THESIS FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

SMART CITIES AIR POLLUTION MONITORING SYSTEM
Developing a Potential Data Collecting Platform based on Raspberry Pi

Prepared By:
Shu Chen

Supervised by:
Antoine Bagula
Omowunmi E. Isafiade

Department of Computer Science
University of Western Cape
Bellville, Cape Town
7535

Plagiarism declaration

- I acknowledge that plagiarism is wrong. Plagiarism is using another's work and to pretend that it is one's own.
- The Harvard Convention was used for citation and referencing. Each significant contribution to, and quotation in, this report from the work, or works of other people has been attributed and has been cited and referenced.
- This report is prepared and produced by myself.
- I have not allowed, and will not allow, anyone to copy our work with the intention of passing it off as his or her own work.
- I acknowledge that copying someone else's work, or part of it, is wrong, and declares that this is the company's own work.

Student No.	Name	Signature
3308638	Shu Chen	[illegible]

Acknowledgements

The research project was funded by the **National Research Fund** (NRF). I thank NRF for their generous and patience and generous financial assistance which gave me financial stability for the duration of my project. I also would like to acknowledge the following people who supported me with input and assistance with the research:

Professor Antoine Bagula, my supervisor, provided me with professional guidance and patience in supervisions. He always gave me ideas to solve whatever challenge I faced and answered my questions kindly and humorously. Being a member of his research group ISAT, helps me a lot with my presentation skills.

Post Doctor Elizabeth Omowunmi Isafiade my course supervisor, English is not my first language, I had some challenges with writing in English and she helped with structure and grammar construct.

Tafadzwa Mandava, he was my teammate for the research project but focusing on other research field. He gave me a lot of help with my experiments as well as developmental advices.

Post Doctor Johanna Von Holdt, she is a lecturer from UCT, I would like to thank her for providing the professional device for the test with our device together and help us to improve our device.

Dr Olasupo Ajayi, I would like to thank him for helping me proofread my paper, and he gave me many constructive suggestions and ideas to improve my paper to become more professional.

I also want to take this chance to thank to my girlfriend **Bodi Liu**, who offered me the confidence and discuss with me about my paper. Of course, I also need to thank my father, **Zhenhua Chen**, my mother, **Guoying Lou**, without their support, I could not finish the research.

Abstract

Air pollution is becoming a challenging issue in our daily lives due to advanced industrialization. This thesis presents a solution to collection and dissemination of pollution data. Most of the devices that monitor air quality are costly and have limited features. The aim of this study is to revisit the issue of pollution in cities with the aim of providing a cheaper and scalable solution to the challenge of pollution data collection and dissemination. The solution proposed in this paper uses Raspberry Pi and Arduino micro-controller boards as the foundation, combined with specific sensors to facilitate the collection and transfer of pollution data reliably and effectively. While most traditional air pollution monitoring equipment and similar projects use memory cards as a medium for data storage, the system proposed in this research is built around a new network selection model that transfers data to the server by using either Bluetooth, Wi-Fi, GSM, or the LoRa protocol. The connectivity protocol is selected automatically and opportunistically by the network selection algorithm defined in the micro-controller board. The final data will be presented to the user through a mobile application and website interface effectively and intuitively after being processed in the server. This data transfer system can effectively reduce the cost and input of human resources. It is a viable solution. For other environmental research, this system can provide an air quality data support for analysis and reference. Modularity and cost-effectiveness are fully considered when designing the system. It is a viable solution. We can generalize the system by slightly changing the data transmission modules. In other case, it can be used as a platform for similar data transmission and offer help for other research directions.

Key words:

Air pollution, Raspberry Pi, Arduino, Networking Selection.

Table of contents

List of tables

List of figures

Glossary

Android is an open-source operating system used for smartphones and tablet computers.

Arduino is an open-source electronics micro controller with an easy to use hardware and software

BeagleBoard is a low-power open-source single-board computer, it is also a credit card-sized computer.

Atmospheric monitoring refers to continuous collection of data and information on atmospheric composition.

β-ray is one of the most common forms of radiation produced by radioactive materials

Bluetooth is a wireless technology standard for exchanging data over short distances

Body Sensor Networks is the network built with multiple types of biomedical sensors (such as human heart rate sensor) and gateway (such smartphone) that can collect data from human body and transmit the data to the server.

Ethernet is a family of computer networking technologies commonly used in local area networks.

Household Air Pollution is the indoor pollution such as indoor smoke.

Human Physiological Parameters are the value which describe human health state such as heart rate, blood oxygen content.

Internet of Things is the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data.

Linux is an open-source operating system modelled on UNIX.

LoRa short for Long Range, it is a patented digital wireless data communication technology

Mesh networks is a local network topology in which the infrastructure nodes connect directly, dynamically and non-hierarchically to as many other nodes as possible and cooperate with one another to efficiently route data from/to clients.

Mie theory describes the scattering of an electromagnetic plane wave by a homogeneous sphere.

MQ Sensor It is series of electro-chemical gas sensor with a small heater.

NB-IoT short for Narrowband IoT, it is a Low Power Wide Area Network (LPWAN) radio technology standard developed by 3GPP to enable a wide range of cellular devices and services.

PubNub is a global Data Stream Network (DSN) and real-time network service company, which focus on innovation and fast launches with readily available APIs and global messaging infrastructure for web, mobile, and Internet of Things.

Raspberry Pi is a credit card-sized computer originally designed for education, it is able to run Ubuntu and Windows Operating system.

Ubuntu is an open-source operating system based on the Linux distribution.

WeChat is a Chinese multi-purpose messaging, social media and mobile payment app.

Xbee refers to a family of devices from Digi that share form factor, host interface and a group of protocols, Zigbee is from one of these.

Zigbee is an IEEE 802.15.4-based specification for a suite of high-level communication protocols.

Acronyms and abbreviations

2G/3G/4G/ second/third/fourth-generation cellular technology

ADC Analog to Digital Converter

ADK Assessment and Deployment Kit

API Application Programming Interface

AQMP Air Quality Management Planning

AT Assistive Technology

BSNs Body Sensor Networks

C_4H_{10} Isobutane

$C_6H_5CH_3$ Toluene

CC Correlation Coefficient

CH_3CH_2OH Ethanol

CH_4 Methane

CO Carbon Monoxide

CO_2 Carbon Dioxide

DC Direct Current

ESSID Extended Service Set Identification

FIFO First In First Out

GPIO General-Purpose Input/Output

GPRS General Packet Radio Services

GPS Global Positioning System

GSM Global System for Mobile communication

H_2 Hydrogen Molecule

H_2S Hydrogen Sulphide

HAP Household Air Pollution

HDMI High Definition Multimedia Interface

I/O Input/Output

I2C Inter-Integrated Circuit

IDE Integrated Development Environment

IoT Internet of Things

IP Internet Protocol

ISO International Standards Organization

LAN Local Area Network

LCD	Liquid Crystal Displays
LED	light-emitting diode
LMI	Low and Middle-Income
LPWANs	Low Power Wide Area Networks
LTE	Long-Term Evolution
MCU	Microcontroller Unit
MIT	Massachusetts Institute of Technology
MQTT	Message Queuing Telemetry Transport
NAT	Network Address Translation
NH_3	Ammonia
NO	Nitrogen Oxide
NO_2	Nitrogen Dioxide
O_2	Molecular Oxygen
O_3	Ozone
OLED	Organic Light-Emitting Diode
PM	Particulate Matter
PM2.5	Particulate Matters with diameter of less than 2.5 micrometres
PM10	Particulate Matters with diameter of less than 10 micrometres
RTC	Real-time Clock
RX/TX	Transmit/Receive
SAAQIS	South African Air Quality Information System
SAWS	South African Weather Service
SCADA	Supervisory Control and Data Acquisition
SCP	Secure Copy Protocol
SCV	Single Customer View
SD card Secure	Digital card
SMS	Text Messages
SO_2	Sulphur Dioxide
SPI	Serial Peripheral Interface
SSH	Secure Shell
TCP	Transmission Control Protocol
TM5	Transport Model 5
TTL	Time To Live
TXD/RXD	Transmit Data/Receive Data

UART	Universal Asynchronous Receiver/Transmitter
UCT	University of Cape Town
UDP	User Datagram Protocol
USN4D	Ubiquitous Sensor Networking for Development
USN	Ubiquitous Sensor Networking
USB	Universal Serial Bus
UV	Ultraviolet Radiation
UWC	University of the Western Cape
VOCs	Volatile Organic Compounds
WHO	World Health Organization
WLAN	Wireless Local Area Network

1. Introduction

1.1 Background and motivation

Environmental pollution has become an increasingly serious issue. Air pollution is a type of environmental pollution and is now one of the prevalent issues that affects human lives. According to the definition by the International Standards Organization (ISO), air pollution is a phenomenon that endangers human comfort, health and even environment. It is usually caused by human activities or natural processes that result in the release of certain substances into the air. Overtime, the concentration of these substances in the air increases and results in polluted air. (CSC, 2017) . Air pollution in urban areas is mostly caused by automobiles, emissions from industrial production, combustion of petroleum products for the transportation and electricity generation; burning of coal (in stoves, heaters and boilers); smoke from forest fire among others. These processes emit particulate matter (PM), lead, nitrogen oxide (NO), hydrocarbons, carbon monoxide (CO), Sulphur dioxide (SO_2), Ozone (O_3) and Volatile Organic Compounds (VOCs) (Lake and Jones, 1975). The effects of air pollution are numerous but main ones are the impact on the human respiratory tract and physiological function; it impacts on plants (withered, fallen and even death); as well as the direct impact on the climate resulting in global warming and acid rain.

$PM_{2.5}$ is a standard used to measure the quality of air and it means the mas per cubic meter of air particles with diameter less than 2.5 micrometres. According to the data provided by the World Health Organization (WHO) in September 2016, 92% of the world's population live in the areas where the $PM_{2.5}$ is higher than the limits given by WHO. Air pollution $PM_{2.5}$ includes sulphate, nitrate and carbon and other pollutants, they can penetrate deep into the lungs and the cardiovascular system, causing great risk to human health(WHO, 2017).

1.2 Statement of Problems

Environmental pollution is a global problem to which the whole world should actively respond to avoid it getting worse. As a developing continent, the impact of air pollution in Africa is immense. The Africa Progress Panel released her "Power, People, Planet: Seizing Africa's Energy and Climate Opportunities at the World Economic Forum" report in 2015 and it showed that about 600,000 Africans were killed every year by air pollution caused by the use of solid biomass for cooking (Cordeur, 2015). In reaction to this, South Africa could strengthen air pollution control and monitor watch the status of air quality from sources. To achieve efficient monitoring, there is however the need to develop an effective environmental inspection system. Though there currently is an air quality monitoring system in South Africa, called the South African Air Quality Information System (SAAQIS), much still needs to be done to make it effective, to use same or less resources to achieve a wider range of monitoring. Information from the official website of SAAQIS shows that the Air Quality Management Planning (AQMP) only covers 20 of the 54 municipals which is only 37% coverage (SAAQIS, 2015a).

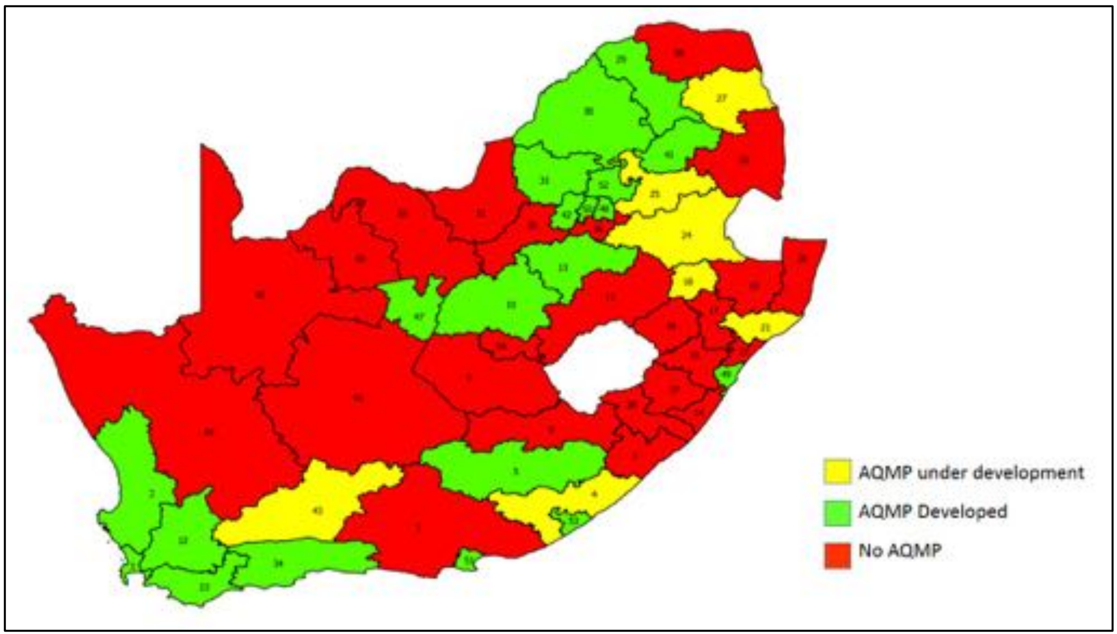


Figure 1-1: District Municipality AQMP development status(SAAQIS, 2015a)

This coverage Figure 1-1 shows that air quality monitoring system coverage is far from adequate for the entire country. Recent research has shown that the air quality data from the website are incomplete in many areas. There is also a wide difference in the types of air monitoring done at different areas. This may be due to the differences in equipment used in the monitoring. At the same time, due to the high cost of purchasing air pollution detectors, it would cost a lot of money to cover the entire country completely. Therefore, it is needed to find an alternative, which is easy to use, inexpensive and reliable to complement and improve SAAQIS. Furthermore, the data collected by SAAQIS needs to be presented to end users in a way that is intuitive and easy to understand. As of the beginning of this research, SAAQIS still adopts a very traditional way to show the data back to the users by asking users to submit an online application and then send the requested data back users through email.

1.3 Research Content and Objectives

The aim of the research is to develop a new, independent data collection and dissemination system that can collect data relating to air quality (environmental information) and provide a platform for sharing the data among users who need the information on environment pollution. Considering that majority of existing air pollution detecting system are expensive and have limited functionality, this research proposes a relatively inexpensive yet reliable solution. The solution focus on low cost, effective and reliable distributed infrastructure based on wireless sensor networks, physical circuits and microcontroller board computing technologies for air pollution monitoring in the South Africa. The underlying system will be used to:

1. collect data related to air quality in relation to human physiological parameters

2. mine this data for situation recognition
3. provide a platform for sharing and dissemination of the data to the general public, government and environmental agencies that need the information
4. provide a more cost-effective way on air pollution monitoring than SAAQIS

The solution consists of both hardware and software components. The hardware part of our solution is mainly for data acquisition and temporary data storage, while the software part of the solution serves the purpose of rigorously analysing, extracting useful information and presenting them in a way that the users can interpret with ease. The system can be used either as an independent air pollution monitoring system combining fixed and mobile sensors or as a complementary system to support the existing air pollution equipment in providing data collection in areas not covered by the AQMP of the SAAQIS. When combined with body sensor networks (BSNs) carried by humans, it can be used to provide useful insights on the impact of pollution on the citizens and/or advise on safer living areas of the city.

1.4 Research Value

The rapid deterioration of air quality has necessitated the need to actively monitor air conditions. The importance of monitoring air quality is numerous some of which include but are not limited to: providing data for scientific research and providing the general public with useful and easy to understand information on the adverse effect poor air quality can have on their health.

Traditional air monitoring stations usually acquire data through tedious and manual means. The large footprints of these stations also implies huge operational and maintenance cost. They are also often limited both in terms of coverage and capacity to effectively capture timely and accurate data needed for air pollution monitoring and control.

The solution proposed in this research work is a low cost, low power consumption and easy to install system. It can meet the current market demand and can achieve a wide range of deployments. It has the advantage of filling the visibility gap left by the costly and complex stations currently being used to monitor air quality in South Africa. Information obtained from SAAQIS website shows that there are 18 monitoring stations in Cape Town area. These are shown in Figure 1-2. Each station can only take measurement with a certain range. These coverage areas are shown with green circles in Figure 1-2. From this, it is clear that there are still a lot of areas not covered by the SAAQIS and these are shown with red circles in the figure.

