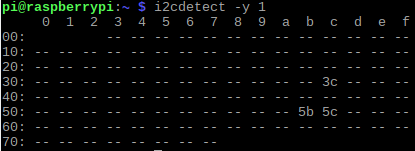


Figure 14. Display output of 3 sensors detected.

## Unit Testing

Download the program [greenhouse.py](https://github.com/PrincessHernandez/GreenhouseMonitoringSystem/blob/master/python%20script/greenhouse.py). Now that you've downloaded the codes and detected your sensors, go to terminal and run

sudo python greenhouse.py

This command should run the program and display on the terminal. An example output is shown below:

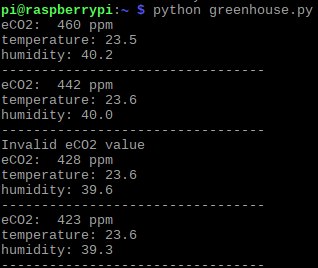


Figure 15. Sample output terminal.

## Production Testing

Now that you've seen what the sensors' output will look like, download the program [greenhouse2.py](https://github.com/PrincessHernandez/GreenhouseMonitoringSystem/blob/master/python%20script/greenhouse2.py) and run

sudo python greenhouse2.py

This command should run the program and display the output on the OLED screen. An example output is shown below:

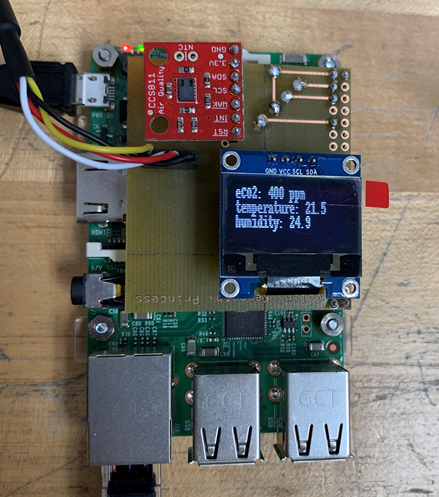


Figure 16. Sample output on OLED screen.

# 

# Problems Encountered

## VOC Sensor Inconsistent Readings

According to the hookup guide on the SparkFun website, the VOC sensor needs a 48-hour burn in time and a 20-minute warmup. However, the readings go above 8192 ppm each time we run the program. Also, there are moments the readings jump spontaneously, e.g. 50 000 ppm, which is inconsistent with previous readings that start from 400 ppm. Many users of the sensor are also having the same issue and it is yet to be solved.

## Date-Time Python to Timestamp Java

We are having trouble converting the date and time from the database to the application. The date and time for Python is formatted in string, while the timestamp for Java is in integer. This is the very reason we cannot plot the time on the graph in the application.

## Plotting Data

After fetching the data and timestamp from the database, we noticed that the levels of the eCO2 has a bigger range compared to temperature or humidity. Looking at the graph, it did not look pleasing visually and the data was hard to see on the Y-axis.

## Data Updates

The mobile application crashes every time the Raspberry Pi sends data to the Firebase.

# Approaches and Tests

## After Burn-In Period

In order to ensure that the VOC sensor performance is stable, we must run the sensor for 20 minutes before we start testing and/or troubleshooting. We noticed more accurate readings are generated when doing this step and therefore, we must do this each time we are testing the device.

## Converting Time to Seconds

To plot the time in the graph, we worked on sending the time as seconds by converting epoch to readable date and time. The application grabs the time from the database as a string in seconds and parse the time into a date format. The date is plotted into the graph and is multiplied by 1000 to be displayed. The date uses a SimpleDateFormat function to display the date in any format we want. In this case, the format is MMM HH:MM.

## Different Graphs

We have decided that the best way to plot the data is to have different graphs for all three – temperature, humidity and equivalent carbon dioxide. This will show a clearer line graph as the results of the data correspond with their respective units on the Y-axis.

# Progress Reports

## Report 1

|  |  |
| --- | --- |
|  | Thu, Jan 31, 2019 at 6:06 PM |
|  | |
| |  | | --- | | Dear Kristian,    **Group Update**  As of today we now have all of the hardware for our project. We have decide to create a new PCB to integrate all three of our sensors. The fritzing files has been complete and sent to the prototype lab for creation. Colin has begun working on the database. Kenneth hasstarted work on our application.    **Financial**  Financially we needed to purchase an OLED screen at the cost of $14.40 and a fan for $13.55. The project will also require us to purchase 2 four pin headers and a 5 pin header along with a 24 pin GPIO header with a total cost $15.09.    **Problems**  Some issues we have encountered are trying to get all of parts together and into a case and how they will all fit together because two sensors have to be exposed outside of the case.  Sincerely,  Princess, Colin, Kenneth | | |

## Report 2

|  |  |
| --- | --- |
|  | Thu, Feb 28, 2019 at 6:53 PM |
|  | |
| |  | | --- | | Dear Kristian,    **Summary**  As of today, we are on track with the project. Kenneth finished the structure of the database, but it may change over the next few weeks (we may rearrange the structure because the tested data and user is separate - tied by User ID). The application is functional - reads the data from the database and displays data in a graph or the most recent value. The new PCB we created has been soldered. Princess and Colin are able to get all sensors to work together.    **Financial**  The Prototype Lab does not have 20-pin headers stocked, and Kelly and Vlad are not sure when they will be back in stock. So, we used a 25-pin header to make a 20-pin header. During reading week, we were able to finish soldering the PCB.  **Plans**   We need to work on getting the data from the Raspberry Pi to the Firebase. Also, we are planning to omit the fan in our hardware because it is outside of the scope of our project. We are also planning on creating a new case for the Pi and sensors.  Sincerely,  Colin, Kenneth, Princess | | |

## Report 3

|  |  |
| --- | --- |
|  | Thu, Mar 14, 2019 at 5:51 PM |
|  | |
| |  | | --- | | Dear Kristian,    **Group Update**  We are following the schedule and we believe we can get everything done on time before the Project Demonstration on April 11. Colin was able to send data from the sensors to the database.    **Current Progress**  Princess is working on designing the enclosure. We expect to have it created by next week. Kenneth is making progress on the mobile app, and is currently working on retrieving the live sensor data from the database to be displayed on the mobile application. The technical report is constantly being updated.  **Financial**  We are planning to buy the presentation board for the upcoming CTI presentations. We are still looking for a cheaper option than from Staples.  Sincerely,  Colin, Kenneth, Princess | | |

## Report 4

|  |  |
| --- | --- |
|  | Thu, Mar 28, 2019 at 6:02 PM |
|  | |
| |  | | --- | | Dear Kristian,    **Current Progress**  Currently, the Raspberry Pi is able to send data to the database (Firebase) dynamically with timestamp. A few changes in the firmware will be made in a couple of days to accommodate with the software part of the application. The Android application is complete (able to read data dynamically), but Kenneth is working on getting the time to show on the graph (Python sends the date in a different format), which is our main priority. Enclosure is complete and no modifications are required.  **Problems**  According to the hookup guide on the SparkFun website, the VOC sensor needs a 48-hour burn in time and a 20-minute warmup. However, the readings go above 8192 ppm each time we run the program. Also, there are moments the readings jump spontaneously, e.g. 50 000 ppm, which is inconsistent with previous readings that start from 400 ppm. We’ve searched on SparkFun Forum (<https://forum.sparkfun.com/viewtopic.php?t=47161>) and many users of the sensor are also having the same issue and it is yet to be solved.  **Solution(s)**  In order to ensure that the VOC sensor’s performance is stable, we must run the sensor for 20 minutes before we start testing (every time). We noticed more accurate readings are generated when doing this step and therefore, we must do this each time we are testing the device. This is the only solution we have come up with so far. The solution for the time issue is to send the time as seconds since the epoch to the database and parse the time into a date format on the application.  **Other**  We are hoping to get the application to plot the time on the graph by the end of this week as we are trying to prepare for the upcoming project demonstration. Overall, there are no other problems we have encountered.  Sincerely,  Colin, Kenneth, Princess | | |

