AIR QUALITY MONITORING SYSTEM BASED ON SENSOR NETWORK

By

Ke Liu

Yuhang Ren

Yuhui Xu

(Alpha order)

Final Report for ECE 445, Senior Design, [Spring 2020]

TA: Rui Lu

28th May 2020

Project No. 8

Abstract

Air pollution is a serious problem in modern society, and everyone want to protect themselves from air pollutions. It is beneficial for our health if we can get alarm when the concentration of air pollutants in our living or working area is too high. In this project, we use IoT technique to develop an air quality monitoring based on sensor network covering the whole Zhejiang University International Campus. The self-designed measure devices are placed in different places in the campus, collect data including real-time temperature, humidity, pressure, the concentration of CO and NO_2, and send these data to cloud server through Wi-Fi. The cloud computing server manages data and generates real-time air quality maps for all specified quantities. Students can go to project website to find real-time air-quality maps as well as the prediction maps in the next few minutes.

Contents

1. Introduction

Air quality is a serious problem in modern society, as people lay more emphasis on improving their life quality and living in a good environment. As students from ZJUI institute, we are not used to some strange smell that we can sometimes sense on campus, even though we have lived for about four years in Zhejiang University International campus. We are looking forward to having a monitoring system that can give us alarm if the concentration of pollutants in the campus is too high.

In this project, we built an air quality monitoring system containing self-designed measure devices, cloud servers and database, and websites used as user interface. The architecture of our system is shown in Figure 1. The measure devices are designed to collect real-time air quality data in the outdoor environment and should be able to work for five days without changing its battery. They contain sensors that can measure PM1, PM2.5, PM10, CO, NO2, TVOC, temperature, humidity and pressure. They also contain a Wi-Fi module, a battery module, a MCU module as well as some waterproof design. The cloud server stores and manage air quality data collected by the measure devices and run algorithm to generate air quality map of different air pollutants and quantities. Two websites are built for users to get access to the data stored in cloud server. One is for students, who are allowed to view the real-time air quality map as well as prediction map in these websites. Students and stuffs can also download air quality data from this website. The other website is designed for debugging personnel, who can check the status of all measure devices, like the battery storage and GPS location.

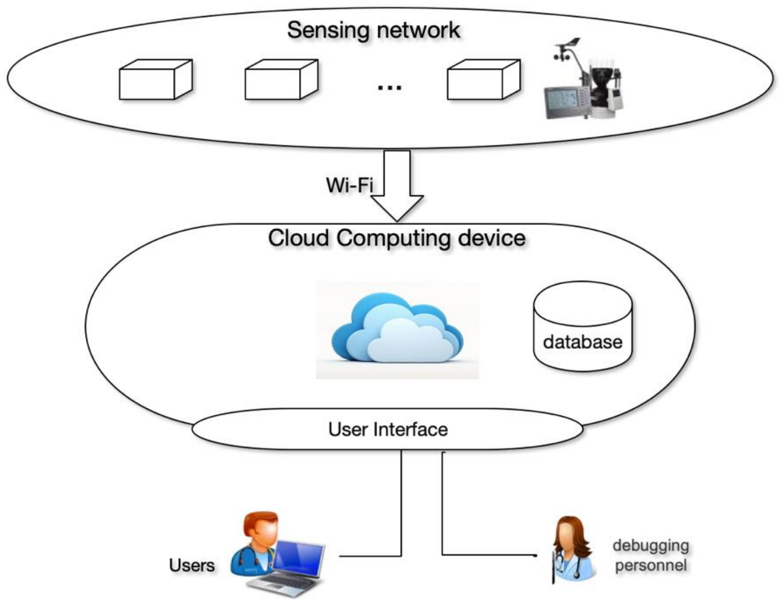


Figure 1: Architecture Overview

2 Design

The overall design includes hardware designs which collect environmental data and upload them to a server, a sensor network design which integrates the devices and a server to form a sensing network, software designs which extract and process the data, and algorithm designs which use environmental data to generate estimation maps and prediction maps of different air quality quotients.

2.1 Sensor Network Design

The sensor network design is the IoT structure that integrates hardware and software together to form a working platform of air quality monitoring system. The sensor network could be simply regarded as a set of self-designed measuring devices plus a weather station, which are connected to the server through Wi-Fi.

2.1.1 Self-designed Measurement Devices

In our implementation of the sensor network aimed for the campus, we se 9 self-designed measuring devices which can measure the following environmental quantities: PM1.0, PM2.5, PM10, CO, NO2, temperature, humidity, and pressure. Seven of them are for map generation and 2 of them are for validation of the maps generated.