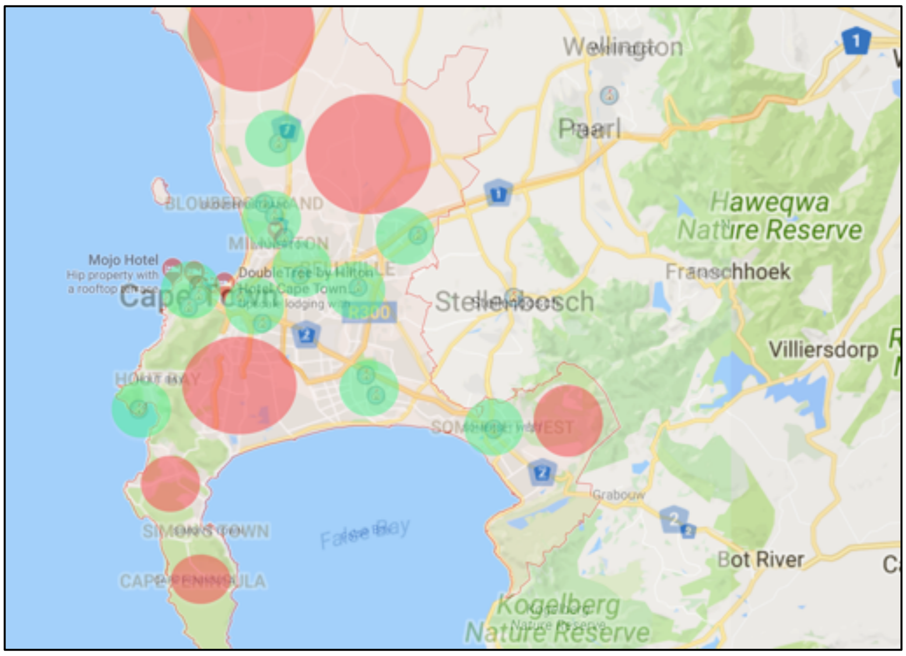


Figure 1-2: Air Quality Monitoring Stations on Map(SAAQIS, 2015b)

The significance of developing this new system is to provide users with real-time air quality data. Compare to SAAQIS' website, the new system would present the air quality information in a simple manner, making it easy to read and understand. The system's design would be modular and have an automated design in order to make it more intelligent and easier for maintain and upgrade in future. A variety of data transmission methods will be included to accommodate for multiple usage scenarios (such as in home). The new system would thus provide information for prevention and pro-active plan to check environment problems before they get worse.

The research would present a reliable, low cost and multi-functional solution capable of determining the various sources of the pollutants and correlate air pollution to its impact on the environment and humans. It thus enables the implementation and enforcement of preventative measures and penalties in order to provide a healthy environment for South African inhabitants.

1.5 Method

The research would use Internet of Things (IoT) technology as this gives the advantage of great functionality at minimal cost. This would help keep the cost of equipment for development low. A microcontroller board such Raspberry Pi and Arduino would have a gas sensor attached to it to sense the level of pollution in the air then send the data to a server. The sensor once exposed to the polluted gas will convert the chemical signal into electronic signal. This electric signal is then sent to the microcontroller. Different sensors detect different gases, by combining different sensors, a variety of gases can be detected. The collected data are stored on the microcontroller board temporary or transmitted to the server via different data transmission modules connected to the microcontroller. At the server side, the received data are processed and useful information (such as the percentage of certain pollutant in the air) obtained. End users can either use an application (App) installed on their mobile phones or visit a website to get the information on air pollution.

A number of technologies and disciplines are combined in this research work, such as chemistry (gas), physics (circuit), and computer science (machine leaning, networking and database). The development of the prototype using a microcontroller can greatly reduce the cost of final products. The choice of cheap but reliable sensors is also one of the focuses of this research. The use of a varied types of sensors enables the system to meet the needs of different data collection.

1.6 Scope and limitations

This research focused mainly on the system development and testing the feasibility of the proposed solution. Therefore, due to budgetary constraints, the proposed solution (hardware) is not ready for mass production at the current stage. Also, it is only able to detect a limited number of gases, as only a few sensors that had detection accuracy as close to those of professional equipment used in air monitoring stations were available for use. Finally, the test environment was mostly indoors. This was because the electronic components of the hardware were affected by environmental factors especially the rain. The prototype has however been tested both in China and Cape Town, South Africa.

1.7 Chapter outline

This research paper is divided into five chapters.

Chapter 1 introduces the background and the problems that the research aims to solve.

Chapter 2 provides some literatures relating to the air pollution monitoring, introduces the current research status and some common air pollution monitoring methods that are currently in use.

Chapter 3 shows the application scenarios of the system and the specific design ideas.

Chapter 4 describes the research, specifically the hardware platform and software development.

Chapter 5 presents the data from the system as well as the analysis of the result

Chapter 6 presents a summary of the findings and future plans

The Appendices provide the supporting documentation and include: the list of the sensors use in the system, their instructions set, and authorization letter of the Stellenbosch gas monitoring station

2. Literature Review

The purpose of including a literature review in this research is to present the state of the art in terms of air monitoring solution and discover which method is more suitable for South Africa. After reading a large number of articles and studying their proposed solutions, it is shown that the scheme of this research work is both feasible and relatively acceptable. This chapter will mainly introduce and discuss the current air pollution situations and some existing air detection solutions used in related projects.

2.1 Air pollution status and development trend

The rapid development of industries has worsened the air quality and global warming is now a real threat. Human life is dependent on air and human health is closely related to the air quality. The need to monitor air quality is therefore extremely important. Monitoring of air quality in different regions can provide data to support future air condition research and possible enable researchers proffer effective solutions to air quality challenges in the earliest future.

There are many cases of diseases and deaths caused by air pollution every year in the world. According to the data from WHO's paper “Burden of disease from Household Air Pollution for 2012”, globally, there were 4.3 million deaths recorded as a result of Household Air Pollution (HAP) in 2012. Most of these people were from low and middle-income (LMI) countries. This number is much higher than the estimated death of 2 million in 2004 (WHO, 2009). Among the 4.3 million, 1.69 million people died in Southeast Asia and 1.62 million died in Western Pacific. Nearly 600,000 people died in Africa, 200,000 in the Eastern Mediterranean, 99,000 died in Europe, and 81,000 died in the Americas. The remaining 19,000 deaths occurred in high-income countries. The data is as depicted in Figure 2-1 extracted from (WHO, 2012).

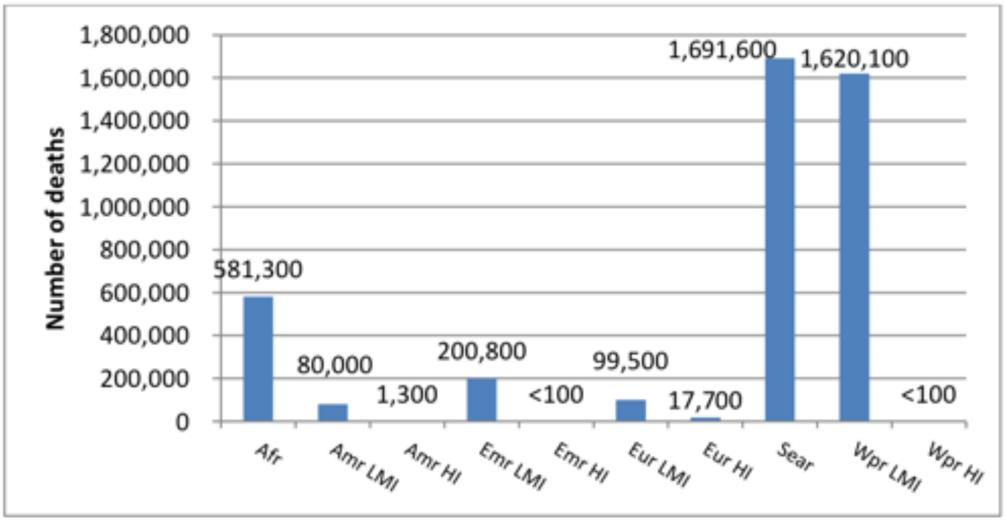


Figure 2-1: Total deaths attributable to HAP in 2012 by region(WHO, 2012)

(HAP: Household Air Pollution; Amr: America, Afr: Africa; Emr: Eastern Mediterranean, Sear: South-East Asia, Wpr: Western Pacific; LMI: Low- and middle-income; HI: High-income.)

It is not difficult to see from Figure 2-1, that in addition to South-East Asia and Western Pacific, Africa is also one of the three most severely affected areas of air pollution. If the number of deaths is converted into proportions, the death toll in Africa accounts for 14% of the world as shown in Figure 2-2. This is a very serious problem. There are three reasons that might be responsible for this, which are: air pollution, incompleteness of the air detection system and limited knowledge on air pollution combined with poor medical care.

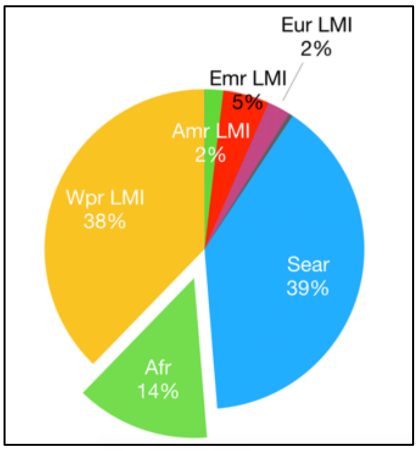


Figure 2-2: World death percentage for different regions

According to "Air Pollution: Africa's Invisible, Silent Killer" of WHO released in 2016, the global urban air pollution level increased by 8% from 2008 to 2013. Also, more than 80% of people living in urban areas are exposed to air quality levels that exceed WHO limits. This is threatening to human lives, productivity and economic efficiency (UNenvironment, 2016). In Africa, urbanization coupled with poorer levels of urban planning has led to a large number of people living in crowded, poorly serviced housing facilities. This situation exacerbates the pollution problem.

John Vidal mentioned in his article “Air pollution more deadly in Africa than malnutrition or dirty water, study warns”, which is a study by a global policy forum; has found that air pollution in Africa is causing more people to die prematurely compare to unsafe water and child malnutrition. If unchecked may deteriorate into a health and climate crisis reminiscent of those seen in China and India (Vidal, 2016). In the article he quotes Rana Roy's research in "The cost of air pollution in Africa", for the continent’s pollution, where it was reported that 712,000 people could be killed prematurely in a year, compared to approximately 542,000 from unsafe water, 275,000 from malnutrition and 391,000 from unsafe sanitation.

Although most major environmental hazards having been improved upon with development and industrialization. On the contrary, outdoor air pollution from transportation, power generation and industry is rapidly increasing; especially in Egypt, South Africa, Ethiopia and Nigeria. Between 1990 and 2013, the world total annual death from ambient particulate matter pollution increased by 36% to approximately 250,000, while deaths from household air pollution increased by 18% to over 450,000. For the whole of Africa, the estimated economic cost of premature air pollution deaths in 2013 was approximately $215 billion for outdoor air pollution and $232 billion in household air pollution per year(Roy, 2016).

Air pollution can be defined as the release of harmful substances into the atmosphere. Common air pollutants include: Sulphur dioxide (SO_2), Nitrogen Oxides (NOX), Ozone (O_3), Particulate Matter (PM), Carbon Monoxide (CO) and Volatile Organic Compounds (VOC). Pollution-related diseases can be attributed to several different forms of exposure. There are three main sources of air pollution which are indoor pollution (from burning of solid fuels), outdoor environmental (Ozone (O_3)) and outdoor Particulate Matter (PM) pollution. Ritchie & Roser pointed out in their article "Air Pollution" that the three sources of air pollution vary from country to country. The charts in Figures 2-3 and 2-4 shows the age-standardized death rate (per 100,000 people) of air pollution from 1990 to 2015 (Ritchie and Roser, 2017).

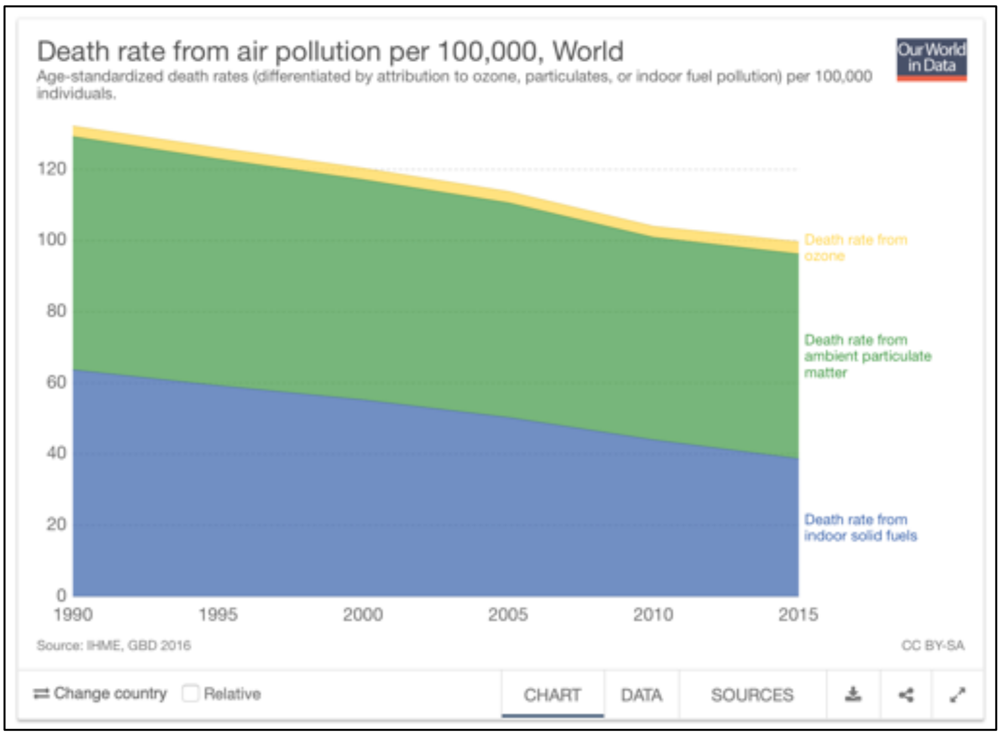


Figure 2-3: Death rate of air pollution(Ritchie and Roser, 2017)

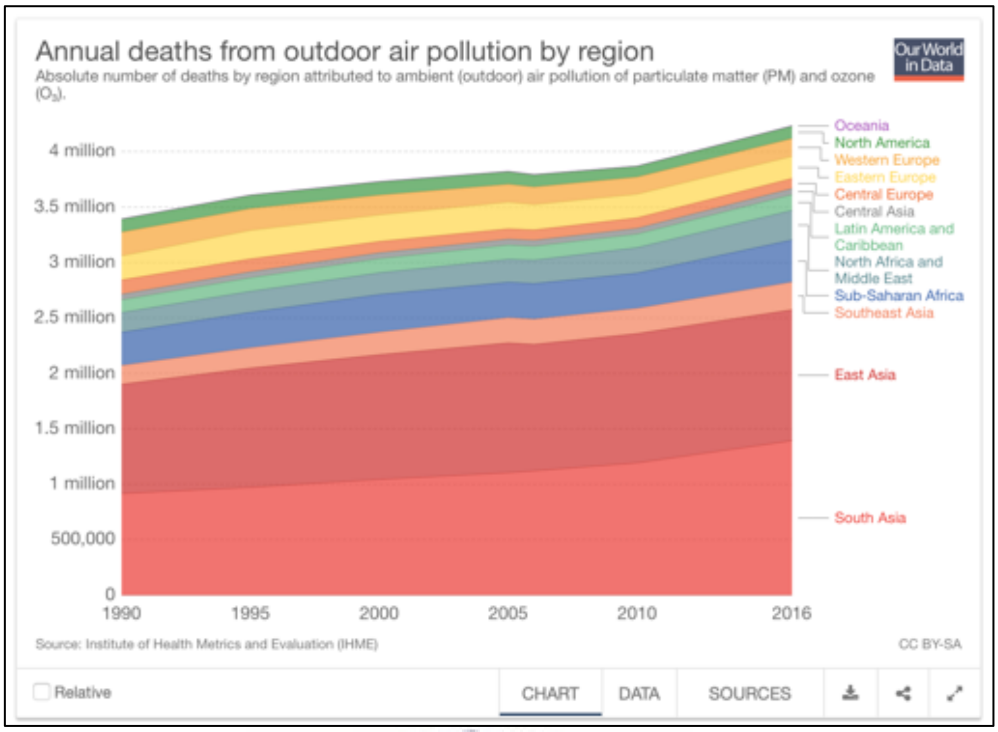


Figure 2-4: Annual death from outdoor air pollution for different regions(Ritchie and Roser, 2017)

Data from the charts shows that although the death rate due to air pollution has generally declined over the past few decades, the number of people dying from outdoor air pollution has risen. Therefore, it is necessary to do some work in detecting and protecting air pollution. Though the technology to measure air pollution levels are available, the lack of funds and availability of these sophisticated monitoring systems in many countries, has restricted measurement to only major urban centres. This has resulted in nearly 90% of the deaths associated with air pollution in low-income and middle-income countries (WHO, 2016).

Most developed countries have high-tech detection equipment and monitoring systems that can collect and analyse air quality and then process the data to generate notifications and adjustments to national policies. However, in Africa the technology in this area is very limited, resulting in the lack of air quality data and indirectly leads to an increase in related health risks. The need for additional responses to air quality data and statistics to guide decisions and address the challenges of poor air quality is critical (UNenvironment, 2016).