## Report 5

|  |  |
| --- | --- |
|  | Thu, Apr 11, 2019 at 7:01 PM |
|  | |
| |  | | --- | | Dear Kristian,    **Current Progress**  On our previous status update, we expected the data+timestamp to show on the graph by the end of week 11. However, this took an extra week to finish, as we've tried various ways of plotting information. We've concluded to having three different graphs as there is a big difference in temperature levels compared to CO2 levels. The hardware portion is complete and no modifications are needed.  **Problem(s)**  The application crashes every time the Pi sends data to the database. The application needs a few modifications for retrieving data.  **Future plans/Expectations**  We expect to have the application fully completed by the end of this week.  **Other**  We've started preparing for the presentation.  Sincerely,  Colin, Kenneth, Princess | | |

# Conclusion

In this technical report, we have designed and implemented a monitoring system that can understand a greenhouse environment. After completing the implementation (Hardware and Software) of the monitoring system, the resulting prototype is advantageous as it employs three components that measure plants’ surrounding environment (Temperature, Humidity and VOC sensors). To assist those who are farmers and technicians in greenhouses, the monitoring system was created to eliminate the extra required work to measure the surrounding environment of plants and enables multitasking by measuring temperature, humidity and carbon dioxide at the same time. The system’s main feature allows greenhouse workers to maintain the growth of plants by checking the monitoring system’s display and/or by checking the mobile application. The system has the potential to be used in greenhouses to simplify work for greenhouse farmers and technicians, and to monitor the atmospheric factors anytime and anywhere on the mobile application for a more efficient use, so long as they are connected to the internet.

# Recommendations and Future Work

Greenhouse control environment considerations provide a way for us to look at available options on greenhouse monitors. We are aware that our device is a small scale for the intention of monitoring a large area. One solution to this problem is to install multiple number of this device on a greenhouse depending on its size. However, this will create a problem for the application as it is connected with only one device. The mobile application and hardware can be refined to be integrated with multiple devices in the future for a larger scale production.

# Technical References

Brander, M. (2012, August). *Greenhouse Gases, CO2, CO2e, and Carbon: What*

*Do All These Terms Mean?* [PDF File]. Retrieved from https://ecometrica.com/assets/GHGs-CO2-CO2e-and-Carbon-What-Do-These-Mean-v2.1.pdf

DeVito, J. & DiCola, T. (2017). Adafruit Circuit Python SSD1306. GitHub repository,

https://github.com/adafruit/Adafruit\_CircuitPython\_SSD1306

Gohar, L. K. & Shine, K. P. (2007, November). *Equivalent CO2 and its use in*

*understanding the climate effects of increased greenhouse gas concentrations*.

[PDF File]. Retrieved from https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/wea.103

Blakley, C., Chen, K. & Hernandez, P. (2017, July 11). Greenhouse Monitoring System.

GitHub repository,

https://github.com/PrincessHernandez/GreenhouseMonitoringSystem

Jones, J.B. (2013, December 1). *Maintaining Control in the Greenhouse*. Retrieved from

https://www.maximumyield.com/maintaining-control-in-the-greenhouse/2/949

Lukas, P. (2017, July 11). CCS811\_RPi. GitHubGist repository,

https://gist.github.com/xxlukas42/60ae08f75e68a0cfcdb7c9dd60145d34

Omid, Mahmoud & A, Shafaei. (2005). *Temperature and relative humidity changes inside*

*greenhouse.* International Agrophysics. 19. 153-158. [PDF File]. Retrieved from

https://www.researchgate.net/publication/26552070\_Temperature\_and\_relative\_hu

midity\_changes\_inside\_greenhouse

Taylor, M. (2017, April 27). CCS811 Air Quality Breakout Hookup Guide. SpakFun

Electronics. Retrieved from https://learn.sparkfun.com/tutorials/ccs811-air-quality-

breakout-hookup-guide?\_ga=2.265865676.210527324.1553258399-

1038958827.1553258399

# Appendix A: Glossary

Epoch – Date and time relative to time on the Raspberry Pi

Equivalent Carbon Dioxide – Different compounds that is made of the same common unit of Carbon and Oxygen

GPIO – Interface used to connect other microcontrollers to other electronic devices