Due to the complex environment of the campus including buildings, lawns, lakes, and streets, we need several requirements of the locations to implement the sensors. The requirements include but are not limited to:

1) The places should guarantee stable Wi-Fi connection
2) The sensors should be placed in corners where few people visit, so that the sensors are not likely to be disturbed by passers-by.
3) The sensors should be set to places that are open to air flow and GPS localization.
4) It would be better to put the sensors under shade so that the sun would not heat up the sensor which would give inaccurate measurements of temperatures.

2.1.2 Davis Weather Station

We employed a commercial weather station for the measurement of wind speed and wind direction. The location we put the weather station is on the roof top platform of Block 4 at No. 1 Residential College of the campus.

2.2 Measure Device Design

A measure device is composed of four parts: sensor set, communication tools, control units, and power supply. All four parts work together to provide local environmental data of an outdoor place and upload the data to the server. This section introduces the modules and sensors we use in our PCB on the aspect of hardware implementation, but how to program these sensors would be included in the section of Software Design.

2.2.1 Sensor Set

The sensor set contains all the sensor except the weather station which is a mature commercial device and excluded from our self-designed measure device. The employed sensors are: BME 680, MiCS 6814, and PMS 5003.

BME 680 module

BME 680 is the most complicated sensor in our device. It is capable of measuring general air quality quotients including temperature, pressure, and humidity, as well as specified parameters including CO2 and TVOC. However, CO2 and TVOC are not employed in our design, since the two values are unstable and use the device's own closed-source algorithms where we could not debug. We use BME 680 modules sold by Taobao providers rather than the sensor itself. The module contains 4 more pins besides VCC and GND: SCL, SDA, SDO, AND CS. The module has two programming interfaces: SPI and I2C. We choose I2C. The SCL pin is used as SCK pin in I2C programming, SDA for SDI and SDO is still for SDO in I2C.

MiCS 6814 module

MiCS 6814 is designed for the measurement of NO_2, CO, and NH_3. Because we lack sources for validation of NH_3 data, we neglected this parameter. The output signals CO and NO2 are analog signals, so they are connected to analog pins of our MCU. And the output pins also need pull-up resisters to function correctly, the details of which are specified in Software Design section.

PMS 5003 sensor

PMS 5003 is a commercial sensor for the measurement of PM1.0, PM2.5, and PM10. It uses UART interface (namely, serial port) and sends 32-byte packets on each measurement.

2.2.2 Communication Tools

We use ATGM336H GPS module for localization of different devices, and ESP8266 (esp01) Wi-Fi module for communications of all devices with the server.

2.2.3 Power Supply

For each of the self-designed measure device, we use two 18650 batteries as the power supply. Each battery has a 3.7 voltage and a 3350mAh capacity.

However, 3.7 volts is not suitable, as power supply demands 5 volts or 3.3 volts. We use a power supply module to transform the voltage from ~3.7V to 5V.

We also divide the battery voltage by two equal resistors, and input half of the battery voltage to the MCU for measurement of remaining power.

2.2.4 Control Units

We use STM32 module as the MCU that controls other modules. Signal pins are connected to pins of STM32 for control purposes. Programming details are included in the Software Design section.

In addition, for the modules that do not have a sleep mode and thus require the cutoff of power to switch off, we use the combination of MOSFET and BJT as a switch for those modules. The schematic for the switch is included in Figure 2. An input control signal means switch-off if it is high, and switch-on if it is low (active low).