2.2 Status of Air Pollution Monitoring and Detection Methods

For air pollution detection systems, the commonly used detection methods are basically divided into two types: the grid-based monitoring system, which is often used to monitor air quality over a wide area range; and the point detection system which is applied to factory pollutant detection, home environmental air pollution detection, and for monitoring pollution around specific areas. Grid-based systems usually use environmental testing equipment plus big data and other technologies to set up monitoring sites and online monitoring equipment in a certain area to form a wide-ranging and dense grid-based monitoring; the grid-based system focus on monitoring air quality levels and sources of pollution and provide a strong reference to improve air quality and show governance results based on comparative data. That is why a region or country often uses the grid method to detect air pollution.

Ma *et al.*, (2008) had applied a method based on sensor grid for air pollution monitoring and mining on London. In their experiment, they installed sensors on various vehicles (including buses and taxis) to act as mobile air quality collection platforms. The Air pollution data collected were sent to the server via the nearest static sensor nodes, as shown in Figure 2-5.

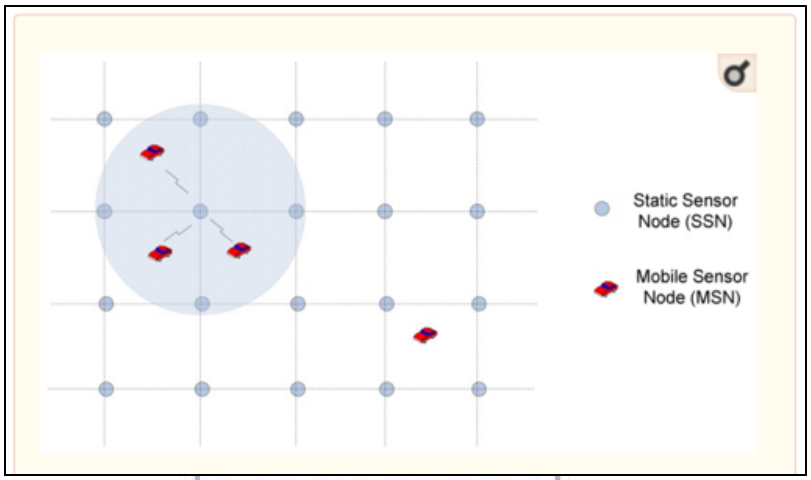
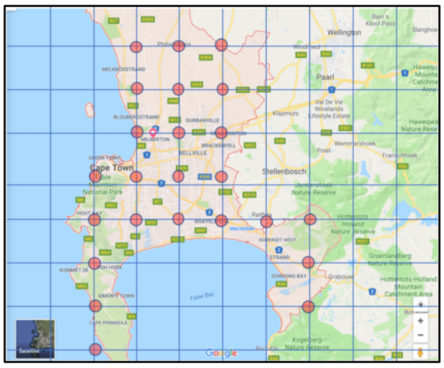


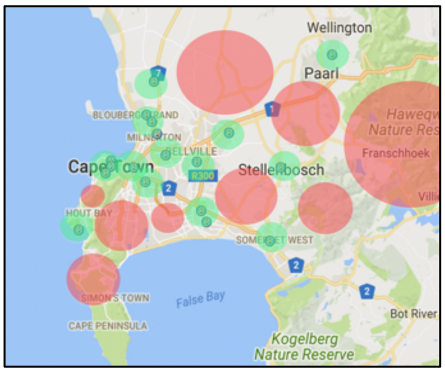
Figure 2-5: The network architecture of mobile distance network(Ma *et al.*, 2008)

Through the method, they effectively collected air pollution data and trends in parts of London. By analysing the changes in air quality in different grid areas, it was easy to determine the source of pollution and the trend of pollutants. It is worth mentioning that a part of the data given by WHO for "Total deaths attributable to HAP in 2012" in the previous section was also collected by using a grid-based model.

The model used to evaluate the content of $PM_{2.5}$ uses a combination of satellite data, global chemical transport model TM5, and ground measurements (Brauer *et al.*, 2012). The model was built on a grid of 10km x 10km. The model collects data from 1990 to 2005 and uses the data binding model as the basis for the 2010 disease burden estimation. WHO also pointed out that the resolution of the model should also be appropriately adjusted in conjunction with local situations. The estimates for data-poor countries may be relatively inaccurate, which may result in significant changes in air pollution levels at the borders of data-poor countries. In order to reduce the uncertainty of modelling data, countries must continue to improve ground measurement plans(Dalgleish *et al.*, 2007).

Therefore, in order to achieve the purpose of regional air pollution monitoring and management, the application of atmospheric grid detection requires comprehensive coverage for effective monitoring of the entire area. Monitoring points should be placed in the target area according to the type and demand of different pollutants, so as to achieve the real-time reflection of overall air quality conditions and trends in the region. If monitoring system built according to the grid model used by WHO (based on 11km x11km), then at least 24 or more monitoring points are needed in the Cape Town area. Currently there are only 18 gas monitoring stations in the Cape Town area as shown in Figure 2-6.


(a)



(b)

Figure 2-6: Example grid-based model for Cape Town Area(a), Current gas stations in Cape Town(b) (SAAQIS, 2015b)

It can be seen from Figure1-6(a) due to the large grid monitoring area, a large number of points need to be deployed. Most the deployed monitoring points are fixed, and the cost associated with deploying each single point is enormous. These have limited wide-spread deployment in low-income countries. As a result of this, challenges such as coverage and incomplete monitoring pollutants, low level of information, incomplete monitoring and supervision, and low quality of monitoring data are prevalent. It is difficult to meet the needs of air pollution control. At the same time, the grid-based monitoring system based on the traditional air pollution monitoring stations is also accompanied by problems especially higher operating costs.

Thus, the use of alternative monitoring technologies such as sensor-based monitoring equipment have spawned. It can achieve the wide distribution, comprehensive coverage of the monitoring area, eliminate the blind spots in monitoring and provide scientific and technological support for local governments to conduct environmental monitoring of atmospheric grids due to its low cost, convenient installation and continuous dynamic monitoring(CNEMC, 2017).

However, in the actual application process, sensor monitoring method alone cannot be used for grid module construction. This is because sensor-based equipment is prone to drift in data, resulting in inaccurate data. Only after this problem has been solved can the advantages of sensor-based grid module be effectively exploited. Therefore, the use of a hybrid combination of traditional monitoring stations, sensor-based equipment deployment and data unified linkage calibration, would not only provide accurate data to ensure that the data is complete and scientific, but also greatly reduce cost. This hybrid monitoring solution can be effectively used to achieve air detection in LMI countries.

2.3 The existing projects and solutions

2.3.1 South African Air Quality Information System (SAAQIS)

Currently the air pollution monitoring system used in South Africa is the South African Air Quality Information System (SAAQIS). It is a public platform affiliated with the department of environmental affairs that provides and manages air quality data for South Africa. Key features of the system include data collection, storage, management, and data analysis. The objectives of SAAQIS include managing air quality data for the revision and establishment of the National Environmental Management: Air Quality Act; ensuring that air quality information is complete, accurate, and timely, and providing it to all stakeholders and the public to alert the public whether current air quality is harmful to health, etc. (SAAQIS, 2004).

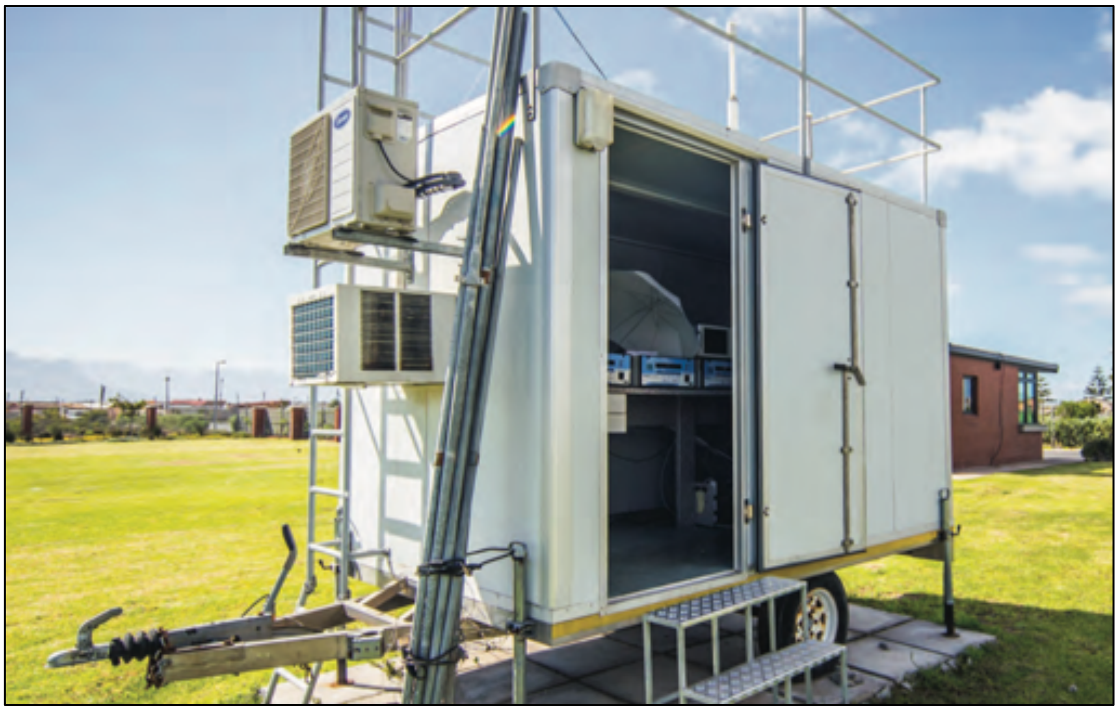


Figure 2-7 SAAQIS gas monitoring station(Western Cape Provincial Government, 2015)

The air pollution data collection method used by the SAAQIS is mainly based on air pollution monitoring stations and combined with third-party data sources such as Eskom, Coega Development Corporation, etc. For example, in Cape Town area there are 18 inspection stations that can be found on SAAQIS website with 13 monitoring points for the gas stations within the city, 4 belong to the Western Cape government and the last belonging to the South African Weather Service (SAWS). The monitoring equipment used in the stations are mainly the air testing equipment produced by Thermo Fisher Scientific. According to the information provided by employees in SAAQIS, the Thermo Fisher Scientific i-series of monitoring devices are used for O_3, SO_2, NO & CO detection; the Thermo Fisher Scientific FH62 and C14 are used for PM10 & $PM_{2.5}$ detection respectively; while the Thermo Fisher Scientific 5012 MAAP is used for black carbon. The SAAQIS uses a plurality of detection devices to jointly operate in a simultaneous manner to form a monitoring station to complete collection of various air pollution data. The method of data collection is mainly based on monitoring stations. The devices at the monitoring station send the data back to the database via the Internet, or manually by the personnel who download and record the data from the devices. The data collected by the system was design for the public and system website also has the function of displaying air quality data, unfortunately the system does not display any data by clicking on the webpage for current stage. As a part of the research, our project tried to use the data request function on system website but got no response from the system. This is probably because the data display function was either incomplete or imperfect.

The advantage of the system is that the monitoring station using professional equipment and can this obtain more accurate data. The uniform installation of the monitoring equipment in the station can also reduce the influence of the installation environment of the equipment. The equipment is capable to collect data in a stable and long-term manner and is capable of monitoring a variety of air pollution. The greatest disadvantage is obviously the high cost of the monitoring equipment that make up the station. Figure 2-8 shows an example of the Thermo Fisher Scientific 48i gas filter correlation CO analyser that used for the CO detection.

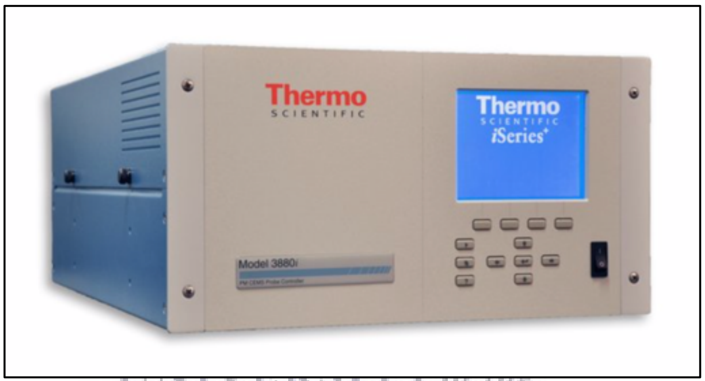


Figure 2-8: Thermo Fisher model 48i CO analyser(ThermoFisher, 2018)

Although the precision of the data collected by the device is as high as ±0.1 ppm, the price of a single CO detect device is as high as ZAR 185,712 (quoting price from www.eroelectronic.co.za at 31st Sept. 2017). There is the need for multiple monitoring equipment to be installed to monitor multiple air pollutants at different areas at the same time, but it is prohibitively expensive to do so. Also, the large size of the monitoring station, makes it is difficult to change the monitoring location at will, which limits the use of such monitoring stations in some special scenarios. This is also one of the main reasons for the low coverage of this system. Another major disadvantage is that data does not have the ability to updated in real time.

2.3.2 Ubiquitous Sensor Networking for Development (USN4D): An Application to Pollution Monitoring(Bagula *et al.*, 2012)

USN4D is an improvement and application based on USN proposed by Bagula *et al.* in 2012. The USN4D architecture combines the concept of IoT with sensor-based technology. In addition to data transmission, it also has the ability to be deployed over long distances to meet the needs of developing countries. The main benefits of adopting this architecture is the lower deployment cost. The network of sensors consists of thousands of sensor nodes, making it easy to achieve a wider range of coverage and efficient data dissemination. For developed countries, there are generally well-established wireless or wired network facilities that can be used for long-distance propagation of data collected by sensors, while developing countries lack such infrastructure, thus with one of the sensor nodes acting as a gateway, so the transmission distance of the developing countries will be much limited by wireless or wired network facilities. The article also pointed out that this model can help provide early warnings for natural disasters such as floods, hurricanes, droughts, earthquakes, epidemics etc. as well as provide support of urban monitoring and management, and environmental monitoring. Specifically, the researchers used an air pollution monitoring application in the article to verify its practicability. The air quality monitoring experiment was tested in Cape Town to verify the feasibility of the USN4D architecture for the establishment of sensor equipment in developing countries.

In the specific experiment, the researchers used Waspmote motherboard-based equipment connected with some gas sensors to measure the pollutants. The device is shown in Figure 2-9. The sensors connected to the device include temperature, humidity, atmospheric pressure and 11 different types of gas sensors which are Carbon Monoxide (CO), Carbon Dioxide (CO_2), Molecular Oxygen (O_2), Methane (CH_4), Hydrogen Molecule (H_2), Ammonia (NH_3), Isobutane (C_4H_{10}), Ethanol (CH_3CH_2OH), Toluene ($C_6H_5CH_3$), Hydrogen Sulphide (H_2S), and Nitrogen Dioxide (NO_2). The device allowed access to the gas sensors simultaneously. The device records the sensor data as well as the time and location on a SD card.

The communication modules included in the device include GPRS, ZigBee and XBeePro. When the device detects the existence of the corresponding communication network, the data stored in the device will be uploaded to the server. The collected data is then analysed and pushed to citizens and relevant agencies.

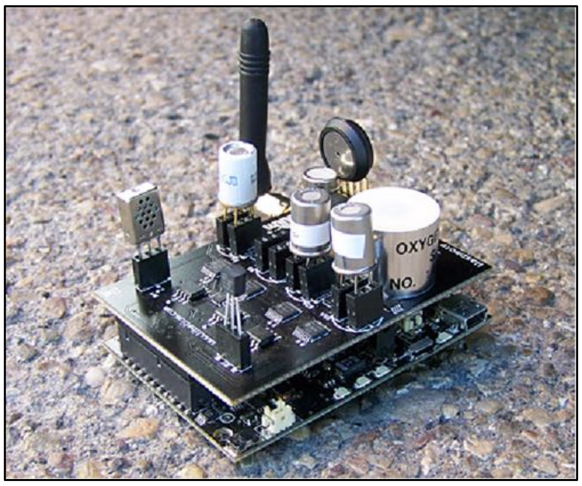
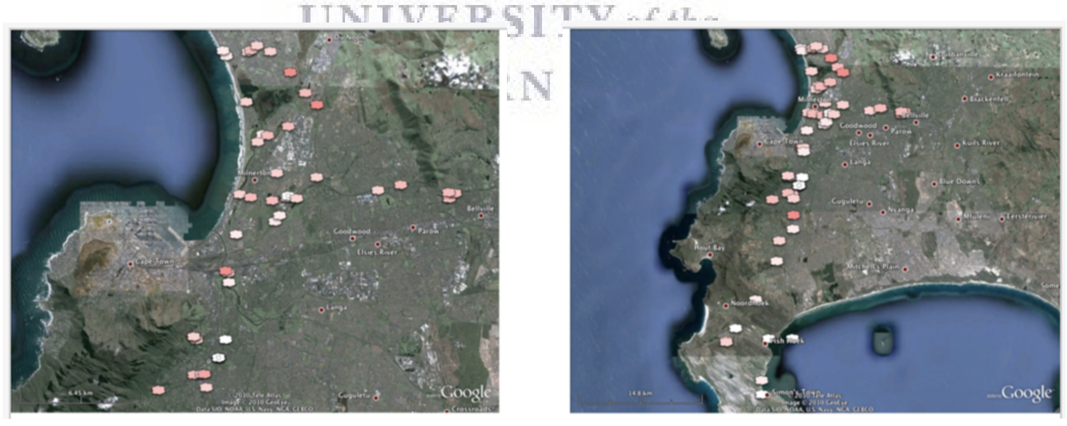


Figure 2-9: A Waspmote Device(Bagula *et al.*, 2012)

Due to the use of uncalibrated sensors in the device, the researchers had to first calibrate the device in a low-density population area called Fernwood near the National Botanical Gardens 23/07/2010 before usage. The air quality in this area was used as the standard for clean air and the resistance value of the sensors were recorded. Air quality data was subsequently collected in the Cape Town area using the device on 24/09/2010 and 29/09/2010 respectively. Although the test did not produce accurate air pollution data, the paper plotted the values of the sensor resistance on google map and found the following results: the northern part of Cape Town area was more polluted than the southern area (the deeper the colour of the marking on the map the more serious the degree of air pollution).