I2C – Multiple chips connected to the same bus

PPM – A unit used to determine extent of pollution existing in the air

# Appendix B: Abbreviations

CO2 – Carbon Dioxide

eCO2 – Equivalent Carbon Dioxide

GPIO – General Purpose Input/Output

I2C – Inter-Integrated Circuit

PCB – Printed Circuit Board

PPM – Parts Per Million

VOC – Volatile Organic Compound

# Android Mobile Application

## Activities

### MainActivity.java

|  |
| --- |
| package com.greenhouseproject; |
|  |  |
|  | import android.content.Intent; |
|  | import android.os.Bundle; |
|  | import android.support.annotation.NonNull; |
|  | import android.support.v4.app.FragmentManager; |
|  | import android.support.design.widget.NavigationView; |
|  | import android.support.v4.app.FragmentTransaction; |
|  | import android.support.v4.view.GravityCompat; |
|  | import android.support.v4.widget.DrawerLayout; |
|  | import android.support.v7.app.ActionBarDrawerToggle; |
|  | import android.support.v7.app.AppCompatActivity; |
|  | import android.support.v7.widget.Toolbar; |
|  | import android.util.Log; |
|  | import android.view.MenuItem; |
|  | import android.view.View; |
|  | import android.widget.TextView; |
|  | import android.widget.Toast; |
|  |  |
|  | import com.google.firebase.auth.FirebaseAuth; |
|  | import com.google.firebase.auth.FirebaseUser; |
|  | import com.google.firebase.database.DatabaseReference; |
|  | import com.google.firebase.database.FirebaseDatabase; |
|  |  |
|  | public class MainActivity extends AppCompatActivity |
|  | implements NavigationView.OnNavigationItemSelectedListener { |
|  |  |
|  | private FirebaseAuth.AuthStateListener mAuthStateListener; |
|  | private static final String TAG = "login out"; |
|  |  |
|  | @Override |
|  | protected void onCreate(Bundle savedInstanceState) { |
|  | super.onCreate(savedInstanceState); |
|  | setContentView(R.layout.activity\_main); |
|  |  |
|  | setupFirebaseListener(); |
|  |  |
|  | Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar); |
|  | setSupportActionBar(toolbar); |
|  | /\* |
|  | FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab); |
|  | fab.setOnClickListener(new View.OnClickListener() { |
|  | @Override |
|  | public void onClick(View view) { |
|  | Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH\_LONG) |
|  | .setAction("Action", null).show(); |
|  | } |
|  | }); |
|  | \*/ |
|  |  |
|  | DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout); |
|  | ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( |
|  | this, drawer, toolbar, R.string.navigation\_drawer\_open, R.string.navigation\_drawer\_close); |
|  | drawer.addDrawerListener(toggle); |
|  | toggle.syncState(); |
|  |  |
|  | //NavigationView navigationView = (NavigationView) findViewById(R.id.nav\_view); |
|  | //navigationView.setNavigationItemSelectedListener(this); |
|  |  |
|  | FirebaseAuth firebaseauth = FirebaseAuth.getInstance(); |
|  |  |
|  | NavigationView navigationView = (NavigationView) findViewById(R.id.nav\_view); //displays text of header of nav drawer. |
|  | navigationView.setNavigationItemSelectedListener(this); |
|  | View headerview = navigationView.getHeaderView(0); |
|  |  |
|  | TextView tt1 = (TextView) headerview.findViewById(R.id.textview\_username); |
|  | tt1.setText(firebaseauth.getCurrentUser().getDisplayName());//username of logged in user. |
|  |  |
|  | TextView tt = (TextView) headerview.findViewById(R.id.textView\_emailid); |
|  | tt.setText(firebaseauth.getCurrentUser().getEmail()); //email id of logged in user. |
|  |  |
|  | FragmentManager fragmentManager = getSupportFragmentManager(); |
|  | fragmentManager.beginTransaction() |
|  | .replace(R.id.content\_frame, new SummaryFragment()).commit(); |
|  | } |
|  |  |
|  | private void setupFirebaseListener() { |
|  | Log.d(TAG, "setupFirebaseListener: setting up the auth state listener."); |
|  | mAuthStateListener = new FirebaseAuth.AuthStateListener() { |
|  | @Override |
|  | public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) { |
|  | FirebaseUser user = firebaseAuth.getCurrentUser(); |
|  | if (user != null) { |
|  | Log.d(TAG, "onAuthStateChanged: signed\_in: " + user.getUid()); |
|  | } else { |
|  | Log.d(TAG, "onAuthStateChanged: signed\_out"); |
|  | Toast.makeText(MainActivity.this, "Signed out", Toast.LENGTH\_SHORT).show(); |
|  | Intent intent = new Intent(MainActivity.this, LoginActivity.class); |
|  | intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK); |
|  | startActivity(intent); |
|  | } |
|  | } |
|  | }; |
|  | } |
|  |  |
|  | @Override |
|  | public void onStart() { |
|  | super.onStart(); |
|  | FirebaseAuth.getInstance().addAuthStateListener(mAuthStateListener); |
|  | } |
|  |  |
|  | @Override |
|  | public void onStop() { |
|  | super.onStop(); |
|  | if(mAuthStateListener != null){ |
|  | FirebaseAuth.getInstance().removeAuthStateListener(mAuthStateListener); |
|  | } |
|  | } |
|  |  |
|  | @Override |
|  | public void onBackPressed() { |
|  | DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout); |
|  | if (drawer.isDrawerOpen(GravityCompat.START)) { |
|  | drawer.closeDrawer(GravityCompat.START); |
|  | } else { |
|  | super.onBackPressed(); |
|  | } |
|  | } |
|  |  |
|  | /\* remove the setting button |
|  | @Override |
|  | public boolean onCreateOptionsMenu(Menu menu) { |
|  | // Inflate the menu; this adds items to the action bar if it is present. |
|  | getMenuInflater().inflate(R.menu.main, menu); |
|  | return true; |
|  | } |
|  |  |
|  | @Override |
|  | public boolean onOptionsItemSelected(MenuItem item) { |
|  | // Handle action bar item clicks here. The action bar will |
|  | // automatically handle clicks on the Home/Up button, so long |
|  | // as you specify a parent activity in AndroidManifest.xml. |
|  | int id = item.getItemId(); |
|  |  |
|  | //noinspection SimplifiableIfStatement |
|  | if (id == R.id.action\_settings) { |
|  | return true; |
|  | } |
|  |  |
|  | return super.onOptionsItemSelected(item); |
|  | } |
|  | \*/ |
|  |  |
|  | @SuppressWarnings("StatementWithEmptyBody") |
|  | @Override |
|  | public boolean onNavigationItemSelected(MenuItem item) { |
|  | // Handle navigation view item clicks here. |
|  | int id = item.getItemId(); |
|  | FragmentManager fragmentManager = getSupportFragmentManager(); |
|  |  |
|  | if (id == R.id.nav\_summary) { |
|  | // Handle the camera action |
|  | fragmentManager.beginTransaction() |
|  | .replace(R.id.content\_frame, new SummaryFragment()) |
|  | .commit(); |
|  | } else if (id == R.id.nav\_humidity) { |
|  | fragmentManager.beginTransaction() |
|  | .replace(R.id.content\_frame, new HumidityFragment()) |
|  | .commit(); |
|  | } else if (id == R.id.nav\_temperature) { |
|  | fragmentManager.beginTransaction() |
|  | .replace(R.id.content\_frame, new TemperatureFragment()) |
|  | .commit(); |
|  |  |
|  | } else if (id == R.id.nav\_eco2) { |
|  | fragmentManager.beginTransaction() |
|  | .replace(R.id.content\_frame, new eCO2Fragment()) |
|  | .commit(); |
|  |  |
|  | // } else if (id == R.id.nav\_slideshow) { |
|  |  |
|  | // } else if (id == R.id.nav\_manage) { |
|  | /\* |
|  | } else if (id == R.id.nav\_settings) { |
|  | Intent i = new Intent(this,SettingsActivity.class); |
|  | startActivity(i); |
|  | \*/ |
|  | } else if (id == R.id.nav\_sign\_out) { |
|  | Log.d(TAG, "onClick: attempting to sign out the user."); |
|  | FirebaseAuth.getInstance().signOut(); |
|  |  |
|  | } |
|  |  |
|  | DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer\_layout); |
|  | drawer.closeDrawer(GravityCompat.START); |
|  | return true; |
|  | } |
|  | } |