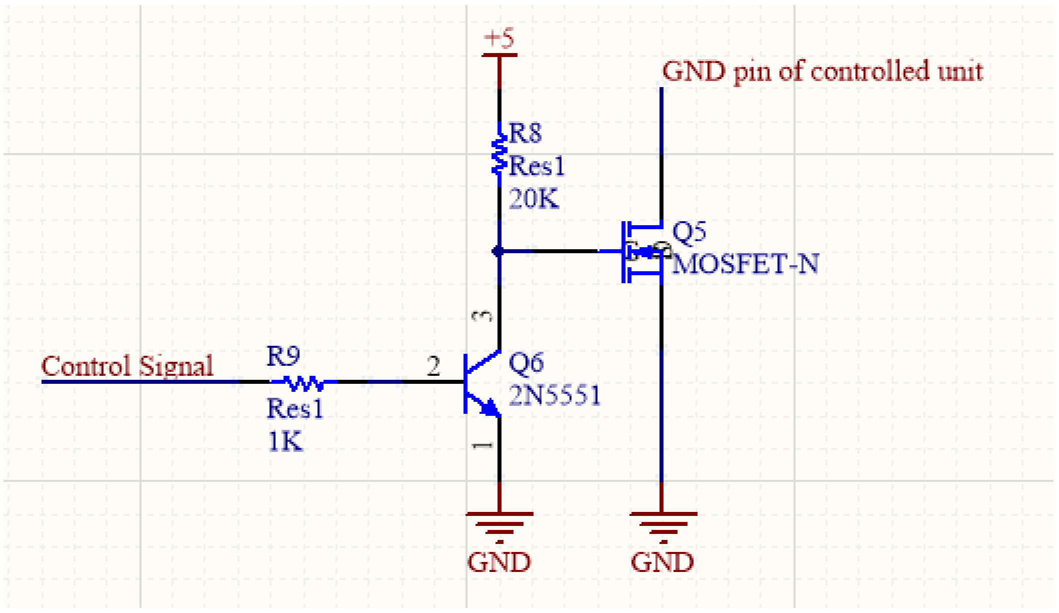


Figure 2: Implementation of the MOSFET-BJT combined switch.

2.2.5 Casing

The self-designed measure device is cased in an acrylic-plate-made box. The two bars connecting two sides of the box make the support of the PCB containing electrical components which is fixed on the bars by several screws. The other two sides of the box are window-shade like to prevent rain from entering while keeping airflow. For better waterproof, the gaps where acrylic plates are connected can be sealed with glue. PCBs are also painted with water-resistant paint for waterproof.

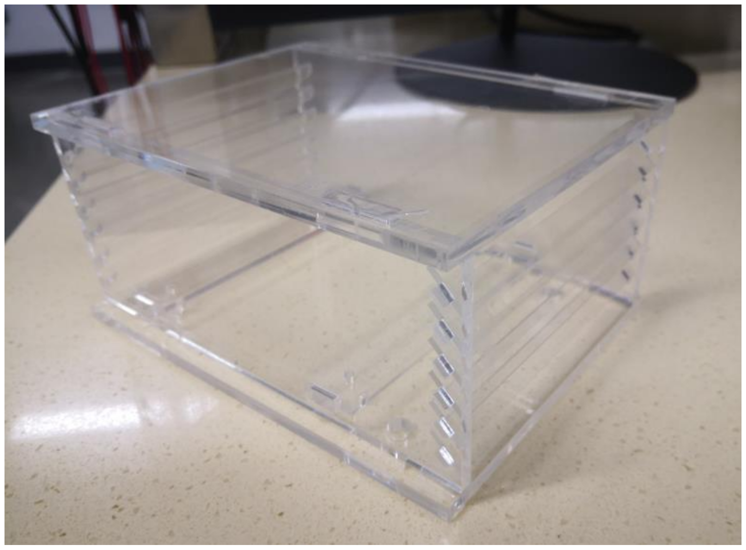


Figure 3: Casing design of self-designed measure devices.

2.3 Algorithm Design

The generation of estimation maps uses input values of measuring devices, calculates the weights of each value, and finds the estimated value at a specified location. The method for calculating the weights is as follows:

$$Estimated\ value(x) = \sum_i^n w_{\mathrm{i}} v_i, \tag{2.3.1}$$

where $w_i = \frac{\frac{1}{di}}{\left(\frac{1}{d1}+\frac{1}{d2}+\cdots\right)}$, di is the distance between point i and point x, and v_i is the concentration/value of sample point i.

Prediction maps are based on the assumption that buildings have no effect on wind flow, in which case we can use air diffusion and wind speed & direction to make a prediction directly without considerations for buildings. Air diffusion in two dimensions follows the equation

$$\frac{\partial C}{\partial t} = u\frac{\partial^2 C}{\partial x^2} + u\frac{\partial^2 C}{\partial y^2}, \tag{2.3.2}$$

where C means the concentration of a specified substance or a collection of substances, t stands for unit time, and u stands for diffusion coefficient. The diffusion equation is approximately solved by [4]

$$C_{i,j}^{n+1} = C_{i,j}^{n} + \frac{\mathrm{u}\Delta t}{\Delta x2}\left(C_{i+1,j}^{n} - 2C_{i,j}^{n} + C_{i-1,j}^{n}\right) + \frac{\mathrm{u}\Delta t}{\Delta y2}\left(C_{i,j+1}^{n} - 2C_{i,j}^{n} + C_{i,j-1}^{n}\right). \tag{2.3.3}$$

Wind speed and wind direction affects the diffusion following such equation:

$$C_{\vec{p}}^{n+1} = C_{\vec{p}+\vec{d}}^{n}, \tag{2.3.4}$$

where $\vec{d}$ can be computed based on speed and direction.