(a) (b)

Figure 2-10: Pollution mapping, first day test(a), second day test(b)(Bagula *et al.*, 2012)

The air monitoring application described in this paper uses inexpensive sensors and it is capable of simultaneously monitoring multiple air pollutants on the same device, which undoubtedly reduces the cost of air pollution monitoring. At the same time, the paper proposed the use of wireless sensor networks to help data transmission, which greatly increased the distance of data transmission, improved the performance of data transmission and reduced the data transmission costs. It also greatly reduced the dependence on wireless networks and wired networks. The shortcomings exist in the paper was the use of sensors that had not been professionally calibrated, so accurate data of the air pollution could not be obtained. The result of the device could therefore only be used as a reference for changes in air pollution.

2.3.3 AirPi project and Goleta Air Monitoring Project

The AirPi Project (Hartley, 2013) is a shield developed for Raspberry Pi. The project was developed in 2013 by Alyssa Dayan and Tom Hartley from Westminster School, UK. The AirPi is similar in size to the Raspberry Pi, and its functions includes collecting temperature, humidity, air pressure, light levels, UV levels, carbon monoxide, nitrogen dioxide and smoke levels and uploading it to the Internet. The main connection method is to directly plug the AirPi shield on the GPIO of Raspberry Pi to start air pollution monitoring. The Raspberry Pi can read the data on the sensor through I2C and SPI communication protocol. According to different needs, users can choose to install a full set of sensors on the shield or just install individual sensors, and then download the corresponding code to the Raspberry Pi to complete the installation of the device. Due to the low cost of the equipment and the simple deployment, users can easily use the equipment to observe and monitor the air quality in their area. The data collected by the device will be available in the Internet, and users can view real-time air quality trends to predict changes in the surrounding environment. AirPi is an open source project that is especially suited for use at the individual level, such as where the user's area is not covered by the government's weather monitoring system. A properly placed AirPi device can even be used to provide weather data to support the government’s systems.

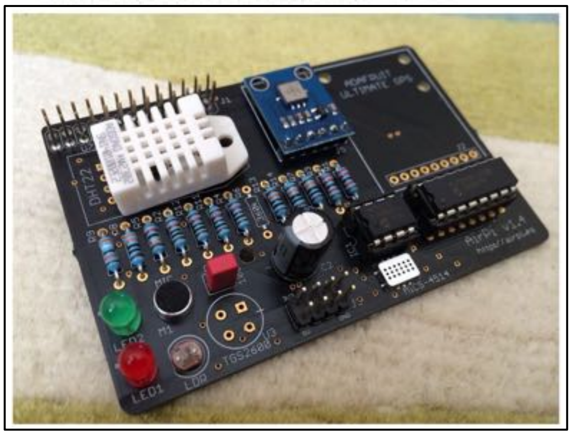


Figure 2-11: AirPi kit for Raspberry Pi(Hartely, 2014)

The Goleta Air Monitoring project(Zymbit, 2016) is a device developed for the commercial environment. It was developed by the Zymbit community and the sensors used in the project are all commercial grade. The manufacturers designed a customized motherboard to integrate multiple sensors (all sensors are mounted on a customized motherboard) and the motherboard is then connected to an Arduino Zero board. The Arduino simply collects and packs the data from the sensors and sends it to the Raspberry Pi through a serial connection. The customized sensor board contains PM, CO_2, relative humidity, temperature, and air pressure sensors, and it is able to collect five parameters at the same time. The entire device only needs a simple USB cable for power supply, and the users can choose to connect the device to Wi-Fi or Ethernet in order to automatically upload the data collected by the device to the Zymbit community server. The received data can be used on the Zymbit side to generate real-time charts to help analyse the data.

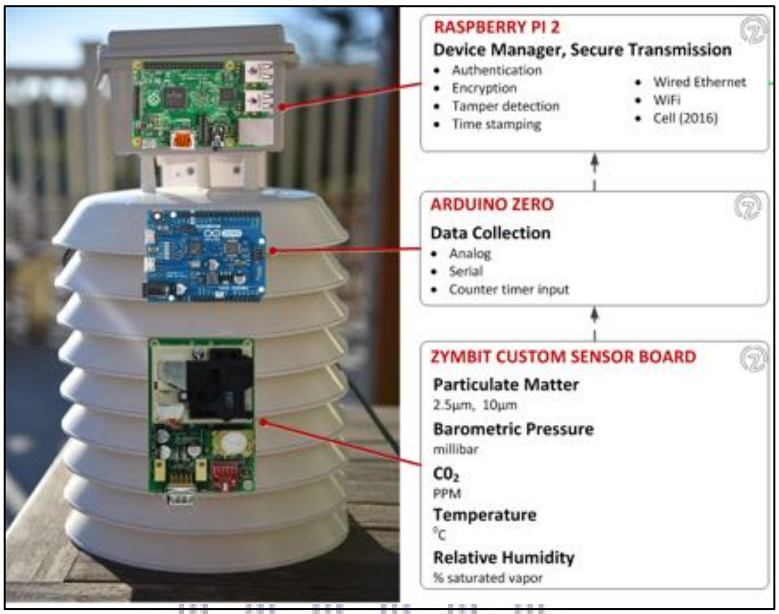


Figure 2-12: Goleta air monitoring device (Zymbit, 2016)

As seen in Figure 2-12, the entire device is placed in a multi-plate radiation shield to help protect the unit from damage from rain while ensuring that external airflow can pass through the sensor. This makes the device extremely easy to deploy. The project also has a very friendly visualization interface in the final data section. Figure 2-13 shows an example of temperature data collected by five different devices placed in different locations, where GAM represents the device name.

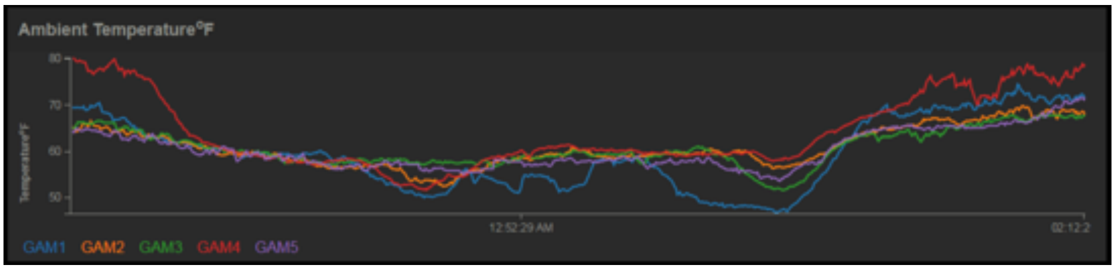


Figure 2-13: Real-time charts of the Goleta air monitoring project(Zymbit, 2016)

Both projects are sensor-based air pollution devices developed based on the Raspberry Pi. All have the advantages of low cost of IoT devices, small and portable, and support real-time online query data.

As a completely open source project, AirPi's software and the sensing components are both modular in design, which means that the given sensor selection can be installed exactly as needed, so that other project developments based on this will become very easy. Another project, the Goleta Air Monitoring, incorporates a time module that allows device to be running in long-term data collection without need of Internet. Though sensors can be added as needed, both the AirPi and Goleta Air Monitoring project use a customized mother board and are designed for specialized sensors, this invariable puts restrictions on the choose of available sensors (to suit different scenarios and different precision of detection requirements). At the same time, due to the use of an integrated mother board for the sensors, it will be slightly more troublesome to add new sensors in the future, because the sensors used in current equipment are still far from the atmospheric monitoring in terms of accuracy. On the other hand, both projects only temporarily support Wi-Fi and Ethernet communication for data transmission, the transmission path is relatively simple, which is very limited in deployment in countries and regions where the Internet is not well developed. However, these two projects do not require high learning costs, and are still very suitable for use in the home environment and industrial environment.

2.4 Theoretical Background

2.4.1 IoT technology

With the advent of smart manufacturing or Industry 4.0, the IoT has the power to change our world, says Gasiorowski-Denis(Elizabeth, 2016). Professor Kevin Ash-ton from Massachusetts Institute of Technology Auto-ID Center first proposed the concept of the IoT when studying RFID in 1999 (Ashton, 2009).The concept of IoT is actually based on the Internet, which is a network concept to connect any objects through the IoT domain name in accordance with the agreed agreement to exchange information and communication, in order to achieve intelligent identification, positioning, managing and monitoring. Its purpose is to realize the connection between objects and objects, objects and people, all items and networks. In 2003, the US "Technical Review" proposed that sensor network technology will be the first of the top ten technologies that will change people's lives in the future. The IoT is a network of connected devices and gateways that continuously collect and send data. Broadly speaking, the current application of information technology can be included in the scope of the IoT, which has three important characteristics: intelligence, advanced, and interconnected. Nowadays, with the development of the IoT, many IoT devices have entered our homes, such as thermostats, smoke detectors and alarm systems.

Common IoT development platforms include Arduino, Raspberry Pi, BeagleBoard, etc. Compared with traditional project development, projects based on the IoT platform have lower development costs, better communication compatibility using not only close-range communication such as Zigbee, Wi-Fi, and Bluetooth but long-distance transmission protocols such NB-IoT and LoRa. Data model development is more flexible for different application scenarios.

2.4.2 Common Air Pollution Detection Methods

Gravimetric method:

The principle of the gravimetric method is to filter the pollutants using a specific filter that can absorb pollutants, and then use a balance to calculate the change in mass of the filter to determine the amount of pollutants in the air(Ji, 2014). The calculation formula is given in Equation 2-1:

$$C = \frac{W2 - W1}{V}$$

Equation 2-1

Where: C is the concentration in ug/m^3,

W2 is the mass of the filter after the sampling in ug

W1 is the mass of the filter before the sampling in ug

V is the volume under standard state (101.325 kPa, 273 K) in m^3

A typical application of this method is bucket detection to collect the concentration of PM2.5. The benefits of the gravimetric method are simple, reliable, and the detection cost is low, but it requires a lot of human intervention, and the program is complex and cannot display real-time data.

Optical method:

The optical method is based on the principle of laser light scattering. When the laser from the sensor is scattered by the pollutant in the air, the sensor can obtain a change curve according to the change of the scattered light intensity. Combined with the algorithm of Mie theory, it is computes the equivalent particle size of the particles and the number of particles in different sizes per unit volume (Yong and Haoxin, 2016). The advantages of optical sensors include the ability to simultaneously detect the concentration of several pollutants in the same band. The sensitivity of the sensor is high. Typical applications of the optical method are PM laser sensors and CO laser sensors.

Electrochemical method:

This method uses the principle of electrochemistry to detect the presence of CO, SO_2, NO_2, O_3 and other gases in the air. The internal resistance of the sensor changes as the pollutant in the air changes, and the concentration of the pollutant is calculated by detecting the change in the resistance. A typical application of this method is the MQ sensors. This method has the characteristics of low power consumption, long service life, and low cost.

Radiation method:

The radiation method utilizes the principle of β-ray attenuation. The sampling gas is sucked into the tube by a pump and discharged through a filter, so that the particles can be deposited on the filter. When the β-ray passes through the filter, the energy of the β-ray will be attenuated, and the concentration of the pollutant can be calculated by measuring the amount of β-ray attenuation. The advantage of this method is that the volume of sample gas needed is small, and the concentration of the sample can be reacted in real-time, but the disadvantage is relatively high cost (Xingyu, 2016). Other common methods include titration analysis, spectrophotometry, potentiometry, chromatography, microbiological methods, and other instrumental analysis methods. These methods belong to the traditional detection methods, they take long detection time. Usually these detection methods are also complicated, and they are more suitable for laboratory testing not for the atmospheric environment detection.

2.4.3 Common Data Dissemination Methods

In addition to the Wi-Fi and Ethernet that have been commonly used in existing projects, Bluetooth, GSM/GPRS, LoRa, Zigbee, are other data dissemination methods which have gained popularity. Wi-Fi and Bluetooth are the two most popular communication methods. Wi-Fi is a technology that allows electronic devices to connect a network without physical wired connection by using radio frequency. Wi-Fi is suitable for large data volume transmission. It has been widely used at home, school and so on. Bluetooth is mainly used for short-distance data exchange by using short-wavelength radio transmissions, such as personal wearable devices. The use of these two technologies in mobile phones and notebooks is quite mature.

GSM/GPRS

Usually, the GSM module integrates a GSM radio frequency chip, a baseband processing chip, a memory, a power amplifier device on a circuit board. It is a functional module with an independent operating system. The GSM module has all the basic functions available on the GSM network such as SMS messages, voice calls, GPRS data transmissions. Developers can implement various wireless communication functions by using the standard AT command to control the GSM module. For instance, data transmission could be through SMS, TCP and UDP (Elecfans, 2018). It is a perfect addition model for the networking connection when there is no Wi-Fi connection available.

The early GSM module was mainly used by mobile phone manufacturers. The mobile phone factory purchased the module directly and equipped with peripheral devices to produce a mobile phone (Wu, 2014). Presently, the GSM module is used in a wider range of fields, such as the GSM-based wireless alarm control system. In the event of an accident occurring in the user's home, a fire outbreak, an illegal intrusion, etc., the GSM intelligent controller will automatically send corresponding alarms to all valid mobile phones through SMS and automatically alert the appropriate government emergency agencies. Some other applications areas include in power and water system where the consumption readings can be collected through the GSM module, which monitor the user's electricity consumption and water consumption in real time (Letswireless, 2018).

LoRa:

LoRa (LoRa Alliance, https://lora-alliance.org) is a low-power LAN wireless standard created by Semtech. The advantages of LoRa's direct-sequence spread spectrum technology is not only that it consumes low power, but that it has farther transmission distance than other wireless methods. It also has strong anti-interference ability and excellent anti-multipath fading performance. Its low power consumption feature is ideal for battery-powered projects. The maximum transmission distance can be up to 20 kilometres to form a wide range of message transmission. Because of its low hardware cost, it is suitable for small data volume but large-scale transmission applications. For example, in remote areas, LoRa can be used to form data transmission between devices in the local area network and transmitted to an external Internet network. From where the data of the entire area of data is aggregated for overall analysis(Ebyte, 2018b).

Augustin *et al.* also verified in their experiments that LoRa technology is suitable for low-power, low-throughput and remote network projects in their paper "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things". In their experiments, their equipment provided a stable network covering 3 kilometres for dense residential suburbs(Augustin *et al.*, 2016).

Zigbee:

Zigbee technology is a two-way wireless communication technology with features of close proximity, low complexity, low power consumption, low speed and low cost. It is mainly used for data transmission between various electronic devices with short distance with low power consumption. According to ON World analysts, Zigbee-based products will account for up to 85% (3.8 billion) of the total 4.5 billion cumulative shipments IEEE 802.15.4 devices in 2023(Ebyte, 2018a).

Compared to other IoT wireless technology- Low Power Wide Area Networks (LPWANs), such as LoRa networks, use a typical centralized network structure (i.e. a star network), Zigbee's advantage is the use of Mesh networks. Compared with the traditional Wireless Local Area Network (WLAN), any wireless device in the Mesh can serve as both an access point and a router. Each node in the network can send and receive signals, and each node can communicate directly with one or more nodes. The biggest benefit of this architecture is that if the nearest node is not available, the data can be automatically rerouted to another neighbouring node for transmission until the final destination is reached. Compared with traditional WLAN, wireless Mesh network has several unparalleled advantages such as easy deployment and flexible structure. In a wireless Mesh network, if it is needed to add a new device, it only needs to simply connect the new device to the power supply. It can automatically configure itself and determine the best multi-hop transmission path(Ebyte, 2018c).

A typical application is Ngandu's "Smart Meter Data Collection Using Public Taxis" published in 2018, which describes a mobile data collection system based on Zigbee communication technology that can be used in mobile buses or taxis to collect reading of electricity meter. The results show that when the taxi is driving with the device at a speed of 5 km / hour, the device can collect data from a distance of up to 172.56 m, and when the taxi is driving at 60 km / hour, the device can collect data from a distance of up to 107.25 m(Ngandu, Ouahada and Rimer, 2018). Although the transmission distance is not very long, it shows the potential of Zigbee being used in mobile development.