## Fragments

### SummaryFragment.java

|  |
| --- |
| package com.greenhouseproject; |
|  |  |
|  | import android.graphics.Color; |
|  | import android.os.Bundle; |
|  | import android.support.annotation.NonNull; |
|  | import android.support.annotation.Nullable; |
|  | import android.support.v4.app.Fragment; |
|  | import android.util.Log; |
|  | import android.view.LayoutInflater; |
|  | import android.view.View; |
|  | import android.view.ViewGroup; |
|  | import android.widget.Button; |
|  | import android.widget.EditText; |
|  | import android.widget.TextView; |
|  |  |
|  | import java.util.ArrayList; |
|  | import java.util.List; |
|  |  |
|  | import lecho.lib.hellocharts.model.PieChartData; |
|  | import lecho.lib.hellocharts.model.SliceValue; |
|  | import lecho.lib.hellocharts.view.PieChartView; |
|  |  |
|  | import com.google.firebase.auth.FirebaseAuth; |
|  | import com.google.firebase.auth.FirebaseUser; |
|  | import com.google.firebase.database.DataSnapshot; |
|  | import com.google.firebase.database.DatabaseError; |
|  | import com.google.firebase.database.DatabaseReference; |
|  | import com.google.firebase.database.FirebaseDatabase; |
|  | import com.google.firebase.database.ValueEventListener; |
|  |  |
|  | public class SummaryFragment extends Fragment{ |
|  |  |
|  | View myView; |
|  |  |
|  | TextView tempText, humidText, co2Text, statText; |
|  | Button btnCData; |
|  | FirebaseDatabase database; |
|  | DatabaseReference ref; |
|  |  |
|  | FirebaseUser authData = FirebaseAuth.getInstance().getCurrentUser() ; |
|  |  |
|  | String userid = FirebaseAuth.getInstance().getCurrentUser().getUid(); |
|  |  |
|  | PieChartView pieChartView[] = new PieChartView[3]; |
|  | List<SliceValue> pieData[] = new List[3]; |
|  | PieChartData pieChartData[] = new PieChartData[3]; |
|  | String gas\_n[] = {"TEMP", "CO2", "Humid"}; |
|  |  |
|  | private static final String TAG = "tag me"; |
|  |  |
|  | @Nullable |
|  | @Override |
|  | public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) { |
|  | myView = inflater.inflate(R.layout.activity\_summary, container, false); |
|  |  |
|  | /\* |
|  | tempVal = (EditText) myView.findViewById(R.id.tempVal); |
|  | //vocVal = (EditText) myView.findViewById(R.id.vocVal); |
|  | co2Val = (EditText) myView.findViewById(R.id.co2Val); |
|  | humidVal = (EditText) myView.findViewById(R.id.humidVal); |
|  | btnCData = (Button) myView.findViewById(R.id.btn\_current\_data); |
|  | \*/ |
|  | statText = (TextView) myView.findViewById(R.id.textViewStat); |
|  | tempText = (TextView) myView.findViewById(R.id.textViewTemp); |
|  | humidText = (TextView) myView.findViewById(R.id.textViewHumid); |
|  | co2Text = (TextView) myView.findViewById(R.id.textViewCo2); |
|  |  |
|  | database = FirebaseDatabase.getInstance(); |
|  | //ref = database.getReference(authData.getUid()); |
|  | //ref = database.getReference("Current Data"); |
|  | ref = database.getReference("Data"); // -- -- -- Ref here -- -- -- |
|  |  |
|  | //pieChartView[0] = myView.findViewById(R.id.chart\_temp); |
|  | //pieChartView[1] = myView.findViewById(R.id.chart\_co2); |
|  | //pieChartView[2] = myView.findViewById(R.id.chart\_humid); |
|  |  |
|  | // setListeners(); |
|  |  |
|  | return myView; |
|  | } |
|  | /\* |
|  | private void setListeners() { |
|  | btnCData.setOnClickListener(new View.OnClickListener() { |
|  | @Override |
|  | public void onClick(View v) { |
|  | String id = ref.push().getKey(); |
|  |  |
|  | int temp = Integer.parseInt(tempVal.getText().toString()); |
|  | int co2 = Integer.parseInt(co2Val.getText().toString()); |
|  | int humid = Integer.parseInt(humidVal.getText().toString()); |
|  |  |
|  | //DataValue dataValue = new DataValue(temp, co2, humid); |
|  | // ref.child(userid).child(id).setValue(dataValue); |
|  | } |
|  | }); |
|  | } |
|  | \*/ |
|  |  |
|  | @Override |
|  | public void onStart() { |
|  | super.onStart(); |
|  | //userid |
|  | ref.child("pi").addValueEventListener(new ValueEventListener() { |
|  | @Override |
|  | public void onDataChange(@NonNull DataSnapshot dataSnapshot) { |
|  |  |
|  | for(DataSnapshot myDataSnapshot : dataSnapshot.getChildren()){ |
|  | float temp, humid; |
|  | int eco2; |
|  | try { |
|  | temp = myDataSnapshot.child("temperature").getValue(Float.class); |
|  | humid = myDataSnapshot.child("humidity").getValue(Float.class); |
|  | eco2 = myDataSnapshot.child("eC02/eCO2").getValue(int.class); |
|  |  |
|  | } catch (NullPointerException e) { |
|  | temp = 0; |
|  | humid = 0; |
|  | eco2 = 0; |
|  | } catch (IllegalArgumentException e) { |
|  | temp = 1; |
|  | humid = 1; |
|  | eco2 = 1; |
|  | } |
|  |  |
|  | //Data dataValue = myDataSnapshot.getValue(Data.class); |
|  | //Data myCO2Value = myDataSnapshot.child("eC02").getValue(Data.class); |
|  |  |
|  | tempText.setText("Temp: " + temp + "°C"); |
|  | humidText.setText("Humidity: " + humid + "%"); |
|  | co2Text.setText("Co2: " + eco2 + " ppm"); |
|  |  |
|  | if (eco2 > 8192){ |
|  | co2Text.setText("Co2: Invalid"); |
|  | } |
|  |  |
|  | if (temp == 0 && humid == 0 && eco2 == 0){ |
|  | statText.setText("Database Error"); |
|  | statText.setTextColor(Color.GRAY); |
|  | } |
|  | else if (temp <= 20 ) { |
|  | statText.setText("Turn on heater"); |
|  | statText.setTextColor(Color.RED); |
|  | } |
|  | else if (temp >= 30 ) { |
|  | statText.setText("Turn on fans or open windows"); |
|  | statText.setTextColor(Color.RED); |
|  | } |
|  | else { |
|  | if (humid <= 80) { |
|  | statText.setText("Watering required!!!"); |
|  | statText.setTextColor(Color.RED); |
|  | } else { |
|  | statText.setText("Normal"); |
|  | statText.setTextColor(Color.GREEN); |
|  | } |
|  | } |
|  |  |
|  | } |
|  | } |
|  |  |
|  | @Override |
|  | public void onCancelled(@NonNull DatabaseError databaseError) { |
|  |  |
|  | } |
|  | }); |
|  | } |
|  |  |
|  | } |