2.4 Software Design

The software design in this project mainly consists of controlling programs running on the MCU and prediction and visualization algorithm running on the cloud server hosting the website. The program on the MCU is in charge of periodically switching on the sensor modules to collect data from them, parse such data, format the data into an ASCII text based InfluxDB write request, and send the request with the Wi-Fi module to the cloud server hosting the InfluxDB instance. Afterwards, the MCU will turn off all external sensors and communication modules and enter a sleep period of 15 minutes, and then start a new cycle from collecting data from sensors.

The prediction and visualization algorithm needs to read the latest sensor measurements from the InfluxDB as well as the current wind conditions from the Davis weather station, perform map generation and prediction using the algorithm in 2.3, and upload the generated images to web server.

2.4.1 MCU Software

The program on the MCU is in charge of periodically switching on the sensor modules to collect data from them, optionally after a 30 seconds delay for some sensors to start up, including MiCS 6814 which needs to be heated before producing meaningful output, PMS 5003 with an integrated fan, and ATGM336H GPS module to search for satellite signal.

Then, the program needs to parse the output of these modules, format the data into an ASCII text based InfluxDB write request, and send the request with the Wi-Fi module to the cloud server hosting InfluxDB.

The ATGM336H GPS module produces NMEA0183-standard formatted strings as its output, like most GPS modules on the market. Each line of NMEA0183 message is a comma-delimited string which also contains a 6-character header that indicates the type of message. Since for this project the GPS module is only used for locating the measurement device, we only care about messages starting with "$GNGGA", which indicates that this line contains latitude and longitude information. Such a message is formatted as below:

```
$GNGGA,hhmmss.sss,llll.lllll,a,yyyyy.yyyyy,a,x,xx,x.x,xx.x,M,x.
x,M,x.x,xxxx*hh

$GNGGA,074023.000,3005.84164,N,11955.87563,E,1,13,0.9,32.3,M,0.
0,M,,*46
```

To implement the functionality of GPS locating, only column 3-6 in this comma-delimited string are needed, each representing latitude, north/south, longitude, west/east in order. Note that latitude and longitude are represented in degree-minute format, for example 3005.84164 represents a latitude of 30 degrees and 5.84164 minutes, so the data needs conversion on mapping on the webpage.

Due to the 64KB ROM size limit of our STM32 MCU, the fact that we used most of the ROM size (63.5KB) in the final version of program, and that InfluxDB accepts values in forms of ASCII texts instead of binaries, instead of doing a string-to-floating-point conversion whose relevant code may need a few kilobytes of ROM space, we decided to copy the strings from the NMEA0183 message as-is, which only needs a few hundred bytes for string splitting and memory copy functions, and such functions can be reused when interfacing with other sensors.

The MiCS 6814 sensor represents its reading with the resistance on its output port to the ground, so a pull-up resistor connects the MiCS 6814's output to the 3.3V rail, and the MCU reads in the divided voltage of MiCS 6814 and the pull-up resistor with ADC. The resistance of MiCS 6814 is calculated as follows:

$$R_{MICS6814} = \frac{V_{IN}}{3.3V - V_{IN}} * R_{PULL-UP} \qquad (2.4.1)$$

The resistance is then mapped to a numeric reading using information from the relationship graph on MiCS 6814's datasheet.

The PMS 5003 sends its result as 32-byte packets over the UART interface, and each packet contains a fixed header, readings, and a checksum. The MCU verifies each packet to ensure the data integrity and discards any data that failed the checksum test.

The BME 680 represents its data in I^{2}C registers, and Bosch's BSEC library is used to convert register values to actual readings, as well as performing temperature calibration.

In addition, the MCU will also measure three device health related metrics to ensure it properly functions. The positive lead of the battery is connected to the ADC through a voltage divider, so the battery voltage can be read and calculated, and maintenance personnel can be alerted to replace batteries if the voltage is too low. The MCU will also measure the temperature of its chip and the voltage on the 3.3V rail, so that personnel can be alarmed if such metrics are abnormal and a chip damage may happen.

After collecting data from various sensors, the MCU turns on the Wi-Fi module by sending an active low signal to Wi-Fi module's enable pin and send commands to connect to Zhejiang University's ZJUWLAN network. Then, a HTTP request is sent to ZJUWLAN's web authentication portal server, which includes the username and password for logging on to the Wi-Fi network. After Wi-Fi is connected and authenticated, the MCU will create a HTTP POST request with the ASCII-formatted measurement data and send it to the InfluxDB server.

Afterwards, the MCU will turn off all sensors and communication modules and enter asleep-awake cycles of 15 minutes, with 9 second sleep and 0.5 second wakeup in each cycle. Since the battery management module will automatically shut down power after 30 seconds of low power consumption (less than 45 mA), in the wakeup period, an active low signal is sent to the battery management module to emulate pressing the power key, which keeps the power supply active. After 15 minutes is over, the MCU program starts a new cycle of measurements and uploads.