2.4.4 Instant messaging protocol Message Queuing Telemetry Transport (MQTT)

Froiz-Míguez used MQTT, Zigbee and Wi-Fi to design and implement an IoT home automation system for fog computing applications(Froiz-Míguez *et al.*, 2018). He pointed out that MQTT consumes very little computing resources and solves most of the compatibility issues in home automation which are mostly agreements, technical and related standards. In fact, as early as 1999, Andy Stanford-Clark of IBM and Arlen Nipper of Arcom invented MQTT technology. The technology supports all platforms, it can connect almost all internal networked devices to the outside Internet, and it is used as a communication protocol for sensors and drives. The MQTT is also known as the SCADA protocol (Supervisory Control and Data Acquisition), it has now become an important part of the IoT protocol

The biggest advantage of MQTT is that it provides real-time and reliable messaging services for connecting remote devices with very little code and limited bandwidth. As a low-overhead, low-bandwidth instant messaging protocol, it has a wide range of applications in IoT, small devices, mobile applications, smart homes and so on. The MQTT protocol is a protocol designed for the communication of remote sensors, controlling devices with limited computing power and works in low-bandwidth and even unreliable networks. Its main features according to (Locke, 2010) include:

5. Using the publish/subscribe message mode to provide one-to-many message publishing method
6. Providing network connection using TCP/IP
7. Small transmission, low overhead, protocol exchange is minimized to reduce network traffic

2.5 The main work of the research

From the literature review above, it can be seen that some of the research gaps found in the state-of-art pollution monitoring include

- Rigid data dissemination as a result of using only one communication protocol for pollution data dissemination
- Scarcity of low-cost off-the-shelf devices that can be used to complement local/regional pollution monitoring stations.

Based on the shortcomings of some existing research projects, this research proposes to use IoT development platform to find some effective solutions to solving them. For example, the accuracy of sensors for atmospheric monitoring is often higher than that of home monitoring. Therefore, for different scenarios, the sensor part should be designed as a separate module so that it can be better replaced, added or removed in the future. In addition, most of the current detection devices only have a single transmission mode through Wi-Fi or Ethernet. In order to reduce this limitation as well as increase the coverage, a variety of data communication methods suitable for data transmission at various distances are introduced. The specific hardware and software design of this research are:

1. Selecting hardware development platform
2. Finding high-precision sensors with low price
3. Designing hardware systems to ensure that devices can be used regardless of the Internet access
4. Modular design, system is able to replace different precision sensors for different inspection scenarios
5. Introducing network selection to increase multiple intelligent data transmission methods
6. Real-time data push

2.6 Summary

Air pollution is a serious problem of global scale, in developing countries in particular, the lack of corresponding air information monitoring equipment has brought many management difficulties to the countries. This chapter presented the current status and development trend of air pollution by referring to existing literatures. This chapter focuses on the analysis of several existing projects, their methodologies as well as advantages and disadvantages. It was also shown that combining professional detection station with sensor-based equipment is a feasible development direction of a low-cost air monitoring program. This chapter also describes some related technical support to provide some reference for the research system design of the next chapter.

3. Research Method

This chapter details how the sensor method helps to develop a system that can be used for air pollution monitoring. A sensor-based device was designed to detect the air quality data. The air pollution parameters include: CO, temperature, humidity, PM. In order to be used in a variety of application scenarios, the device's data transmission methods include Ethernet, Wi-Fi, GSM/GPRS, LoRa, Zigbee, and Bluetooth. The research method is divided into three primary phases: designing and developing the hardware and software of the system, collecting air quality data by using the system, and evaluating the system by comparing and analysing the data collected with the current system.

3.1 Requirements

3.1.1 Usage Scenarios

As individuals and families pay more attention to the environment, the demand for one-stop air quality monitoring services is increasingly urgent. The future development trend of the air quality monitoring services will or already naturally entered our lives the way weather forecasts already have. Therefore, an air pollution detection device not only needs to help users monitor the quality of their environment in real time, but also needs to consider portability, accuracy and the possibility of multi-scenario operations.

In the system design requirements, it is necessary to make a correct judgment on the use of the sensors. According to the knowledge from sensor supplier WinSensor, the application fields of the sensor are mainly divided into three categories(Winsensor, 2012):

1. Atmospheric monitoring sensors, mainly used as analytical instruments in environmental monitoring stations, environmental testing or calibration and research institutes.
2. Industrial environmental monitoring sensors, suitable for indoor and outdoor environmental monitoring at the job site. For example, the detection of pollutant content around the factory.
3. Civic environmental monitoring sensors, mainly used for indoor and outdoor residential buildings or shopping malls, stations, airports, schools, roads and other common commercial places.

However, no matter the type of sensor used in the air quality monitoring system, the system should have the ability to manage and control each monitoring equipment. It should be able to store the data collected by the monitoring equipment, such as industrial pollution, traffic pollution and particulate matter. In addition, the system should be able to access relevant data from other systems, such as meteorological data, traffic data and even demographic data. After integration and analysis, the obtained result should form the application foundation of big data to support all-round and refined environmental applications and realizes real-time monitoring, management and governance of air pollution. Based on the above application requirements, this paper proposes the system architecture depicted in Figure 3-1.

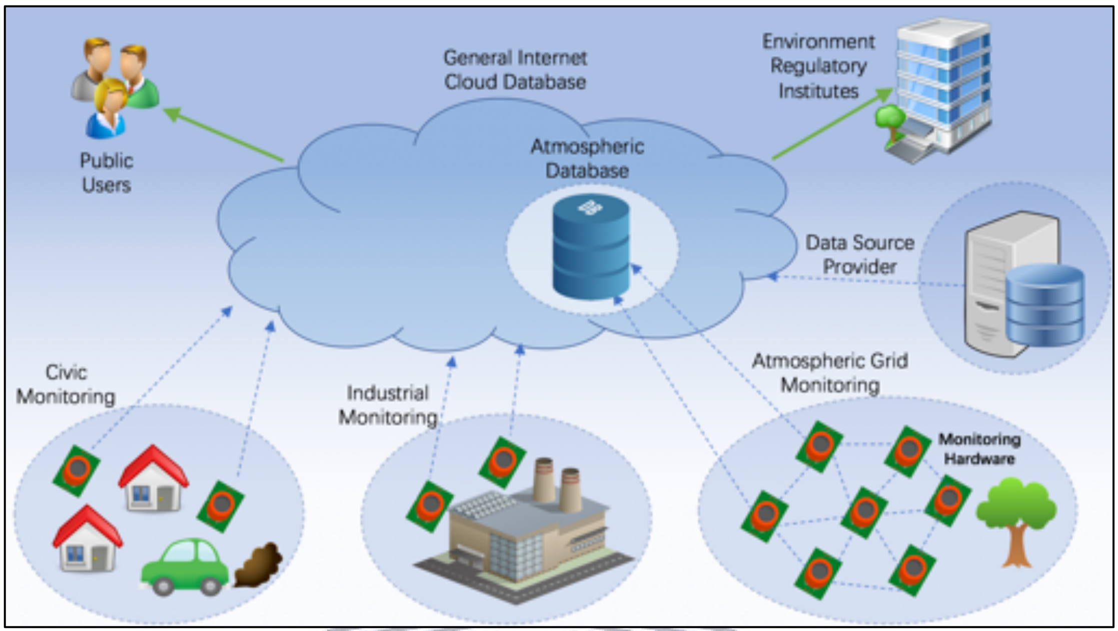


Figure 3-1: System Architecture

As a civic monitoring equipment, the system can feed users with real-time indoor air quality information, enabling users know if their homes are well ventilated. When the equipment is placed outdoors, it can detect the surrounding environment and enable users better understanding of the outdoor air quality and determine if the weather is suitable for outdoor activities or not, etc. When the equipment is used for industrial purposes, such as in factories, it allows detecting if the factory emissions meets or exceed the regulatory standards. When the emissions exceed the standard or in events of dangerous gas leakages, the equipment can timely report the data to the factory management or send a message to notify surrounding users. When the equipment is used for atmospheric monitoring, multiple detection equipment needs to be deployed within a certain range to collect the data for a large area, such a city and country. The collected data will be uniformly uploaded to the cloud consisting a server with a database. Data processing is performed by the server and data pushed to users and related scientific research institutions. On the server side, the system can simultaneously introduce big data from other fields and perform various analysis and processing on the overall data combined with user requirements to meet different user needs. For example, machine learning can be introduced on the server side to learn the trend of air quality, as well as evaluate and predict the future trends of the air quality. At the server side, data about the physical health status of the population in the designated area can be analysed and inferences made about the relationship between specific air pollutants on certain diseases.

3.1.2 Requirements Analysis

The general aim of this research was to stress the relevance and value of low-cost wearable and plugin technologies for the delivery of an increasingly flexible environmental and healthcare monitoring system, the economic importance and privacy concerns. For the traditional monitoring equipment, the method used to send data is generally through Wi-Fi or USB cable, such as the one for SAAQIS. However, the developed equipment was designed as a portable device, as the users do not want it to be limited by the use of a cable to send data all the time. Thus, it is important to develop a method that will work with this specific equipment and send data from the equipment to the system database without cables wirelessly. There are many methods that have been helped to send data to a remote server through methods such as Wi-Fi, Bluetooth, radio, infrared ray, microwave and so on. It is important to select an appropriate method for transmitting data based on the specific needs of the system. Ideally, the developed equipment should offer more than one method to send data in reality, as the device will function in a number of different environments and it may be the case that one of the methods is not suitable for a particular environment or it loses its efficacy for some reasons. Thus, the developed equipment needs to have an adequate number of methods that can suit a number of different scenarios to maximize efficiency and applicability of the system. The developed equipment was designed to test air pollution both indoor and outdoor and the aim of the research is to collect information pertaining to air quality, the developed equipment has to measure as much as it can to get an accuracy result to present to the users. And because of the quick changing of the weather and that huge amount of data, so that is why choose a good method to send data is so important for the system. That is where the network selection algorithm comes in the research.

When the data has been collected, the equipment needs to analyse what is the network environment it is in and also consider how big of the data has been collected to choose the method to transmit data automatically or manually someway. For example, in the environment of Wi-Fi the equipment will send data through Wi-Fi networking rather than GSM/GPRS, because the data transfer rate is significantly higher when using Wi-Fi. When the equipment is in an environment without Wi-Fi, then the device will send data through GSM/GPRS if the data is not that big. But choosing the best method to send data that works well with different environments still needs us to run a lot of tests and combine different actual situations.

3.1.3 High Level Constituent Parts

The requirements can thus be summarized into three points, which are:

1. The developed monitoring equipment should have a high degree of accuracy, but the price of the overall equipment should be acceptable.
2. The sensor part of the developed monitoring equipment can be replaced with different other sensors in line with the need to detect different pollutants. When used to detect the same pollutants, the sensors should be interchangeable based on the different precision levels required.

3. In order to meet the requirements of the grid monitoring and different scenarios, the developed equipment needs to have multiple means of data transmission.

In order to better realize the development of the system, the system was built on a multi-layer framework IoT depicted by Figure 3-2.

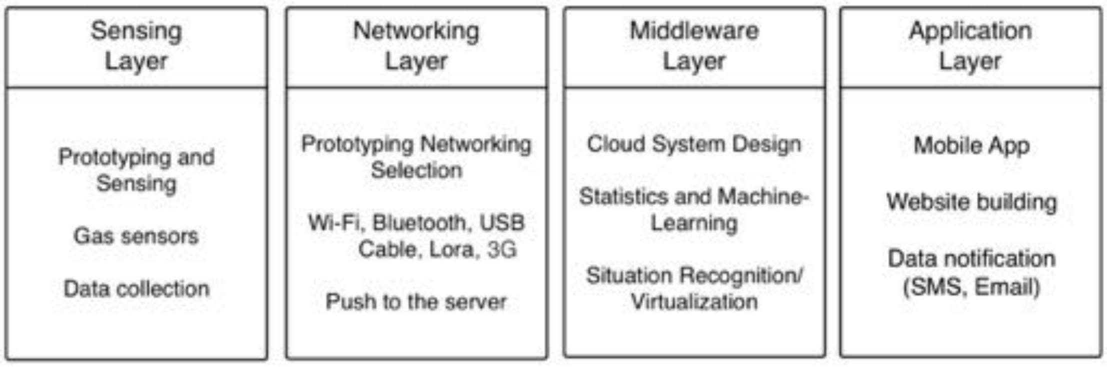


Figure 3-2: Adopted IoT Framework

This framework enables breaking down the whole system into four layers which highlight the paradigms, the software and the hardware systems that best fit the user requirements. While each layer will be developed separately from the other, the final product will integrate the constituent component of each layers in a unique design that addresses the user requirements.

The framework and project are structured into four layers which together form the prototype being developed.

A. The first layer is the sensing layer which senses the pollution in the air. The received data is then transferred into a computer or similar device for analysis. This is where the networking layer comes in.
B. The networking layer collects data from the sensing layer and passes it to the middleware layer. The networking layer will use network selection algorithm to choose the best communication method to send data while also ensuring quick and successful delivery to the next layer.
C. The middleware is mainly responsible for data storage and analysis before being passed to the application layer.
D. Application layer presents the data to the users.

In line with the objectives, this research will primarily focus on the design and development of data collectors and transmission, which is the development of sensing layer and networking layer. The latter parts (data analysis at the middleware and presentation at the Application layer) are outside the scope of this work. They are however discussed in a related work titled “A Decision support system for accessing the impact of air pollution on human health”(Mandava, 2019).

3.2 Design

3.2.1 Overall design ideas

In order to achieve low cost goal of the system, there is a need to develop a new hardware equipment, using the idea of IoT technology with the combination of microcontroller and sensor modules. The hardware equipment needs to have its own programs to collect data. It is also necessary to build a server that receives the data at the backend. A small simulation experiment will be tested first, then more placements of hardware prototypes can be set in the city to build a monitoring network in order to collect more data to prove the efficiencies of the system prototype in providing data support relating to air pollution.

As discussed in section 3.1 the research has been separated into four layers. The main task of the sensing layer is to collect data and the networking layer is mainly responsible for the transmission of data to the server. These two layers are directly in contact with the physical / hardware component of the project. The development of the new hardware will mainly focus on these two layers and which integrate with the IoT development board as shown in Figure 3-3.

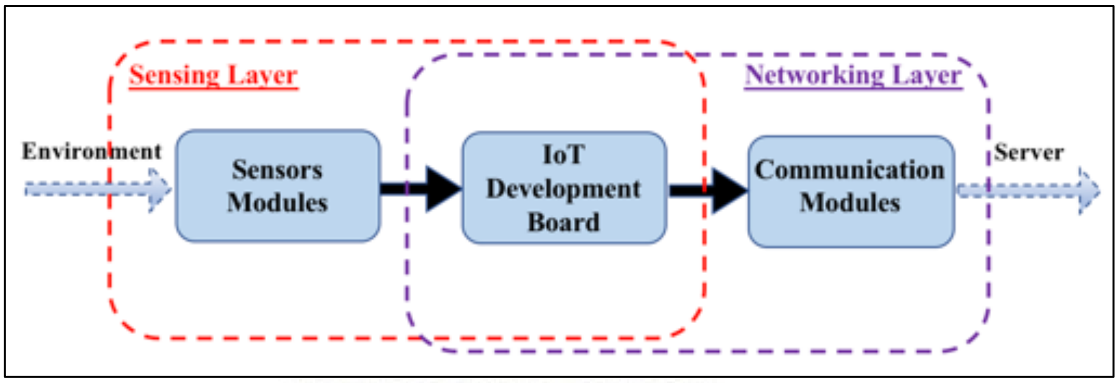


Figure 3-3: Hardware overall design

For the software, there are three main aspects, which are listed below and depicted in Figure 3-4:

1. Hardware equipment software which run on the sensing and networking layers and are responsible for data collecting and data transmission.
2. Server software which belongs to the middleware layer and responsible for receiving and analysing data at server side.
3. Mobile software which belongs to application layer and is responsible for displaying the data to the users.