### TemperatureFragment.java

|  |
| --- |
|  |
| package com.greenhouseproject; |
|  |  |
|  | import android.graphics.Color; |
|  | import android.os.Bundle; |
|  | import android.support.annotation.NonNull; |
|  | import android.support.annotation.Nullable; |
|  | import android.support.v4.app.Fragment; |
|  | import android.view.LayoutInflater; |
|  | import android.view.View; |
|  | import android.view.ViewGroup; |
|  | import android.widget.ArrayAdapter; |
|  | import android.widget.Button; |
|  | import android.widget.EditText; |
|  | import android.widget.ListView; |
|  |  |
|  | import java.text.NumberFormat; |
|  | import java.text.SimpleDateFormat; |
|  | import java.util.ArrayList; |
|  | import java.util.Date; |
|  | import java.util.List; |
|  |  |
|  | import com.google.firebase.auth.FirebaseAuth; |
|  | import com.google.firebase.auth.FirebaseUser; |
|  | import com.google.firebase.database.DataSnapshot; |
|  | import com.google.firebase.database.DatabaseError; |
|  | import com.google.firebase.database.DatabaseReference; |
|  | import com.google.firebase.database.FirebaseDatabase; |
|  | import com.google.firebase.database.ValueEventListener; |
|  | import com.jjoe64.graphview.DefaultLabelFormatter; |
|  | import com.jjoe64.graphview.GraphView; |
|  | import com.jjoe64.graphview.LegendRenderer; |
|  | import com.jjoe64.graphview.series.DataPoint; |
|  | import com.jjoe64.graphview.series.LineGraphSeries; |
|  |  |
|  | public class TemperatureFragment extends Fragment{ |
|  | View myView; |
|  |  |
|  | String gas\_n = "temperature"; |
|  |  |
|  | String[] mobileArray = {"15","18", "14"}; |
|  |  |
|  | //ArrayAdapter adapter; |
|  | private ListView listView; |
|  |  |
|  | FirebaseDatabase database; |
|  | DatabaseReference ref; |
|  | String userid = FirebaseAuth.getInstance().getCurrentUser().getUid(); |
|  |  |
|  | GraphView graphView; |
|  | LineGraphSeries series = new LineGraphSeries(); |
|  |  |
|  | FirebaseUser authData = FirebaseAuth.getInstance().getCurrentUser() ; |
|  | SimpleDateFormat sdf = new SimpleDateFormat("MMM d\nHH:mm"); |
|  | SimpleDateFormat sdf2 = new SimpleDateFormat("\t\t\t\t MMM d HH:mm"); |
|  | // HH:mm:ss or |
|  |  |
|  | private ArrayList<String> list = new ArrayList<String>(); |
|  | private ArrayAdapter<String> adapter; |
|  |  |
|  | @Nullable |
|  | @Override |
|  | public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) { |
|  | myView = inflater.inflate(R.layout.fragment\_temperature, container, false); |
|  |  |
|  | graphView = (GraphView) myView.findViewById(R.id.graph); |
|  |  |
|  | graphView.addSeries(series); |
|  | series.setTitle(gas\_n); |
|  |  |
|  |  |
|  | adapter = new ArrayAdapter<String>(getActivity(), |
|  | R.layout.activity\_listview, list); |
|  | adapter.notifyDataSetChanged(); |
|  | listView = (ListView) myView.findViewById(R.id.listViewTemp); |
|  |  |
|  | // legend |
|  | graphView.getLegendRenderer().setVisible(true); |
|  | graphView.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.BOTTOM); |
|  |  |
|  | database = FirebaseDatabase.getInstance(); |
|  | ref = database.getReference("Data"); |
|  |  |
|  | graphView.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter() |
|  | { |
|  | @Override |
|  | public String formatLabel(double value, boolean isValueX) { |
|  | if(isValueX) { |
|  | return sdf.format(new Date((long) value \* 1000)); |
|  | } |
|  | return super.formatLabel(value, isValueX); |
|  | } |
|  | }); |
|  |  |
|  |  |
|  | //setListeners(); |
|  |  |
|  | return myView; |
|  | } |
|  |  |
|  | @Override |
|  | public void onStart() { |
|  | super.onStart(); |
|  | ref.child("pi").addValueEventListener(new ValueEventListener() { |
|  | @Override |
|  | public void onDataChange(@NonNull DataSnapshot dataSnapshot) { |
|  | DataPoint[] dp = new DataPoint[(int) dataSnapshot.getChildrenCount()]; |
|  | int index = 0; |
|  |  |
|  | listView.setAdapter(adapter); |
|  | list.clear(); |
|  |  |
|  | for(DataSnapshot myDataSnapshot : dataSnapshot.getChildren()){ |
|  | float temp; |
|  | String seconds; |
|  | try { |
|  | temp = myDataSnapshot.child("temperature").getValue(float.class); |
|  | seconds = myDataSnapshot.child("seconds").getValue(String.class); |
|  |  |
|  | } catch (NullPointerException e) { |
|  | temp = 0; |
|  | seconds = "1550000000"; |
|  | } catch (IllegalArgumentException e) { |
|  | temp = 0; |
|  | seconds = "1550000000"; |
|  | } |
|  |  |
|  | dp[index] = new DataPoint(Double.parseDouble(seconds) ,temp); |
|  |  |
|  | Long l\_date = (long)Double.parseDouble(seconds); |
|  | String mydate = sdf2.format(new Date((l\_date \* 1000))); |
|  |  |
|  | list.add(String.valueOf(temp) + mydate); |
|  | //list.add(String.valueOf(temp)); |
|  |  |
|  |  |
|  | index++; |
|  | } |
|  |  |
|  |  |
|  | series.resetData(dp); |
|  | } |
|  |  |
|  | @Override |
|  | public void onCancelled(@NonNull DatabaseError databaseError) { |
|  |  |
|  | } |
|  | }); |
|  | } |
|  | } |
|  |  |

### HumidityFragment.java

|  |
| --- |
| package com.greenhouseproject; |
|  |  |
|  | import android.graphics.Color; |
|  | import android.os.Bundle; |
|  | import android.support.annotation.NonNull; |
|  | import android.support.annotation.Nullable; |
|  | import android.support.v4.app.Fragment; |
|  | import android.view.LayoutInflater; |
|  | import android.view.View; |
|  | import android.view.ViewGroup; |
|  | import android.widget.ArrayAdapter; |
|  | import android.widget.Button; |
|  | import android.widget.EditText; |
|  | import android.widget.ListView; |
|  |  |
|  | import java.text.NumberFormat; |
|  | import java.text.SimpleDateFormat; |
|  | import java.util.ArrayList; |
|  | import java.util.Date; |
|  | import java.util.List; |
|  |  |
|  | import com.google.firebase.auth.FirebaseAuth; |
|  | import com.google.firebase.auth.FirebaseUser; |
|  | import com.google.firebase.database.DataSnapshot; |
|  | import com.google.firebase.database.DatabaseError; |
|  | import com.google.firebase.database.DatabaseReference; |
|  | import com.google.firebase.database.FirebaseDatabase; |
|  | import com.google.firebase.database.ValueEventListener; |
|  | import com.jjoe64.graphview.DefaultLabelFormatter; |
|  | import com.jjoe64.graphview.GraphView; |
|  | import com.jjoe64.graphview.LegendRenderer; |
|  | import com.jjoe64.graphview.series.DataPoint; |
|  | import com.jjoe64.graphview.series.LineGraphSeries; |
|  |  |
|  | public class HumidityFragment extends Fragment{ |
|  | View myView; |
|  |  |
|  | String gas\_n = "humid"; |
|  |  |
|  | //String[] mobileArray = {"15","18", "14"}; |
|  |  |
|  | //ArrayAdapter adapter; |
|  | private ListView listView; |
|  |  |
|  | FirebaseDatabase database; |
|  | DatabaseReference ref; |
|  | String userid = FirebaseAuth.getInstance().getCurrentUser().getUid(); |
|  | SimpleDateFormat sdf2 = new SimpleDateFormat("\t\t\t\t MMM d HH:mm"); |
|  |  |
|  | GraphView graphView; |
|  | LineGraphSeries series = new LineGraphSeries(); |
|  |  |
|  | FirebaseUser authData = FirebaseAuth.getInstance().getCurrentUser() ; |
|  | SimpleDateFormat sdf = new SimpleDateFormat("MMM d\nHH:mm"); |
|  | // HH:mm:ss or |
|  |  |
|  | ArrayList<String> list = new ArrayList<String>(); |
|  | ArrayAdapter<String> adapter; |
|  |  |
|  | @Nullable |
|  | @Override |
|  | public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) { |
|  | myView = inflater.inflate(R.layout.activity\_humidity\_fragment, container, false); |
|  |  |
|  | graphView = (GraphView) myView.findViewById(R.id.graph); |
|  |  |
|  | graphView.addSeries(series); |
|  | series.setTitle(gas\_n); |
|  |  |
|  |  |
|  | adapter = new ArrayAdapter<String>(getActivity(), |
|  | R.layout.activity\_listview, list); |
|  | adapter.notifyDataSetChanged(); |
|  | listView = (ListView) myView.findViewById(R.id.listViewHumid); |
|  |  |
|  |  |
|  |  |
|  | // legend |
|  | graphView.getLegendRenderer().setVisible(true); |
|  | graphView.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.BOTTOM); |
|  |  |
|  | database = FirebaseDatabase.getInstance(); |
|  | ref = database.getReference("Data"); |
|  |  |
|  | graphView.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter() |
|  | { |
|  | @Override |
|  | public String formatLabel(double value, boolean isValueX) { |
|  | if(isValueX) { |
|  | return sdf.format(new Date((long) value \* 1000)); |
|  | } |
|  | return super.formatLabel(value, isValueX); |
|  | } |
|  | }); |
|  |  |
|  |  |
|  | //setListeners(); |
|  |  |
|  | return myView; |
|  | } |
|  |  |
|  | @Override |
|  | public void onStart() { |
|  | super.onStart(); |
|  | ref.child("pi").addValueEventListener(new ValueEventListener() { |
|  | @Override |
|  | public void onDataChange(@NonNull DataSnapshot dataSnapshot) { |
|  | DataPoint[] dp = new DataPoint[(int) dataSnapshot.getChildrenCount()]; |
|  | int index = 0; |
|  |  |
|  | listView.setAdapter(adapter); |
|  | list.clear(); |
|  |  |
|  | for(DataSnapshot myDataSnapshot : dataSnapshot.getChildren()){ |
|  | float humid; |
|  | String seconds; |
|  | try { |
|  | humid = myDataSnapshot.child("humidity").getValue(float.class); |
|  | seconds = myDataSnapshot.child("seconds").getValue(String.class); |
|  |  |
|  | } catch (NullPointerException e) { |
|  | humid = 0; |
|  | seconds = "1550000000"; |
|  | } catch (IllegalArgumentException e) { |
|  | humid = 0; |
|  | seconds = "1550000000"; |
|  | } |
|  |  |
|  | dp[index] = new DataPoint(Double.parseDouble(seconds) ,humid); |
|  |  |
|  | Long l\_date = (long)Double.parseDouble(seconds); |
|  | String mydate = sdf2.format(new Date((l\_date \* 1000))); |
|  |  |
|  | list.add(String.valueOf(humid) + mydate); |
|  | //list.add(String.valueOf(humid)); |
|  |  |
|  | index++; |
|  | } |
|  |  |
|  |  |
|  | series.resetData(dp); |
|  | } |
|  |  |
|  | @Override |
|  | public void onCancelled(@NonNull DatabaseError databaseError) { |
|  |  |
|  | } |
|  | }); |
|  | } |
|  | } |