2.4.2 Cloud Server Software

The program running on the cloud server periodically reads latest measurements from the InfluxDB database for later use with map generation and prediction algorithm. For sensors related to air conditions, including temperature, humidity, pressure, CO, NO_2 and particle matters, the latest reading is used; but since GPS module may not be able to receive satellite signals of enough strength in limited time, and the measurement devices will stay unmoved most of the time, when the latest reading of GPS coordinates is not available, the last available reading is used to map sensors onto the webpage.

The algorithm also needs wind speed and wind direction in the region for prediction, and such data is provided by the Davis weather station. However, the Wi-Fi module of Davis weather station is limited to uploading the data to Davis's cloud service only, and given that Davis's cloud service provides a webpage for weather statistics of the station, the server software includes a web crawler that downloads wind speed and direction from Davis's statistics page.

Then, the prediction algorithm is executed to produce a series of progressive images of air quality over the next few minutes, and all images are copied to the website folder, which is served with a web server program for public access.

2.5 Web Design

Two web pages are hosted on the cloud server for ease of access to sensor readings. The first web page shows the map of the campus covered with the air quality mapping image generated by the server software. The web page allows users to view detailed results from a measurement device by clicking on the dot that represents the device, switch between different metrics for the heatmap, see a color bar for representations of different value ranges, see a wind rose map based on the data from Davis weather station, drag a range slider to see predictions of air quality changes across campus, and download data in CSV format for programmatic or statistics usage.

The other page shows the detailed upload time and raw values from each measurement device, mainly for maintenance personnel to be alerted in case of low battery voltage, unexpected MCU chip temperature or power supply voltage, or a loss of connection of device.

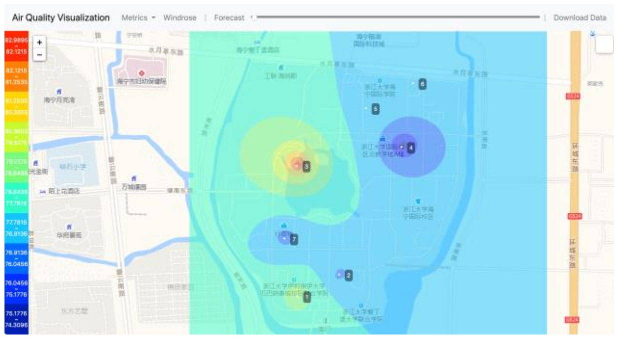
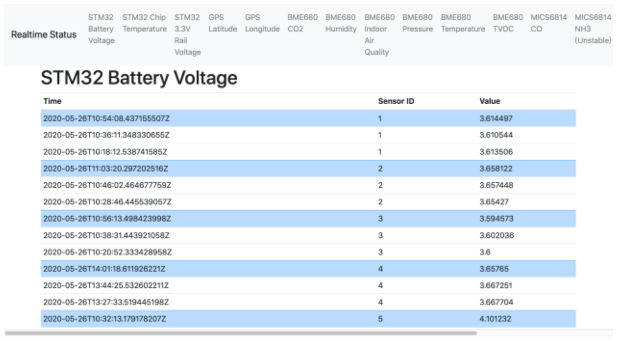


Figure 4:
Web Page for Visualization to End Users



STM32 Battery Voltage

Time	Sensor ID	Value
2020-05-26T10:54:08.437155507Z	1	3.614497
2020-05-26T10:36:11.348330655Z	1	3.610544
2020-05-26T10:18:12.538741585Z	1	3.613506
2020-05-26T11:03:20.297202516Z	2	3.658122
2020-05-26T10:46:02.464677759Z	2	3.657448
2020-05-26T10:28:46.445539057Z	2	3.65427
2020-05-26T10:56:13.498423998Z	3	3.594573
2020-05-26T10:38:31.443921058Z	3	3.602036
2020-05-26T10:20:52.333428958Z	3	3.6
2020-05-26T14:01:18.611926221Z	4	3.65765
2020-05-26T13:44:25.532602211Z	4	3.667251
2020-05-26T13:27:33.519445198Z	4	3.667704
2020-05-26T10:32:13.179178207Z	5	4.101232

Figure 5:
Web Page for Maintenance Personnel

3. Design Verification

The design verification section includes our methods and data on ensuring the functionality of our air quality monitoring system and sensor network, specifically that it needs to meet requirements in the design document.

3.1 Hardware Verification

In this section, we verify that the measurement device hardware can meet the expected design requirements to fulfill their duty.

3.1.1 Seven Days of Battery Life

In our original design document, we expected that our measurement devices should run battery-powered for at least seven days at one-hour measurement intervals, to reduce human efforts in recharging and replacing the batteries. But since then, we decided that one-hour interval is not sufficient for accurate prediction of air quality changes and reduced the interval from one hour to 15 minutes.