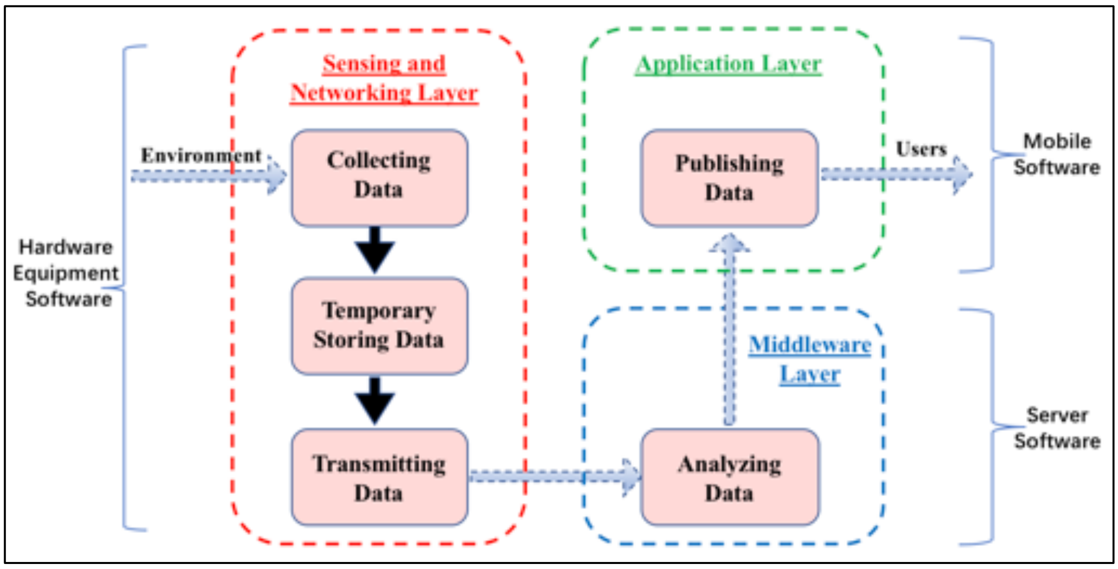


Figure 3-4: Software overall design

3.2.2 Choice of Hardware Development Platform

Common hardware development platforms

With the development of integrated circuit technology, the size of computers is also shrinking. In the rapid development of the IoT and integrated circuits, some different hardware development platforms have been derived based on different hardware products. The most common development platforms include: Arduino, Raspberry Pi, BeagleBoard and so on.

Arduino is an easy-to-use open source electronic prototyping platform developed by Massimo Benzi and his team in 2005 (Baidu, 2018a). Arduino has a rich interface such as digital I/O port, analog I/O port, it also supports SPI, I2C, UART serial communication. It is very compatible with a wide variety of sensors, lights and motors. In addition, the Arduino platform encapsulates many functions and a large number of sensor function libraries, the development difficulty is greatly reduced. The Arduino platform consists of two main parts: the hardware part and the software part. The hardware part is the Arduino board can be used for circuit connections and can be connected to a wide variety of sensors. The software part is the Arduino scripts, which are used as a development environment for Arduino programs in computers. The Arduino IDE was developed based on the processing IDE. The programming language is based on the wiring language and it is a secondary encapsulation of the avr-gcc library. For beginners, it is easy to master, and it has enough flexibility to be developed quickly without much MCU basics and programming foundation. After the program code written in the IDE on the computer, it only needs to be uploaded to the Arduino board, then the Arduino will execute the corresponding instructions according to the program.

The Raspberry Pi was developed by the Raspberry Pi Foundation, a charity registered in the UK, and officially launched by project leader Eben Upton in 2012 (Baidu, 2013). The Raspberry Pi is also known as a card computer. Its original intention was designed for students' computer programming education. The shape of the Raspberry Pi board is only credit card size, but it has the full basic functions as computers. Raspberry Pi is an ARM-based microcomputer with SD card as the memory disk. There are several USB ports and one network port on the motherboard, which can directly connect the keyboard, mouse and network cable. It has HDMI high-definition video output interface as well, just turn on the TV and keyboard, you will be able perform functions such as spreadsheet, word processing, playing games, playing HD video and so on. The Raspberry Pi system is based on Linux, which means developers are also able to develop in their most proficient languages (like Python, Java, etc.) and familiar libraries.

BeagleBoard is a low-power open source single-board computer produced by Texas Instruments in conjunction with Digi-Key and Newark element14 (Baidu, 2018b). As early as 2008, they launched the first BeagleBoard, followed by BeagleBoard-xM, BeagleBone Black and other versions. BeagleBoard is designed as an open source software development platform that is widely used by universities around the world for open source hardware and software functions development. In terms of hardware, there is not much difference between the pattern of BeagleBoard and the Raspberry Pi. The BeagleBoard community also provides a number of hardware expansion modules (Capes) to make it more powerful. At the software level, BeagleBoard comes with Linux and is compatible with other versions of Linux such as Android and Ubuntu. BeagleBoard also comes with its own scripting language BoneScript, which is easy to develop and compatible.

Comparison of the hardware development platform

The reason for choosing the above three development platforms is because they are very easy to obtain, affordable and similar in the size. More importantly, they can all be used in the hardware development of this research. Here is a rough comparison with the common boards of each platform, namely Arduino Uno.

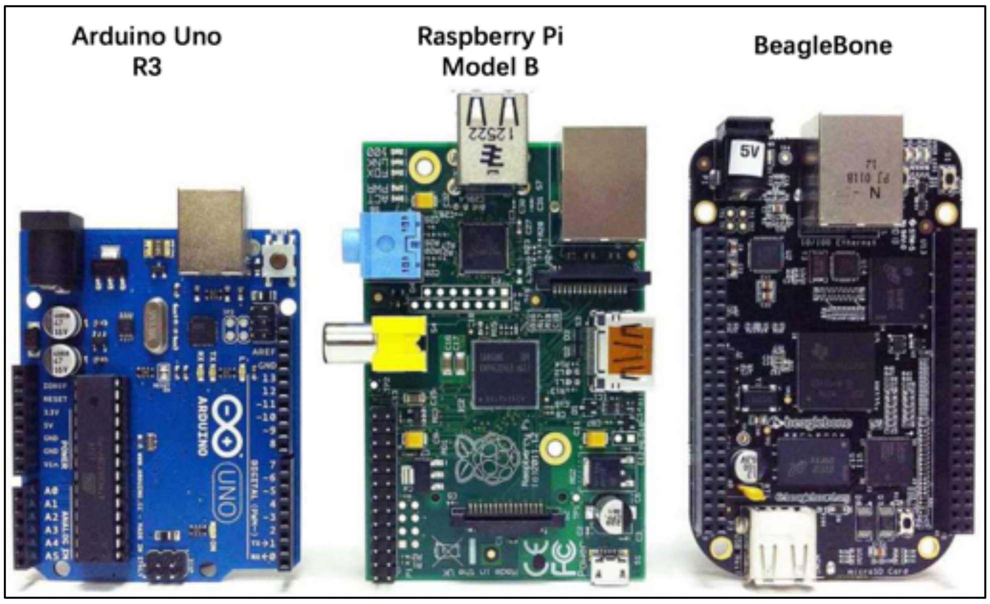
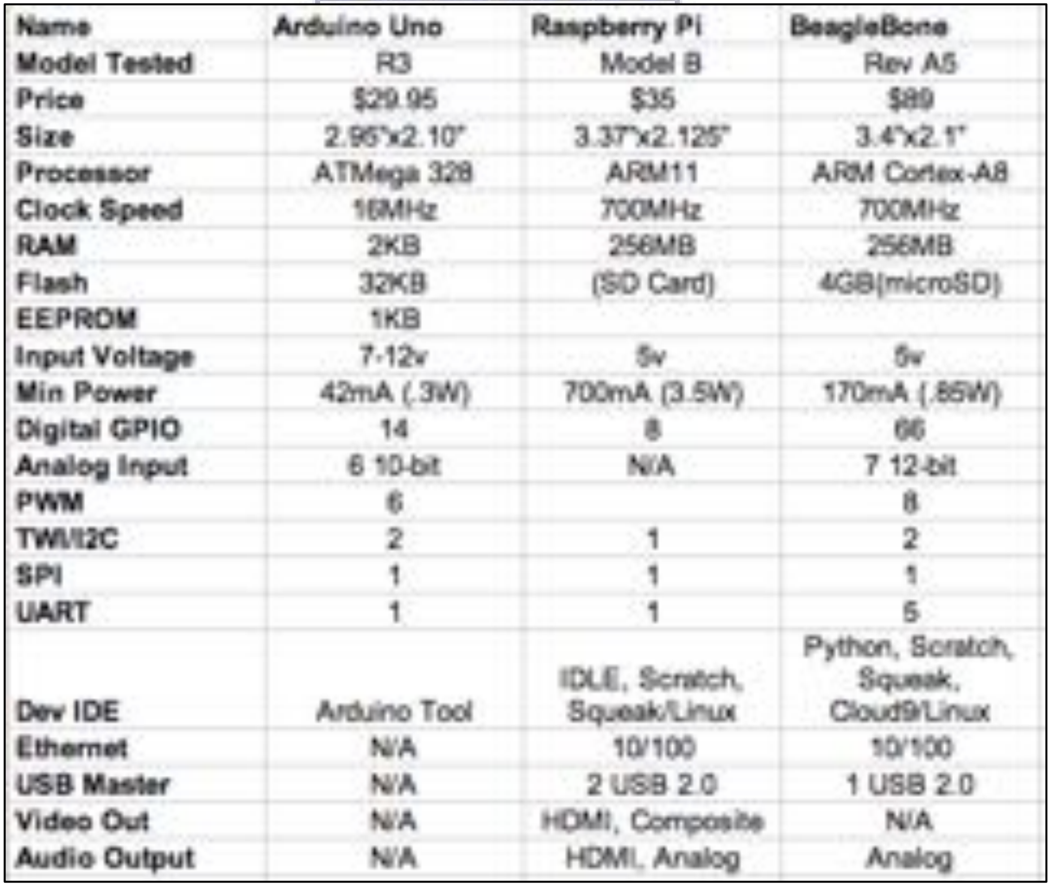


Figure 3-5: Outlook comparison of Arduino, Raspberry Pi and BeagleBone (Smythe, 2017)

Table 3-1: Features comparison(Kithion, 2016)



Name	Arduino Uno	Raspberry Pi	BeagleBone
Model Tested	R3	Model B	Rev A5
Price	$29.95	$35	$89
Size	2.95"x2.10"	3.37"x2.125"	3.4"x2.1"
Processor	ATMega 328	ARM11	ARM Cortex-A8
Clock Speed	16MHz	700MHz	700MHz
RAM	2KB	256MB	256MB
Flash	32KB	(SD Card)	4GB(microSD)
EEPROM	1KB		
Input Voltage	7-12v	5v	5v
Min Power	42mA (.3W)	700mA (3.5W)	170mA (.85W)
Digital GPIO	14	8	66
Analog Input	6 10-bit	N/A	7 12-bit
PWM	6		8
TWI/I2C	2	1	2
SPI	1	1	1
UART	1	1	5
Dev IDE	Arduino Tool	IDLE, Scratch, Squeak/Linux	Python, Scratch, Squeak, Cloud9/Linux
Ethernet	N/A	10/100	10/100
USB Master	N/A	2 USB 2.0	1 USB 2.0
Video Out	N/A	HDMI, Composite	N/A
Audio Output	N/A	HDMI, Analog	Analog

From Table 3-1 above, it is can be seen that the price of Arduino Uno and Raspberry Pi model B are relatively cheaper compared to the BeagleBone. In terms of interfaces however, neither of the two is as rich as BeagleBone. From the performance point of view, although the price of BeagleBone is much higher than that of the Raspberry Pi model B, the performance of both boards are similar while the arithmetic performance of the Arduino Uno is the slowest. Both the Arduino Uno and the BeagleBone have analog and digital signal interfaces, which makes the two boards more convenient to connect sensors with analog output. However, the hardware platform finally adopted by this project is the Raspberry Pi, due to three factors:

The first is the cost. From the initial intention of the research, which is to develop a relatively inexpensive and reliable air quality testing platform. Cost should be considered as a more important reference factor. Although BeagleBone may have absolute advantages in terms of functionality and performance, from a cost perspective, the Arduino and Raspberry Pi are more suitable as a development platform for this research.

Second is the flash memory. The flash memory size of Arduino Uno is only 32KB, but the flash memory size of Raspberry Pi model B and BeagleBone are dependent on the SD card mounted on them, with both able to support between 2GB to 16GB. Building on the previous research “Internet-of-Things for Air Pollution Monitoring in Smart Cities: An Efficient Network Selection Model” published in 2018 (Chen *et al.*, 2018), we proposed a networking selection model, which was originally based on the Arduino Uno board, but was later replaced by a larger memory Arduino board Mega(256KB) for development. The model contains the Arduino Mega board, Bluetooth, Wi-Fi module and some LED lights. The external libraries used occupied most of the memory space at final stage of the project. A key factor for this research work is to develop a hardware that supports multiple sensor at the same time in order to cater for more scenarios. Multiple sensors invariable means more sensors driver program would need to be sorted in memory. An Arduino with its fixed memory size will therefore be insufficient for this research work. In addition, the hardware will need a larger memory in order to temporarily store the collected data as well. Though, the Arduino can be fitted with an SD card shield to provide access to more storage it is less convenient than using the Raspberry Pi and BeagleBone which both come fitted with an SD card slot. The development cost also becomes invisibly higher if develop based on Arduino.

Thirdly, is the multiple tasking requirement. According to the software overall design, the system will have at least two programs running in the hardware, the first for data collection, and the other for data transmission. It takes a certain period of time for sensors to collect data and data transmission to the sender. In order to improve the efficiency of the system, the two programs will need to run simultaneously, which means that the system needs to have the ability to execute multiple tasks at the same time. Both Raspberry Pi and BeagleBoard are Linux-based systems. Being small computers, they can run multiple programs, and support multi-language programming. The Arduino’s design on the other hand is very simple and does not have an operating system. It is also limited to running just one program at a time and also supports low-level C++ like language.

Considering the above factors and due to the price and full functionality, the Raspberry Pi was selected as the development platform for this work. However, because the Raspberry Pi does not have analog pins, it will be difficult to control sensor or motor on it, therefore an Arduino board is also used to partially assisted the Raspberry Pi.

The design was inspired mainly by the AsthmaPi kit prototype designed and launched by Arnav Sharma in 2016(Tech4Good, 2016). He uses the Raspberry Pi and a series of sensors to help diagnosed people with asthma and preventing in advance, and when the environmental air is in danger the AsthmaPi can promptly warn the users. He won the Tech4Good Winner of Winners Award 2016 for this design. Arnav's unit uses two Raspberry Pi, one is for displaying data and the other one is for detecting air quality, where the air detection sensor includes an MQ-135 gas sensor and an optical dust sensor. The MQ-135 sensor uses Analog to Digital Converter (ADC) chip to connect communicate with Raspberry Pi, and the optical dust sensor uses the Arduino UNO to pass the data to the Raspberry Pi via USB. The cost of the sensors are very low, but it can effectively achieve the fluctuation of air pollution.

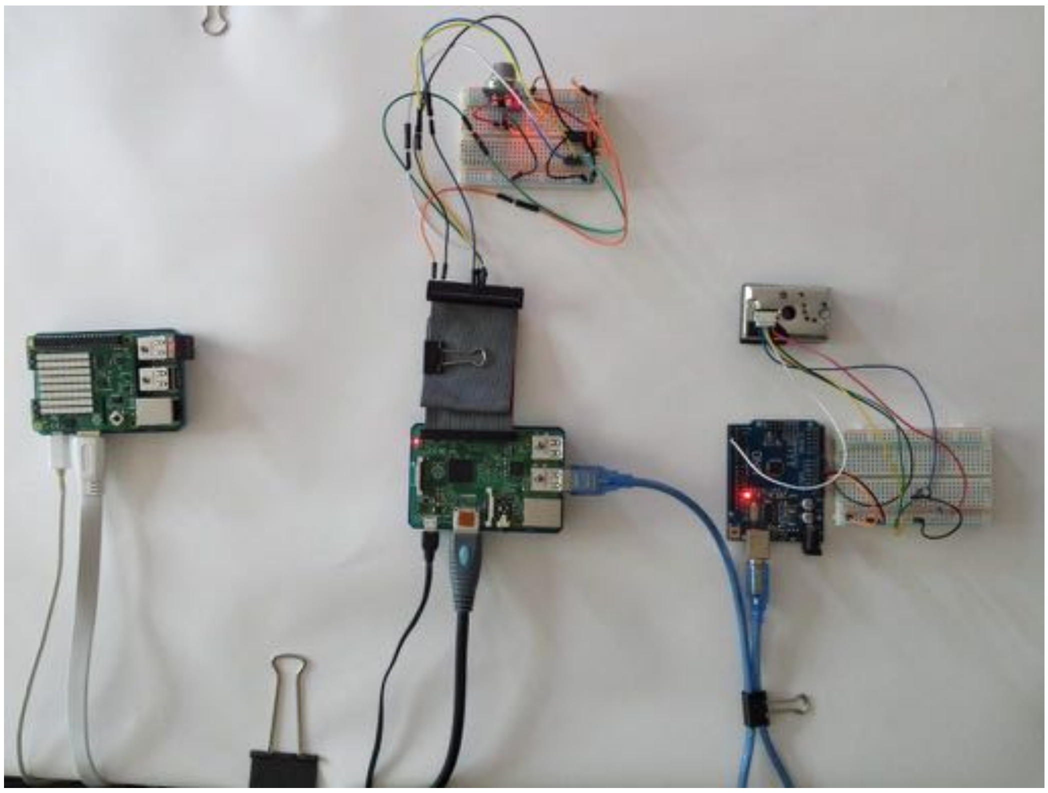


Figure 3-6: Hardware connection of the AsthmaPi(Tech4Good, 2016)

3.2.3 User Interface Specification

Interfaces of sensing layer and networking layer

Since the sensing layer and networking layer are both developed in the hardware, they will be considered as one part, consisting of three interfaces. The first one is the interface to the sensors to collect data, the second one is the interface to the Middleware Layer to transfer the data and the last one is an admin interface. The admin interface is only used for debugging and maintenance. The admin interface will be designed as a visual interface running on a computer or smart phone for the tracking and operation of data transfer processing. The main function of this interface is to manually intervene the data transmission and network selection.