### eCO2Fragment.java

|  |
| --- |
| package com.greenhouseproject; |
|  |  |
|  | import android.content.pm.PackageManager; |
|  | import android.os.Bundle; |
|  | import android.support.annotation.NonNull; |
|  | import android.support.annotation.Nullable; |
|  | import android.support.v4.app.Fragment; |
|  | import android.view.LayoutInflater; |
|  | import android.view.View; |
|  | import android.view.ViewGroup; |
|  | import android.widget.ArrayAdapter; |
|  | import android.widget.ListView; |
|  |  |
|  | import java.text.SimpleDateFormat; |
|  | import java.util.ArrayList; |
|  | import java.util.Date; |
|  | import java.util.HashMap; |
|  |  |
|  | import com.google.firebase.auth.FirebaseAuth; |
|  | import com.google.firebase.auth.FirebaseUser; |
|  | import com.google.firebase.database.DataSnapshot; |
|  | import com.google.firebase.database.DatabaseError; |
|  | import com.google.firebase.database.DatabaseReference; |
|  | import com.google.firebase.database.FirebaseDatabase; |
|  | import com.google.firebase.database.ValueEventListener; |
|  | import com.jjoe64.graphview.DefaultLabelFormatter; |
|  | import com.jjoe64.graphview.GraphView; |
|  | import com.jjoe64.graphview.LegendRenderer; |
|  | import com.jjoe64.graphview.series.DataPoint; |
|  | import com.jjoe64.graphview.series.LineGraphSeries; |
|  |  |
|  | public class eCO2Fragment extends Fragment{ |
|  | View myView; |
|  |  |
|  | String gas\_n = "eCO2"; |
|  |  |
|  | String[] mobileArray = {"15","18", "14"}; |
|  |  |
|  | //ArrayAdapter adapter; |
|  | private ListView listView; |
|  |  |
|  | FirebaseDatabase database; |
|  | DatabaseReference ref; |
|  | String userid = FirebaseAuth.getInstance().getCurrentUser().getUid(); |
|  |  |
|  | GraphView graphView; |
|  | LineGraphSeries series = new LineGraphSeries(); |
|  |  |
|  | FirebaseUser authData = FirebaseAuth.getInstance().getCurrentUser() ; |
|  | SimpleDateFormat sdf = new SimpleDateFormat("MMM d\nHH:mm"); |
|  | SimpleDateFormat sdf2 = new SimpleDateFormat("\t\t\t\t MMM d HH:mm"); |
|  | // HH:mm:ss or |
|  |  |
|  | private ArrayList<String> list = new ArrayList<String>(); |
|  | private ArrayAdapter<String> adapter; |
|  |  |
|  | @Nullable |
|  | @Override |
|  | public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) { |
|  | myView = inflater.inflate(R.layout.fragment\_e\_co2, container, false); |
|  |  |
|  | graphView = (GraphView) myView.findViewById(R.id.graph); |
|  |  |
|  | graphView.addSeries(series); |
|  | series.setTitle(gas\_n); |
|  |  |
|  | //final PackageManager packageManager = getActivity().getPackageManager(); |
|  | adapter = new ArrayAdapter<String>(getActivity(), |
|  | R.layout.activity\_listview, list); |
|  | adapter.notifyDataSetChanged(); |
|  | listView = (ListView) myView.findViewById(R.id.listViewECO2); |
|  |  |
|  |  |
|  |  |
|  | // legend |
|  | graphView.getLegendRenderer().setVisible(true); |
|  | graphView.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.BOTTOM); |
|  |  |
|  | database = FirebaseDatabase.getInstance(); |
|  | ref = database.getReference("Data"); |
|  |  |
|  | graphView.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter() |
|  | { |
|  | @Override |
|  | public String formatLabel(double value, boolean isValueX) { |
|  | if(isValueX) { |
|  | return sdf.format(new Date((long) value \* 1000)); |
|  | } |
|  | return super.formatLabel(value, isValueX); |
|  | } |
|  | }); |
|  |  |
|  |  |
|  | //setListeners(); |
|  |  |
|  | return myView; |
|  | } |
|  |  |
|  | @Override |
|  | public void onStart() { |
|  | super.onStart(); |
|  | ref.child("pi").addValueEventListener(new ValueEventListener() { |
|  | @Override |
|  | public void onDataChange(@NonNull DataSnapshot dataSnapshot) { |
|  | DataPoint[] dp = new DataPoint[(int) dataSnapshot.getChildrenCount()]; |
|  | int index = 0; |
|  |  |
|  | listView.setAdapter(adapter); |
|  | list.clear(); |
|  |  |
|  | for(DataSnapshot myDataSnapshot : dataSnapshot.getChildren()){ |
|  | int eco2; |
|  | String seconds; |
|  | try { |
|  | eco2 = myDataSnapshot.child("eC02/eCO2").getValue(int.class); |
|  | seconds = myDataSnapshot.child("seconds").getValue(String.class); |
|  |  |
|  | } catch (NullPointerException e) { |
|  | eco2 = 0; |
|  | seconds = "1550000000"; |
|  | } catch (IllegalArgumentException e) { |
|  | eco2 = 0; |
|  | seconds = "1550000000"; |
|  | } |
|  |  |
|  | //Data dataValue = myDataSnapshot.getValue(Data.class); |
|  | //Data myCO2Value = myDataSnapshot.child("eC02").getValue(Data.class); |
|  |  |
|  | dp[index] = new DataPoint(Double.parseDouble(seconds) ,eco2); |
|  |  |
|  | Long l\_date = (long)Double.parseDouble(seconds); |
|  | String mydate = sdf2.format(new Date((l\_date \* 1000))); |
|  |  |
|  | list.add(String.valueOf(eco2) + mydate); |
|  |  |
|  | //dp[index] = new DataPoint(index+1,(int) dataValue.getHumidity()); |
|  |  |
|  | index++; |
|  | } |
|  |  |
|  |  |
|  | series.resetData(dp); |
|  | } |
|  |  |
|  | @Override |
|  | public void onCancelled(@NonNull DatabaseError databaseError) { |
|  |  |
|  | } |
|  | }); |
|  | } |
|  | } |