We took a measurement device and run it on battery power continuously and recorded the change of battery output voltage over time as they were uploaded along with sensor data to the database. The voltage change is plotted as below:

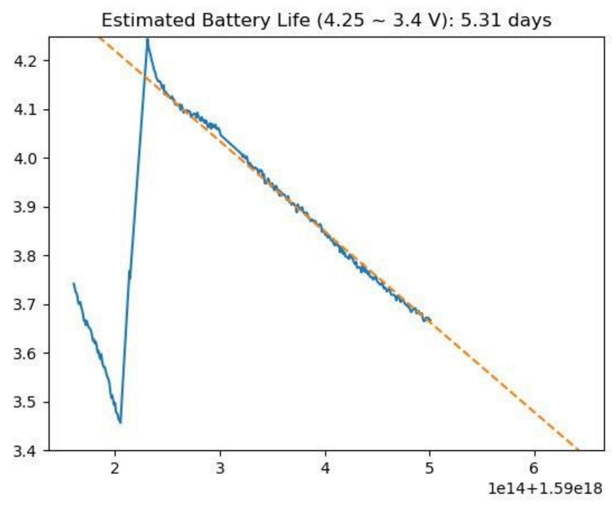
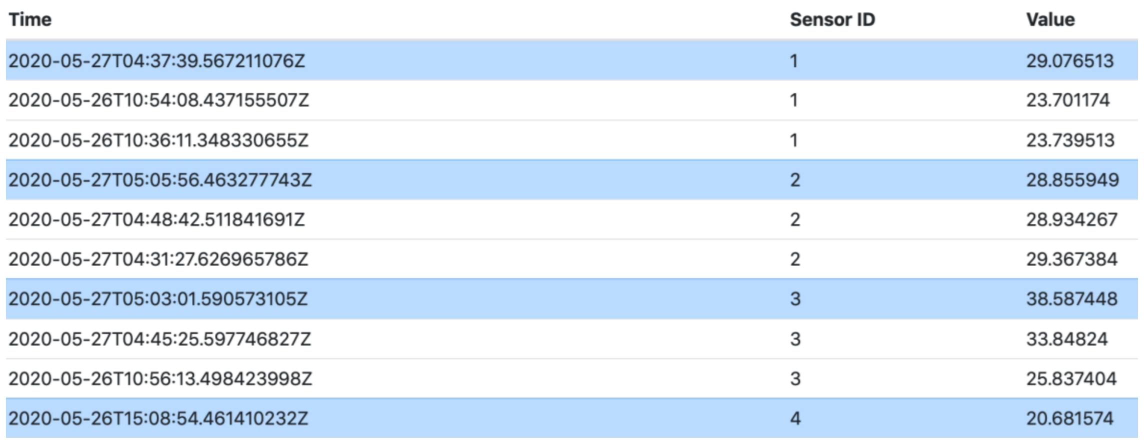


Figure 6: Battery Voltage Curve of Measurement Device

We set the cutoff threshold to 3.4V since it is a safer value for overall lifetime of Lithium batteries. Based on the graph, the batteries contain enough power to run the measurement device for 5.31 days at 15-minute measurement intervals. Although not the same, we still decided that the battery life met our original design expectations.

3.1.2 Safe Component Temperatures

The temperature of all components on the measurement device should be kept below 125°C, so the device will not be in danger of burning or hurting people nearby. We confirmed that by looking at the MCU chip temperature uploaded along with sensor data:



Time	Sensor ID	Value
2020-05-27T04:37:39.567211076Z	1	29.076513
2020-05-26T10:54:08.437155507Z	1	23.701174
2020-05-26T10:36:11.348330655Z	1	23.739513
2020-05-27T05:05:56.463277743Z	2	28.855949
2020-05-27T04:48:42.511841691Z	2	28.934267
2020-05-27T04:31:27.626965786Z	2	29.367384
2020-05-27T05:03:01.590573105Z	3	38.587448
2020-05-27T04:45:25.597746827Z	3	33.84824
2020-05-26T10:56:13.498423998Z	3	25.837404
2020-05-26T15:08:54.461410232Z	4	20.681574

Figure 7: Chip Temperature of MCU

The chip temperature of all measurement devices is within the safe range and will not be a threat.

In addition, we have verified that the batteries we use have built-in short circuit protection during our development process, so in the catastrophic scenario of module failure or water flooding, the batteries will not generate lots of heat for short circuit and risk burning.