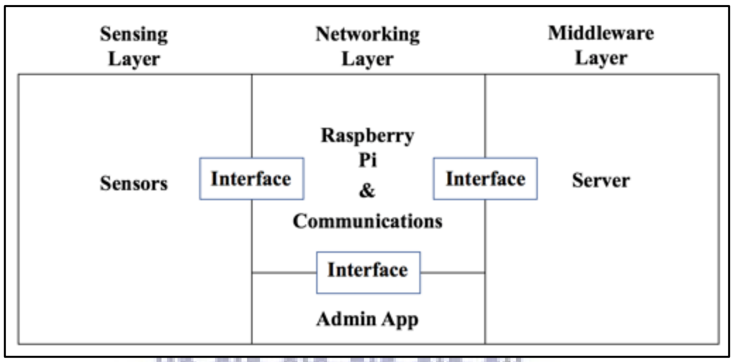


Figure 3-7: Interfaces of sensing layer and networking layer

The admin interface will eventually be presented in the form of a mobile application that can run on an Android smart phone and using some simulator running on a computer through Bluetooth. The mobile application is going to have the functions of displaying the current state of the hardware equipment and some buttons that helps the admin user to connect smart phone with hardware equipment through Bluetooth. Users will be able to choose the different communication method for data transmission and also be able to enter some commands in the application to control the hardware equipment.

The admin interface shows the current status of the system in an efficient manner to the administrators. Through it, the administrators can observe and adjust options in a timely manner. At the same time, from the interface, the network selection precedence can be set. For example, when the Ethernet cable and Wi-Fi are both connected, but the Ethernet connection is not stable due to various reasons, the administrators can manually select Wi-Fi connection as the means of sending data. This design will make the system smart enough to meet different needs.

Interfaces of middleware layer

The middleware layer has two interfaces. The first one is an interface to the air quality monitoring equipment (which is the hardware developed in this research work); while the second one is an interface to the application layer. The middleware analysis data and pushes it to the users. Being a modular design, on completing the system, more interfaces can be fitted, such as those for accepting data from other systems.

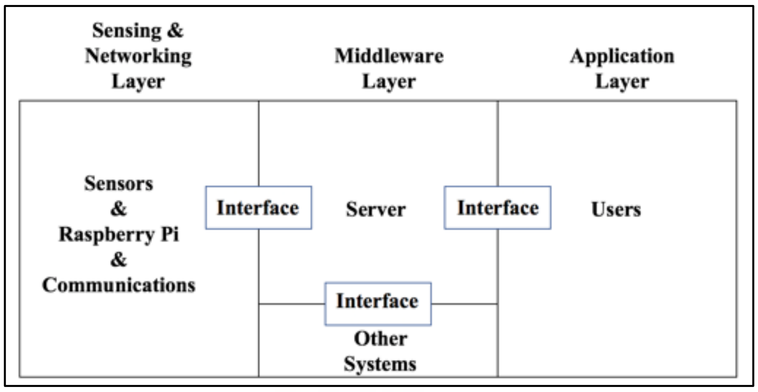


Figure 3-8: Interfaces of middleware layer

The interface to the hardware equipment contains multiple components, such as internet connection, LoRa connection, Zigbee connection, etc. A variety of connections will help the system to better collect data from different scenarios. The interface to the application layer is mainly through the Internet to push data to application layer.

Interfaces of application layer

The application layer also has two interfaces. The first is an interface to the middleware layer which connects to server and receives data from the database. The second is the interface to users, which displays the request data through mobile phones and website. For testing purposes this research work only used mobile phones application.

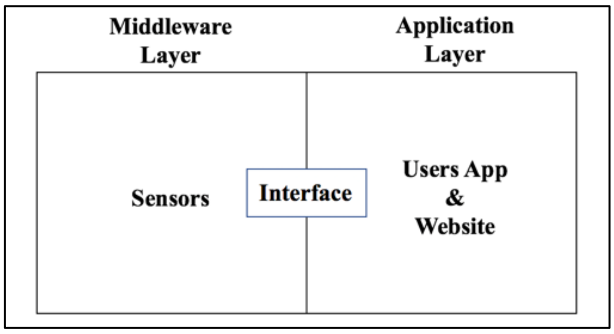


Figure 3-9: Interfaces of application layer

The interface of the application layer to the middleware is mainly through the Internet. Being an application running on smart phones, access would be through 2G/3G/4G/LTE and Wi-Fi connections. The interface to the users in this research is going to use some existing software platforms such as such as Telegram, WhatsApp and WeChat to reduce development costs,. With these being well-developed platforms and usable on both Android and Apple devices, this will greatly reduce the learning costs and make the pushed data more acceptable and usable.

3.2.4 High level Design

The pictorial high level design of the work is shown in Figure 3-10 below. There are seven units in the design, which are: the sensors, Arduino, Raspberry Pi, connection methods, the admin app, the server and the user app. The sensors, Arduino, Raspberry Pi are in the sensing layer, the admin app and communication links are in networking layer, while the server/database belongs to the middleware layer and the user mobile app belongs to the application layer. The communication methods that is going to be applied in the design are Bluetooth, Ethernet, Wi-Fi, GSM/GPRS, LoRa and Zigbee.

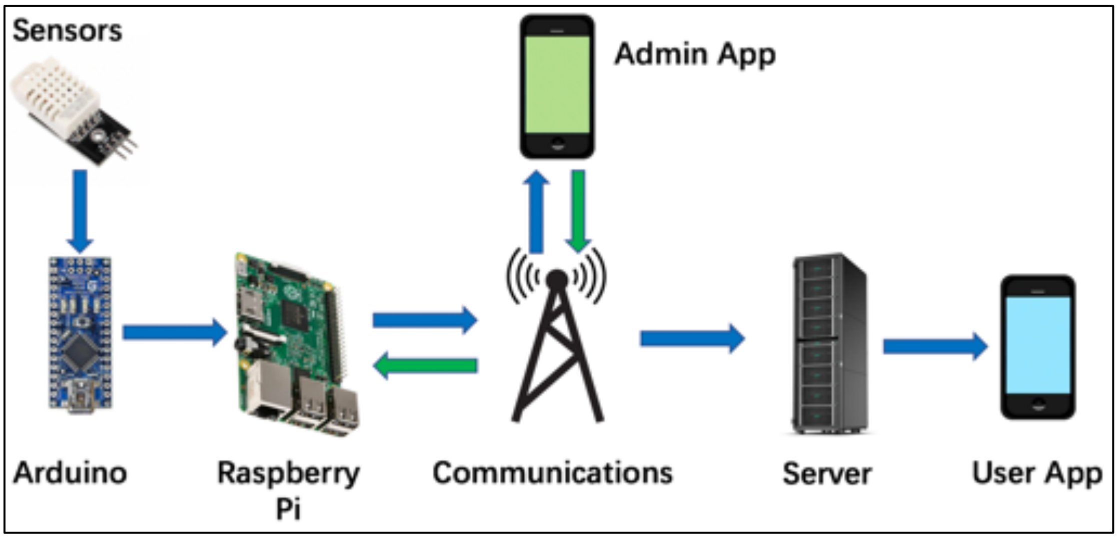


Figure 3-10: High level design

Details of Technical Solutions in subsystems

Raspberry Pi 3 model B

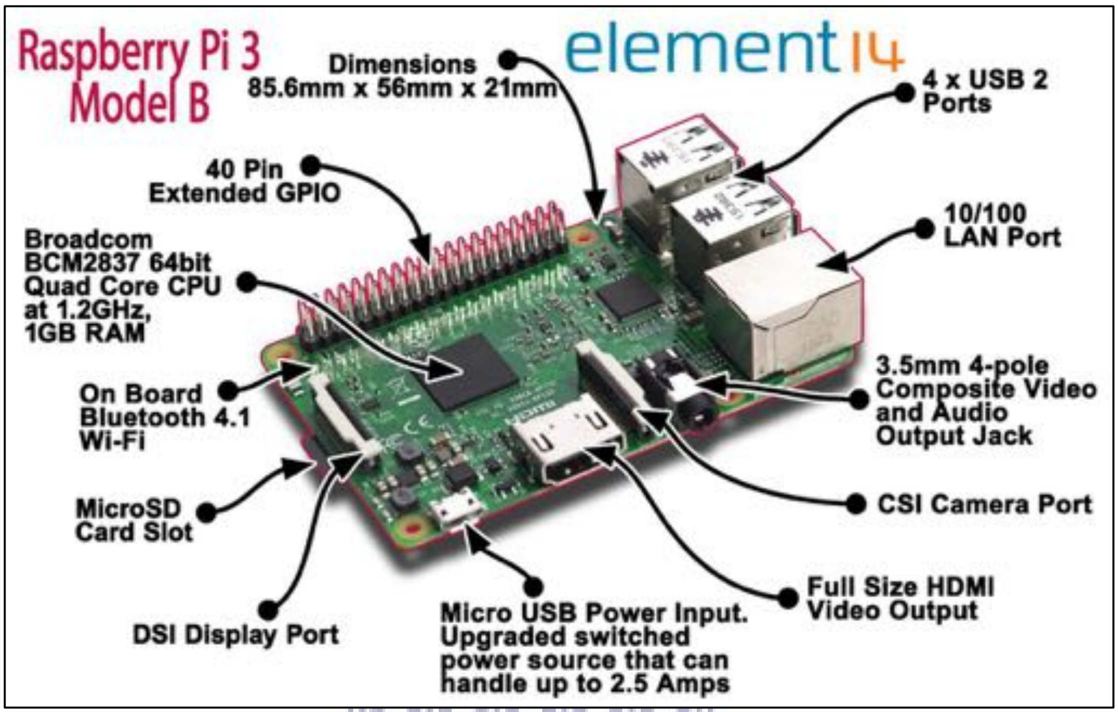

Raspberry Pi 3 Model B
element14
Dimensions 85.6mm x 56mm x 21mm
4 x USB 2 Ports
40 Pin Extended GPIO
10/100 LAN Port
Broadcom BCM2837 64bit Quad Core CPU at 1.2GHz, 1GB RAM
3.5mm 4-pole Composite Video and Audio Output Jack
On Board Bluetooth 4.1 Wi-Fi
CSI Camera Port
MicroSD Card Slot
Full Size HDMI Video Output
DSI Display Port
Micro USB Power Input. Upgraded switched power source that can handle up to 2.5 Amps

Figure 3-11: Specifications of Raspberry Pi 3 model b(Man, 2016)

Raspberry Pi 3 model B is the third generation of the Raspberry Pi, officially released in 2016, it has the functions of previous generations and it is fully compatible with 1st generation B+ and 2nd generation B. It still has the same credit card size, but in contrast, the performance of the processor has become more powerful as it has been upgraded to the 64-bit 1.2GHz quad-core ARM Cortex-A53, making its performance 10 times higher than the original Raspberry Pi. In addition, it integrates features such as 802.11n Wi-Fi, Bluetooth 4.1 and 4 USB ports. This change not only makes the Raspberry Pi 3 model B more convenient, but also makes it more adaptable to a variety of work environments such as the IoT. Raspberry Pi 3 model B can not only run the full range of ARM GNU/Linux distributions, but also supports Snappy Ubuntu Core and Windows 10 (Raspberrypi, 2016). The entire hardware equipment will be developed on the basis of Raspberry Pi 3 model B. The Raspberry Pi here is equivalent to the processing centre of the hardware equipment. It will have full control of the hardware such as when each sensor starts collecting data and when it stops receiving data. The Raspberry Pi here also helps the hardware equipment to transmit data to the server.

Arduino Nano

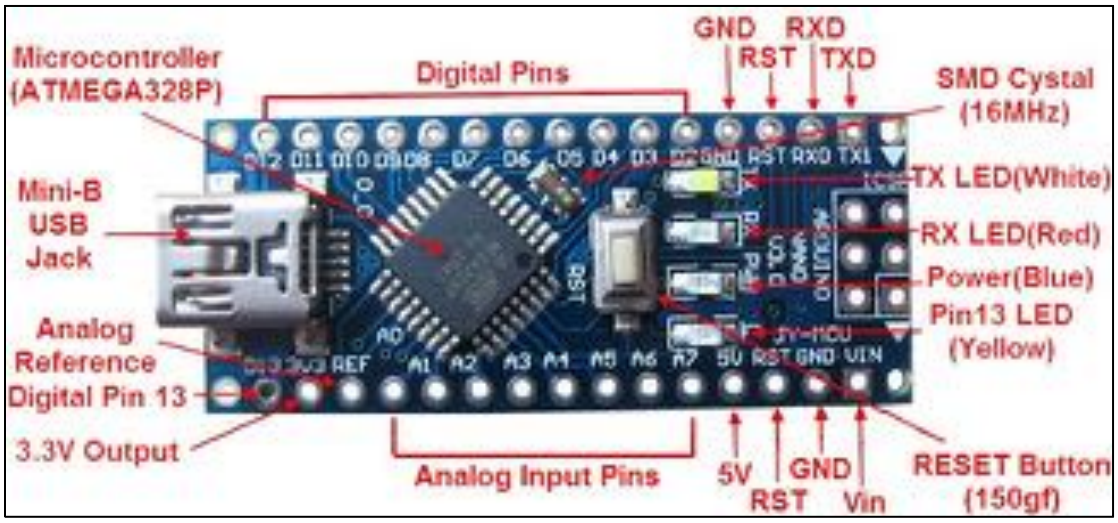


Figure 3-12: Specifications of Arduino Nano(Wiki, 2018)

Arduino Nano is a sub-product of the Arduino family and it is a miniature version of the Arduino Uno. The Arduino Nano removes the DC power interface from the board and uses the Mini-B standard USB interface to connect to the computer. In addition to the appearance change, other interfaces and functions remain same, the controller still uses ATmega328. The Arduino Nano comes with a variety of communication interfaces such as TWI (I2C compatible) interfaces, SPI interfaces and serial communication interface. Arduino Nano can communicate with the external serial port via the built-in UART port 0 (RX) and 1 (TX) (Liu, 2016). In the design, the Arduino will connect various sensors, convert the signals from sensor into data, and then serve as an interface to communicate with the Raspberry Pi.

Built-in connection methods: Ethernet, Bluetooth, Wi-Fi

Raspberry Pi has built-in Ethernet port, Bluetooth and Wi-Fi modules. All Ethernet, Wi-Fi and Bluetooth will be used as the main connections for data transmission. Bluetooth will be mainly responsible for the control of the device and admin/debugging. Since we already have the Ethernet and Wi-Fi that can send data and the range of the Bluetooth is just about 10 meters, so for this stage only the admin interface would be based on Bluetooth communication. However, the system should take consideration of some special circumstances. When all other communication modules are invalid, the data is able to transmit to the Admin’s mobile phone software via Bluetooth, and then the data is uploaded to the server through the mobile phone. This part will be introduced into the system in future improvements, and also the Admin mobile phone software will be added to the system in the function.

External connection methods: GSM/GPRS, LoRa and Zigbee

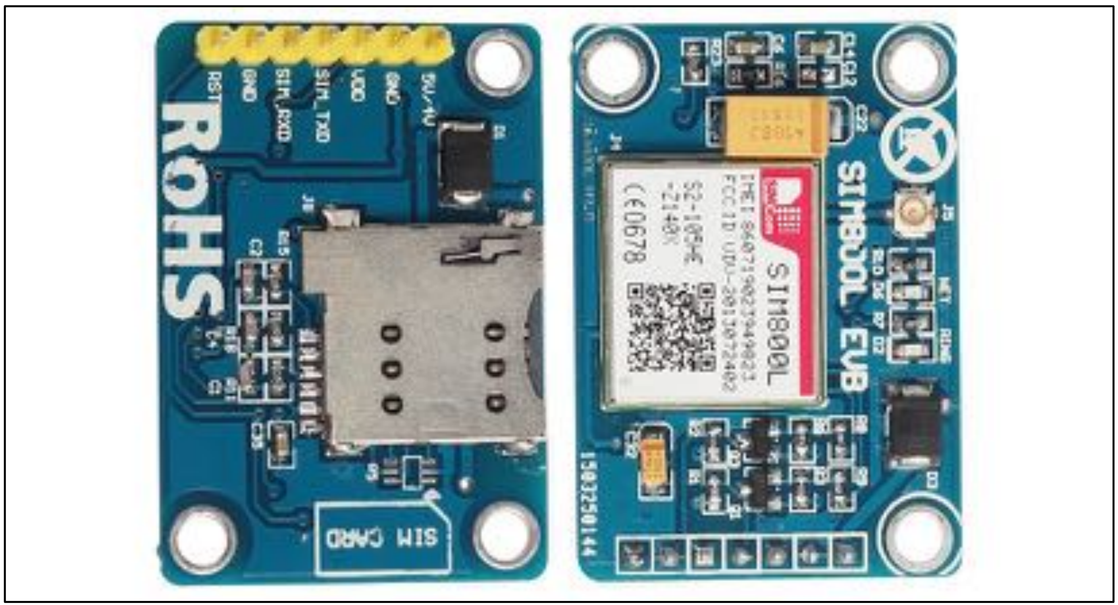


Figure 3-13: Sim800l GSM module(Electronics, 2018)

SIM800L is a quad-band GSM/GPRS module produced by Shanghai SIMCom Wireless Solutions Co., Ltd. It is compact and cost-effective. The module can operate at 850/900/1800/1900MHz and can transmit voice, SMS and data information with low power consumption. The GPRS of the module supports up to 85.6 kbps uplink and downlink data transmission. The module also supports AT cellular command interface for development(SIMCom, 2016).