### HistoryFragment.java

|  |
| --- |
| package com.greenhouseproject; |
|  |  |
|  | import android.graphics.Color; |
|  | import android.os.Bundle; |
|  | import android.support.annotation.NonNull; |
|  | import android.support.annotation.Nullable; |
|  | import android.support.v4.app.Fragment; |
|  | import android.view.LayoutInflater; |
|  | import android.view.View; |
|  | import android.view.ViewGroup; |
|  | import android.widget.Button; |
|  | import android.widget.EditText; |
|  |  |
|  | import java.text.NumberFormat; |
|  | import java.text.SimpleDateFormat; |
|  | import java.util.Date; |
|  |  |
|  | import com.google.firebase.auth.FirebaseAuth; |
|  | import com.google.firebase.auth.FirebaseUser; |
|  | import com.google.firebase.database.DataSnapshot; |
|  | import com.google.firebase.database.DatabaseError; |
|  | import com.google.firebase.database.DatabaseReference; |
|  | import com.google.firebase.database.FirebaseDatabase; |
|  | import com.google.firebase.database.ValueEventListener; |
|  | import com.jjoe64.graphview.DefaultLabelFormatter; |
|  | import com.jjoe64.graphview.GraphView; |
|  | import com.jjoe64.graphview.LegendRenderer; |
|  | import com.jjoe64.graphview.series.DataPoint; |
|  | import com.jjoe64.graphview.series.LineGraphSeries; |
|  |  |
|  | public class HistoryFragment extends Fragment{ |
|  | View myView; |
|  |  |
|  | String gas\_n[] = {"temp", "co2", "humid"}; |
|  |  |
|  | FirebaseDatabase database; |
|  | DatabaseReference ref; |
|  | String userid = FirebaseAuth.getInstance().getCurrentUser().getUid(); |
|  |  |
|  | GraphView graphView; |
|  | LineGraphSeries series[] = {new LineGraphSeries(), new LineGraphSeries(), new LineGraphSeries()}; |
|  |  |
|  | FirebaseUser authData = FirebaseAuth.getInstance().getCurrentUser() ; |
|  | SimpleDateFormat sdf = new SimpleDateFormat("MMM d\nHH:mm"); |
|  | // HH:mm:ss or |
|  |  |
|  | @Nullable |
|  | @Override |
|  | public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) { |
|  | myView = inflater.inflate(R.layout.activity\_history, container, false); |
|  |  |
|  | graphView = (GraphView) myView.findViewById(R.id.graph); |
|  |  |
|  | for (int i=0; i < 3; i++) { |
|  | graphView.addSeries(series[i]); |
|  | series[i].setTitle(gas\_n[i]); |
|  | } |
|  | series[1].setColor(Color.GREEN); |
|  | series[2].setColor(Color.RED); |
|  | //series[3].setColor(Color.MAGENTA); |
|  |  |
|  | // legend |
|  | graphView.getLegendRenderer().setVisible(true); |
|  | graphView.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.BOTTOM); |
|  |  |
|  | database = FirebaseDatabase.getInstance(); |
|  | ref = database.getReference("Current Data"); |
|  |  |
|  | graphView.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter() |
|  | { |
|  | @Override |
|  | public String formatLabel(double value, boolean isValueX) { |
|  | if(isValueX) { |
|  | return sdf.format(new Date((long) value)); |
|  | } |
|  | return super.formatLabel(value, isValueX); |
|  | } |
|  | }); |
|  |  |
|  | //setListeners(); |
|  |  |
|  | return myView; |
|  | } |
|  |  |
|  | @Override |
|  | public void onStart() { |
|  | super.onStart(); |
|  | ref.child(userid).addValueEventListener(new ValueEventListener() { |
|  | @Override |
|  | public void onDataChange(@NonNull DataSnapshot dataSnapshot) { |
|  | DataPoint[][] dp = new DataPoint[3][(int) dataSnapshot.getChildrenCount()]; |
|  |  |
|  | int index = 0; |
|  | for(DataSnapshot myDataSnapshot : dataSnapshot.getChildren()){ |
|  | DataValue dataValue = myDataSnapshot.getValue(DataValue.class); |
|  | int ValArr[] = {dataValue.getTempValue(),dataValue.getCo2Value(),dataValue.getHumidityValue()}; |
|  |  |
|  | for (int i=0; i < 3; i++) { |
|  | dp[i][index] = new DataPoint(dataValue.getDate(),ValArr[i]); |
|  | } |
|  | index++; |
|  | } |
|  |  |
|  | for (int i=0; i < 3; i++) { |
|  | series[i].resetData(dp[i]); |
|  | } |
|  | } |
|  |  |
|  | @Override |
|  | public void onCancelled(@NonNull DatabaseError databaseError) { |
|  |  |
|  | } |
|  | }); |
|  | } |
|  | } |

## Layouts

### fragment\_temperature.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> |
|  | <RelativeLayout |
|  | xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent" |
|  | android:layout\_height="match\_parent"> |
|  |  |
|  | <com.jjoe64.graphview.GraphView |
|  | android:id="@+id/graph" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="260dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_below="@+id/textTitleT" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textTitleT" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_centerHorizontal="true" |
|  | android:text="Temperture" |
|  | android:textSize="25sp" /> |
|  |  |
|  | <ListView |
|  | android:id="@+id/listViewTemp" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="350dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_below="@+id/graph" /> |
|  |  |
|  | </RelativeLayout> |

### fragment\_humidity.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> |
|  | <RelativeLayout |
|  | xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent" |
|  | android:layout\_height="match\_parent"> |
|  |  |
|  | <com.jjoe64.graphview.GraphView |
|  | android:id="@+id/graph" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="260dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_below="@+id/textTitleH" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textTitleH" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_centerHorizontal="true" |
|  | android:text="Humidity" |
|  | android:textSize="25sp" /> |
|  |  |
|  | <ListView |
|  | android:id="@+id/listViewHumid" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="350dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_below="@+id/graph" /> |
|  |  |
|  | </RelativeLayout> |

### fragment\_e\_co2.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> |
|  | <RelativeLayout |
|  | xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent" |
|  | android:layout\_height="match\_parent"> |
|  |  |
|  | <com.jjoe64.graphview.GraphView |
|  | android:id="@+id/graph" |
|  | android:layout\_width="401dp" |
|  | android:layout\_height="260dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_below="@+id/textTitleE" |
|  | android:layout\_marginStart="20dp" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textTitleE" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_centerHorizontal="true" |
|  | android:text="eCO2" |
|  | android:textSize="25sp" /> |
|  |  |
|  | <ListView |
|  | android:id="@+id/listViewECO2" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="350dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_below="@+id/graph" /> |
|  |  |
|  | </RelativeLayout> |