3.1.3 Accuracy of Sensors

We verified that most of the sensors produce correct results within 15% error by comparing them to local air quality reports and a more professional device, Air Quality Egg. We run both our device and the Egg side by side and compared their results, which is shown in Table 1

Table 1:Air quality data measured by different Devices at the same time and same places

Device	Temperature	Humidity	pressure	CO	NO2	PM1	PM2.5	PM10
No.2 measure device	25.855297	56.0576	101044	0.385	15.924	26	30	39
No.9 measure device	25.713614	55.749	101002	0.377	16.574	30	34	38
Air Quality Egg	26	56	101020	N/A	N/A	N/A	N/A	N/A

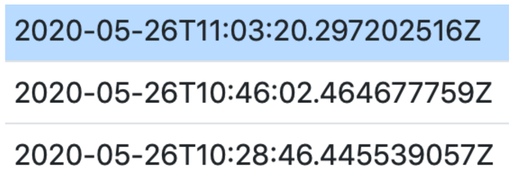
We disabled the TVOC and CO_2 output of the BME 680 sensor as well as the NH_3 reading of the MiCS 6814 sensor, since they are unreliable and are often out of the accuracy range. The accuracy of other readings is verified, and these readings are used.

3.2 Software Verification

In this section we verify the functionality of our software design, including the software running on the MCU and the software on the cloud server.

3.2.1 Sensors Update at Specified Interval

We verified that our sensors could upload data at a specified interval of around 15 minutes by looking at the timestamps of uploaded data recorded by the cloud server. Here is an example of recorded timestamps on our server:



```
2020-05-26T11:03:20.297202516Z
2020-05-26T10:46:02.464677759Z
2020-05-26T10:28:46.445539057Z
```

Figure 8: Timestamps of Uploaded Data from A Sensor

The data is uploaded at actual intervals of around 17 minutes, which is expected since warming up sensors, collecting data, performing GPS localization and uploading data take approximately 2 minutes.

3.2.2 Accuracy of Predicted Map

We verified that the air quality map generated for the present time point (the time when sensors upload data) is within 20% error for most areas of the campus, by comparing the map to one measurement device taken off map. The GPS module of the measurement device is intentionally taken out, so without location data it will neither show on the map nor be considered in map generation. The location of the measurement device is manually recorded, and its data at that location is compared to the generated map.

3.2.3 Able to View Realtime Map in Browser

We checked that the website is available to general public by visiting the website with various computers and browsers to verify its functionality. The website can display maps of different metrics across the campus, display wind rose map based on weather station's data, and provide a download link to the latest data for programmatic or statistics usage.

4. Costs

4.1 Hardware cost

In our project, the major hardware cost includes the cost to build 10 self-designed measure devices and the cost to rent a cloud server. We rent a cloud server for two months to finish our project. For the Davis weather station, we borrow this expensive device from instructor Simon Hu. We can avoid to use this station by getting real-time wind speed and direction from other websites, so we do not count it in the Hardware cost table, which is shown in Table 2.

Table 2: Hardware Cost of the Whole System

Name	Retail Cost (¥)	Bulk Cost(¥)	Actual Cost (¥)
stm32F103C8T6	15.8	15.8	15.8*10=158
ESP8266 (ESP - 01)	16.95	16.95	16.95*10=169.5
pms5003	70	70	70*10=700
BME 680	55	55	55*10=550
18650 battery	88	88	88*10=880
18650 battery holder	3.21	3.21	3.21*10=32.1
CD42 battery module	5.03	5.03	5.03*10=50.3
MiCS 6814	149.6	149.6	149.6*10=1496
GPS Beidou BDS module	38	38	38*10=380
N channel MOSFET	4.15	4,15	4.15*20=41.5
Adjustable resistor	1.5	1.5	1.5*30=45
Connector	0.1	0.1	0.1*50=5
PCB	7	5	5*10=50
Casing	85	85	85*10=850
Cloud server	200/year	200/year	200*2/12=33.3
Total	**N/A**	**N/A**	**5440.7**

4.2 Labor

We use formula (4.2.1) to estimate our labor cost, and the result is shown in Table 3.

$$\text{ideal salary (hourly rate)} * \text{actual hours spent} * 2.5 \qquad (4.2.1)$$

Table 3: Labor Cost

Name	Work Time	Ideal salary	Total
Yuhui Xu	100 hours	¥150	¥37500
Yuhang Ren	100 hours	¥150	¥37500
Ke Liu	100 hours	¥150	¥37500
Total	N/A	N/A	¥112500

5. Conclusion

5.1 Accomplishments

In this project, we designed PCB for our self-designed devices, and send PCB schematic diagram to electric shop to build ten of them. We wrote driver code for all sensors as well as Wi-Fi module and battery module. We designed a water-repellent casing for our measure devices and assembled ten of them. We installed Davis weather station in the roof of platform of the Residential College and wrote web crawler code to collect real-time wind speed and direction data from its official websites. We set up InfluxDB to store and manage air quality data. We implemented both Map Generation algorithm and the Prediction Map algorithm. We build websites as user interface so that students are able to view the real-time air quality map, prediction map, wind rose map and download air quality data.