### activity\_summary.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> |
|  | <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" |
|  | xmlns:app="http://schemas.android.com/apk/res-auto" |
|  | xmlns:tools="http://schemas.android.com/tools" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="match\_parent" |
|  | tools:context=".MainActivity"> |
|  |  |
|  | <android.support.design.widget.BottomNavigationView |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignParentBottom="true" |
|  | android:layout\_alignParentStart="true" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textView8" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_centerHorizontal="true" |
|  | android:layout\_marginTop="20dp" |
|  | android:text="Current Data" |
|  | android:textSize="30sp" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textViewStat" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignParentBottom="true" |
|  | android:layout\_centerHorizontal="true" |
|  | android:layout\_marginBottom="50dp" |
|  | android:text="Normal" |
|  | android:textSize="25sp" /> |
|  |  |
|  |  |
|  | <ImageView |
|  | android:id="@+id/imageTemp" |
|  | android:layout\_width="100dp" |
|  | android:layout\_height="100dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_marginStart="35dp" |
|  | android:layout\_marginTop="125dp" |
|  | app:srcCompat="@drawable/temperature" /> |
|  |  |
|  | <ImageView |
|  | android:id="@+id/imageHumid" |
|  | android:layout\_width="100dp" |
|  | android:layout\_height="100dp" |
|  | android:layout\_alignStart="@+id/imageTemp" |
|  | android:layout\_below="@+id/imageTemp" |
|  | app:srcCompat="@drawable/drop" /> |
|  |  |
|  | <ImageView |
|  | android:id="@+id/imageCarbon2" |
|  | android:layout\_width="100dp" |
|  | android:layout\_height="100dp" |
|  | android:layout\_alignStart="@+id/imageTemp" |
|  | android:layout\_below="@+id/imageHumid" |
|  | app:srcCompat="@drawable/carbondioxide" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textViewTemp" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignBottom="@+id/imageTemp" |
|  | android:layout\_toEndOf="@+id/imageTemp" |
|  | android:text="Temp: " |
|  | android:textSize="25dp"/> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textViewHumid" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignBottom="@+id/imageHumid" |
|  | android:layout\_toEndOf="@+id/imageTemp" |
|  | android:text="Humidity: " |
|  | android:textSize="25dp" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textViewCo2" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignBottom="@+id/imageCarbon2" |
|  | android:layout\_toEndOf="@+id/imageTemp" |
|  | android:text="Co2: " |
|  | android:textSize="25dp" /> |
|  |  |
|  | </RelativeLayout> |

### activity\_history.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> |
|  | <RelativeLayout |
|  | xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent" |
|  | android:layout\_height="match\_parent"> |
|  |  |
|  | <com.jjoe64.graphview.GraphView |
|  | android:id="@+id/graph" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="375dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_marginTop="70dp" /> |
|  |  |
|  | <TextView |
|  | android:id="@+id/textView9" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_centerHorizontal="true" |
|  | android:layout\_marginTop="20dp" |
|  | android:text="Hourly Data" |
|  | android:textSize="30sp" /> |
|  |  |
|  | <!-- <com.github.mikephil.charting.charts.LineChart |
|  | android:id="@+id/lineChart" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="318dp" |
|  | android:layout\_alignParentStart="true" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_marginTop="147dp"></com.github.mikephil.charting.charts.LineChart> |
|  | --> |
|  |  |
|  | </RelativeLayout> |

# Hardware Integrated

## Python Script

### greenhouse2.py

|  |
| --- |
| import datetime |
|  | import time |
|  | from tentacle\_pi.AM2315 import AM2315 |
|  | from CCS811\_RPi import CCS811\_RPi |
|  |  |
|  | import Adafruit\_GPIO.SPI as SPI |
|  | import Adafruit\_SSD1306 |
|  |  |
|  | from PIL import Image |
|  | from PIL import ImageDraw |
|  | from PIL import ImageFont |
|  |  |
|  | import firebase\_admin |
|  | from firebase\_admin import credentials |
|  | from firebase\_admin import db |
|  | cred = credentials.Certificate('/home/pi/GreenhouseMonitoringSystem/greenhouseproject.json') |
|  | default\_app = firebase\_admin.initialize\_app(cred, { |
|  | 'databaseURL':'https://greenhouseproject-58231.firebaseio.com' |
|  | }) |
|  | ref = db.reference('Data') |
|  |  |
|  |  |
|  | am = AM2315(0x5c,"/dev/i2c-1") |
|  | ccs811 = CCS811\_RPi() |
|  |  |
|  | ''' |
|  | MEAS MODE REGISTER AND DRIVE MODE CONFIGURATION |
|  | 0b0 Idle (Measurements are disabled in this mode) |
|  | 0b10000 Constant power mode, IAQ measurement every second |
|  | 0b100000 Pulse heating mode IAQ measurement every 10 seconds |
|  | 0b110000 Low power pulse heating mode IAQ measurement every 60 |
|  | 0b1000000 Constant power mode, sensor measurement every 250ms |
|  | ''' |
|  | # Set MEAS\_MODE (measurement interval) |
|  | configuration = 0b100000 |
|  |  |
|  | # Set read interval for retriveving last measurement data from the sensor |
|  | pause = 10 |
|  |  |
|  | hwid = ccs811.checkHWID() |
|  | if(hwid == hex(129)): |
|  | print 'Hardware ID is correct' |
|  | else: print 'Incorrect hardware ID ' ,hwid, ', should be 0x5B' |
|  |  |
|  | ccs811.configureSensor(configuration) |
|  | print '--------------------------------' |
|  |  |
|  | # Raspberry Pi pin configuration: |
|  | RST = 24 |
|  | # Note the following are only used with SPI: |
|  | DC = 23 |
|  | SPI\_PORT = 0 |
|  | SPI\_DEVICE = 0 |
|  |  |
|  | # 128x32 display with hardware I2C: |
|  | disp = Adafruit\_SSD1306.SSD1306\_128\_32(rst=RST, i2c\_address=0x3C) |
|  |  |
|  | # Initialize library. |
|  | disp.begin() |
|  |  |
|  | # Clear display. |
|  | disp.clear() |
|  | disp.display() |
|  |  |
|  | # Create blank image for drawing. |
|  | # Make sure to create image with mode '1' for 1-bit color. |
|  | width = disp.width |
|  | height = disp.height |
|  | image = Image.new('1', (width, height)) |
|  |  |
|  | # Get drawing object to draw on image. |
|  | draw = ImageDraw.Draw(image) |
|  |  |
|  | # Draw a black filled box to clear the image. |
|  | draw.rectangle((0,0,width,height), outline=0, fill=0) |
|  |  |
|  | # Draw some shapes. |
|  | # First define some constants to allow easy resizing of shapes. |
|  | padding = -2 |
|  | top = padding |
|  | bottom = height-padding |
|  | # Move left to right keeping track of the current x position for drawing shapes. |
|  | x = 0 |
|  |  |
|  | # Load default font. |
|  | font = ImageFont.load\_default() |
|  | user\_ref = ref.child('users') |
|  | user\_ref.set({ |
|  | 'pi':{ |
|  | 'full\_name' : 'Raspberry Pi', |
|  | 'Temperature' : '0', |
|  | 'Humidity' : '0', |
|  | 'eC02' : '0' |
|  | } |
|  | }) |
|  | while True: |
|  | epochtime = time.time() |
|  | localtime = epochtime - 240 #coverts UTC to UTC-4 |
|  | temperature, humidity, crc\_check = am.sense() |
|  | result = ccs811.readAlg(); |
|  | posts\_ref = ref.child('pi') |
|  | new\_post\_ref = posts\_ref.push() |
|  | new\_post\_ref.set({ |
|  | 'temperature' : temperature, |
|  | 'humidity' : humidity, |
|  | 'eC02' : result, |
|  | 'seconds' : str(localtime), |
|  | }) |
|  | #str\_co2 = 'eCO2: ', result['eCO2'], ' ppm' |
|  | #str\_temp = 'temperature: %0.1f' % temperature |
|  | #str\_hum = 'humidity: %0.1f' % humidity |
|  |  |
|  | draw.rectangle((0,0,width,height), outline=0, fill=0) |
|  | if(result['eCO2'] > 8192): |
|  | draw.text((x, top), "eCO2: "+ "Invalid", font=font, fill=255) |
|  | else: |
|  | # Write two lines of text. |
|  | draw.text((x, top), "eCO2: "+ str(result['eCO2']) + " ppm", font=font, fill=255) |
|  |  |
|  | draw.text((x, top+8), "temperature: %0.1f" % temperature, font=font, fill=255) |
|  | draw.text((x, top+16), "humidity: %0.1f" % humidity, font=font, fill=255) |
|  |  |
|  | # Display image. |
|  | disp.image(image) |
|  | disp.display() |
|  |  |
|  | time.sleep(pause) |