As a summary, we used totally 7441 lines of code, including C, python, JavaScript and HTML, and achieved an air quality monitoring system based on sensor network that covers the whole Zhejiang University International Campus. All measure device can work normally for at least 5 days. We can give alarm to students based on the monitoring of our websites when the concentration of air pollutants is too high.

5.3 Ethical considerations

Our air quality monitoring system is pretty safe regarding ethical issues. There is no privacy data processed throughout our architecture as all the data we process are air quality data. We placed our PCB and sensors in casing and placed the casing in the corners where few people pass by so that they have no harm to people. For battery security, we purchase batteries all with short circuit protection, and they are safe even if encountering water.

5.4 Future work

By now we disabled MICS6814's NH3, BME680's TVOC and CO2 reading, either because it is so unreliable or inaccurate. The MICS6814 sensor which is used to measure the concentration of NH3, NO2 and CO has large variation across different units and needs manual calibration. We think we can improve the accuracy of generated map by replacing MICS6814 with a more reliable gas sensor. We can also decrease the energy consumption for the measure device. According to our calculation based on datasheet, we can achieve 1mA current consumption when the device is in idle mode, but now it is 6mA. And our air flow model can be improved to be more practical and accurate. For example, we can use our measure devices to measure the data in air pollutants source points, and apply Gaussian Model to predict the diffusion of pollutants. It is also beneficial for our projects if we can get more solid casing, more measure devices as well as richer function in the project websites.

References

[1] STMICROELECTRONICS. “STM32F103C8T6 Datasheet”, 2017. [Online] Available: https://html.alldatasheet.com/html-pdf/201596/STMICROELECTRONICS/STM32F103C8T6/1950/1/STM32F103C8T6 .html [Accessed: 28-05-2020]

[2] Jim McDowall. “A Guide to Lithium-Ion Battery Safety”.2014. [online] Available: https://cmte.ieee.org/pes-essb/wp-content/uploads/sites/43/2016/06/2015- WM-PN-A-Guide-to-Lithium-ion-safety-Jim-McDowall.pdf[Accessed: 28-05-2020]

[3] Ieee.org. "IEEE Code of Ethics", 2016. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html[Accessed: 28-05-2020]

[4] Crank, J. (1956). “The Mathematics of Diffusion”. Oxford: Clarendon Press. [Online] Available: https://books.google.com/books?hl=en&lr=&id=eHANhZwVouYC&oi=fnd&pg=PA1&dq=Crank,+J.+(1956).+The+Mathematics+of+Diffusion.+Oxford:+Clarendon+Press&ots=fz40BWeoOX&sig=gvupcTdAahNky1AbVv30DP_4zFU#v=onepage&q=Crank%2C%20J.%20(1956).%20The%20Mathematics%20of%20Diffusion.%20Oxford%3A%20Clarendon%20Press&f=false[Accessed: 28-05-2020]

Appendix A Requirement and Verification Table

Table 4 System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
The Voltage regulator can provide 3.3V energy source for MCU for more than seven days.	A. Charge the battery to full status. B. Keep the device running and check its battery status continuously for seven days.	Y
Temperatures of components must be sustained below 125°C	1. Run our device with all sensors working. 2. Measure the temperature with IR thermometer to ensure all components do not reach temperatures greater than 125°C.	Y
All these sensors should get correct results within 15% accuracy.	1. Measure data in three different places, three times for each place. 2. Use a professional device (“Air quality egg” provided by Prof. Hu) to obtain reference data in the same three places. 3. Compare the result from our sensors and the result from air quality egg.	Y
All these sensors should measure data in the frequency we specify (we assume once per hour, but can be adjusted based on real-world requirements).	1. Change the measure frequency to once per hour, twice per hour and once every two hours. 2. Check whether our device would upload data as the frequency we specify.	Y
Generate air quality map and for most of location, the data from our air quality map should be 20% accuracy within the reference result.	1. Randomly choose 10 sample points that do not belong to those points that have measuring devices. 2. Setup10validationdevicestomeasure data in these 10 sample points. 3. Compare the data from our generated air quality map with the	Y

	data from validation devices. The difference should be within 20%	
Able to view and download data from our server.	1. Output sensor data from local devices, and verify that the server shows the same data in real time.	Y

Appendix B Schematics for Self-designed Measure Device