In this research, the GSM/GPRS module SIM800L allows the hardware equipment to send data through 2G network and message to the server. But the SMS will not be used for data transmission in current stage. Usually the SMS sent by the hardware, it is first sent to the server at the carrier's SMS service centre, and then the service centre queues the received SMS and sends the SMS to the receiving terminal. If the receiving terminal is full or the server area is unable to communicate normally, the SMS will be delay and even damaged. Since this system wills be used for air monitoring 24/7, it will collect a large amount of data, and the carrier's price for SMS service is usually too high compare to the internet service. Therefore, using SMS to transmit data is not suitable at the moment. However, for future use, the function of the message data transmission will be developed in order to make our device more widely used in various contexts.

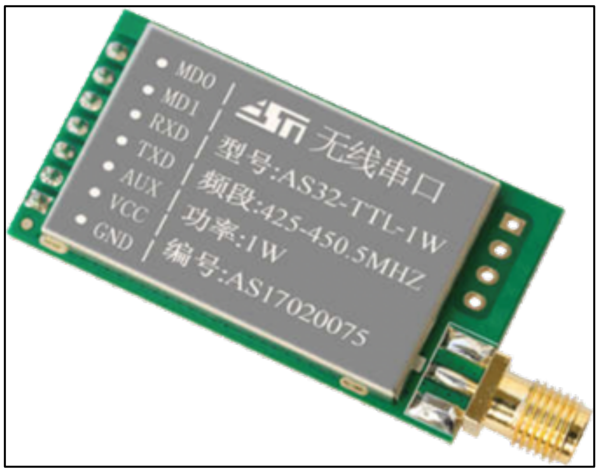


Figure 3-14: AS32-TTL-1W LoRa module(Ashining, 2017a)

The AS32-TTL-1W is a high-stability, industrial-grade wireless LoRa based data transmission module from Ashining Technology Co., Ltd, Chengdu, China. The module uses the SX1278 chip and LoRa spread spectrum transmission technology, enabling the module to achieve an effective communication distance of up to 8 km. It is widely used in PLC wireless communication, data acquisition equipment, monitoring equipment, medical equipment and electronic instrumentation automation control, video surveillance fields and so on. The module is compatible with 3.3v and 5v IO voltages. The module operating rate is from 410MHz to 441MHz with a total of 32 channels, each channel being 1M apart. The module parameters can be set according to the requirements, such as the serial port baud rate, transmission and reception frequency, transmission power, and radio rate. The module supports up to 19.2Kbps air baud rate with a maximum transmit power of 30dBm. The module supports UART protocol and adopts TTL output. Users can complete the development of wireless communication without understanding the complex wireless communication knowledge(Ashining, 2017b).

In this research, the LoRa module AS32-TTL-1W allows the hardware to send data to the server at a relatively long distance. The system also needs another LoRa module to receive data at server side. In this system, LoRa's role is mainly to enable the system to collect data smoothly even in the areas that doesn’t have Internet access or poor carrier service signal. Through the multiple device networking, it is possible to cover a large area with the air monitoring equipment. And because of the multiple device networking, the data can be transmitted through into one of the monitoring devices that close to the server by using LoRa and then the device could also use LoRa or Internet send unified the data back to server.

Figure 3-15: DRF1609H Zigbee module (DTK, 2017)

DRF1609H is a Zigbee module manufactured by DTK Electronics, Shenzhen. The module uses TI's CC2360 chip. With the powerful computing power of this chip, the Zigbee module can set up a standard MESH network for data transmission and retain the characteristics of the MESH network, when a node in the network fails, the network will automatically find other paths to complete the data transmission. This Zigbee module supports multi-level routing, and the data forwarding in the network is up to 200 levels. It supports data transparent transmission and node-to-node transmission. The wireless frequency of the module is 2.4GHz (2460MHz), and users can customize the channel from 2405MHz to 2480MHz. The data transmission distance can be up to 1600 meters in an open field environment. The module interface supports UART data transfer. It greatly facilitates the debugging of the device(DTK, 2017).

The Zigbee module used in this system is mainly designed for short-distance data collection scenarios, such as air detection at multiple points in the mall, air detection in the school's overall campus. Similar to the usage scenario of the LoRa module, it can be used without an internet network connection. Hardware equipment in the Zigbee network can send data to a unified node and upload all the data to the server. It is good choice for scenarios where the cost is limited, and the internet network is not needed, the following subsequent operating costs are also relatively low.

The use of Zigbee and LoRa will also help the system to achieve grid-based network monitoring without the need of Internet. Of course, there are more technologies similar to Zigbee and LoRa, such as XBee that can also be used in the same scenarios. However, due to the time limitation of the research, only GSM/GPRS, LoRa and Zigbee transmission methods were applied in the system. In future research, more suitable transmission methods will be discussed and introduced to the system to make the system more intelligent.

Data Structure and Operations

Since multiple sensors will be connected to the hardware equipment, the data obtained from the different sensors at the same time will be connected into a long String. Different data will be separated by number sign (#) for easy processing and storage on the receiving side later. The data will be temporary saved in the hardware equipment and then transmitted to the middleware layer when there is a suitable connection method available. The raw data will not be altered when being transmitted through the selected network. It will be exactly same as collected from the hardware equipment and passed to middleware layer, then written into database. The data will generally contain the following information: area coordinate, area name, time, temperature and air information. The operations relating to data at the hardware equipment will simply be: start/stop data collection and start/stop data transmission side while at middleware it would be start/stop pushing data. More specific details will be introduced in the low level design below.

Interaction between Subsystems

When the hardware equipment is plugged into the power, the hardware system automatically starts working. The system of the hardware part will remain active and continuously check for new sensors to access the hardware. Whenever a new sensor is connected to the hardware equipment, the hardware system will first collect and verify the sensor information, then begins collecting the sensor's data. The length of the data depends on the number of sensors simultaneously connected. When the hardware part collects the data and the hardware has one or more transmission methods to transfer the data back to the server, the hardware equipment will automatically determine and select a data transmission connection through the built-in network selection algorithm. Different connection methods follow a priority order, and two different transmission methods will not transmit the same set of data at the same time. For example, when the hardware equipment has both Ethernet and Wi-Fi connections, data is transmitted preferentially over the Ethernet connection.

On the other hand, administer can use the admin interface to observe the running status and data transmission status of the hardware equipment through the Bluetooth connection. The administrator can also take over the hardware equipment at any time, whether the hardware equipment does or does not have sensors plugged; or the hardware equipment is transmitting data. The hardware equipment will have some feedback on the commands entered by administer. For example, administrator can control when the hardware equipment needs to pause or start data collection and transmission. The admin interface can also have the functions that controls system to manually select the transmission connection over the network selection algorithm during the data collection and transmission, and this process does not affect the data transmission. Then the server will store the data received through different connections to the database and wait for the request from application layer.

3.2.5 Low level Design

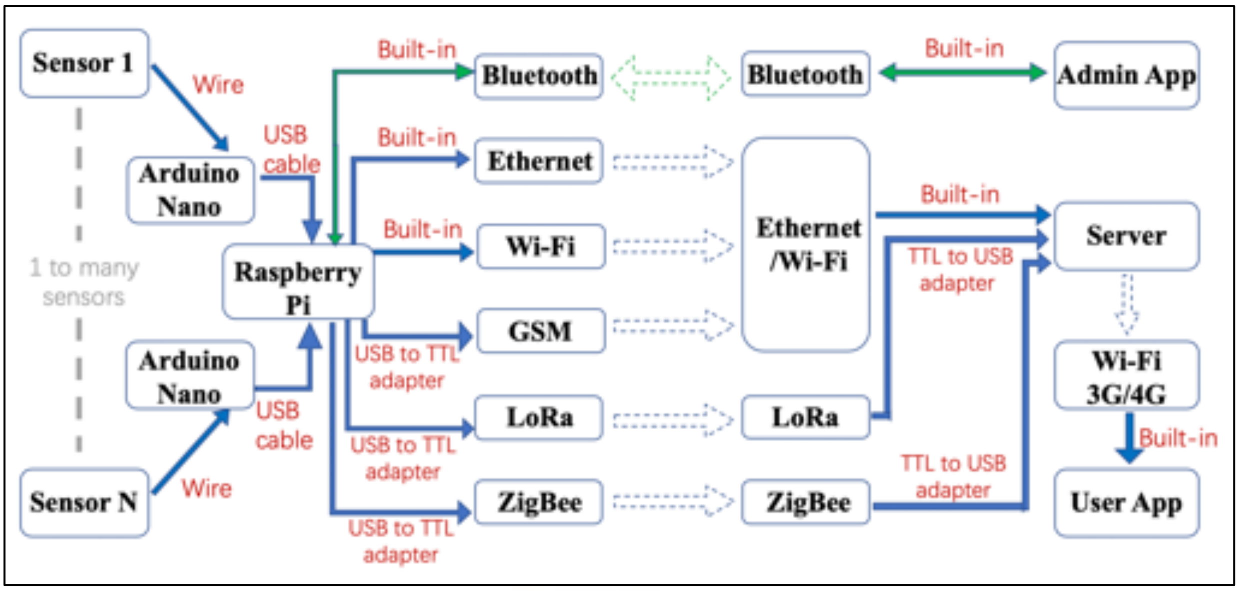


Figure 3-16: Low level design

In Figure 3-16 above, the details of the structure of how each part are physically connected together in the system is depicted. Each sensor is connected to an Arduino Nano board through wire. The Arduinos connect to the Raspberry Pi USB ports though USB type A to USB micro-B cables. The Raspberry Pi 3 model B has three built-in communication modules which are Bluetooth, Ethernet and Wi-Fi, these modules do not require extra hardware to add to the system, they could be configured in the software. For the external communication modules such as GSM, LoRa and Zigbee are needed to use USB to TTL serial adapters to connect Raspberry Pi on the USB ports. Similarly, on server side, expect the Ethernet and Wi-Fi connection methods provided by the server device, other connection methods first need to connect to the USB to TTL serial adapter such as CP2102 to be able to connect the USB port of the server.

The Bluetooth module of the system will be wirelessly connected to the Bluetooth module of the admin phone for data transmission and admin system monitoring. Users will also receive data from the server wirelessly through Wi-Fi and 3G/ 4G.

Modularized design by using USB as the main connection method

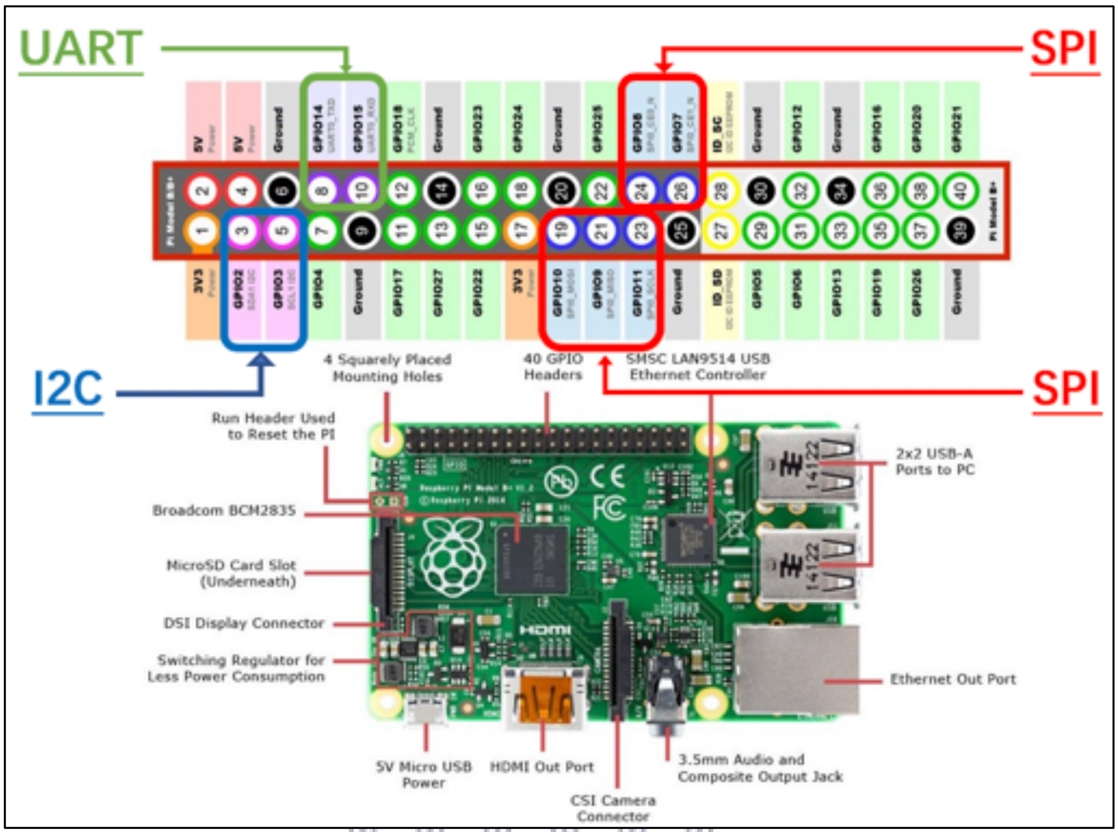

UART
SPI
I2C
SPI
4 Squarely Placed Mounting Holes
40 GPIO Headers
SMSC LAN9514 USB Ethernet Controller
Run Header Used to Reset the PI
Broadcom BCM2835
MicroSD Card Slot (Underneath)
DSI Display Connector
Switching Regulator for Less Power Consumption
2x2 USB-A Ports to PC
Ethernet Out Port
5V Micro USB Power
HDMI Out Port
CSI Camera Connector
3.5mm Audio and Composite Output Jack

Figure 3-17: Raspberry Pi 3 model B GPIO Pinout diagram(Jameco, 2016)

Usually there are two ways to connect the sensors to the Raspberry Pi, through the GPIO and the USB port. Except the power supply pins 3V3, 5V and Ground, all GPIO pins of the Raspberry Pi can be used for digital input or output. However, when the input signal of the sensor is an analog signal, Raspberry Pi is only able to read it after the sensor connected to an Analog-to-Digital Converter (ADC) to change the analog signal to digital signal. The common way to connect sensors is through I2C and SPI communication. As Figure 3-17 shown above, pins GPIO2 and GPIO3 can be used for I2C connections. Pins GPIO10, GPIO9, GPIO11 and GPIO8, GPIO7 can be used to connect high speed SPI devices. Pins GPIO14 and GPIO15 can be used for UART based equipment.

The entire system is designed to achieve the goal of modularization. The requirements of modular are designed that the hardware device should run on its own. When a sensor is connected, the system should be able to automatically identify the connected sensors and also to read the data from the sensors. The sensor modules can be added, removed and replaced to the system at any time to meet the needs of different environmental inspections. The system is also required to support multiple sensors connection simultaneously.

The I2C communication protocol usually supports multiple devices to be connected at the same time. In theory, 12C can connect up to 128 devices. I2C communication is usually used for communication between chips on the same board, not much for long-distance communication. In I2C communication protocol, each connected sensor device has a unique address, and I2C uses the unique address for sensor device identification and data reading. Therefore, when the hardware is connected to different sensor devices, the software code also needs to bring in different sensor device addresses to correctly read data from the sensors.

This is better for fixed-sensor equipment development, but for systems with replaceable sensors, and in software side this means that each time when the sensors changed the system also needs to use different code in the software to be able to read the data from sensors correctly. At the same time, it is also necessary to consider that the I2C communication protocol will not work properly when the two different connected sensor devices have same addresses. The way to connect the sensors in this way cannot meet the requirements of modularity, and it also brings great inconvenience to the use.

In contrast, I2C is often used to connect sensors, while SPI is more used to connect memory cards or liquid crystal displays (LCD) and there are not many air quality sensors available on the market that support SPI communication. Air quality sensors that support the UART communication protocol are more abundant. Due to their simple circuitry, they are very suitable for new developers. The shortcomings of the UART communication are quite obvious, in that they only support one-to-one connection. Thus, the Raspberry Pi 3 model B can only connect to one sensor via UART at a time – this is insufficient for this research. That is why USB was finally choose as the main connection method between sensors and Raspberry Pi. Raspberry Pi 3 model B has 4 USB ports and can also be extended by connecting USB expansion hub to support more sensor access. (The I2C and SPI communication protocols were also used in the system. The I2C communication protocol was used to connect the RTC module, and the SPI was used to connected LCD screen. These two parts will be described in detail in the implementation section).

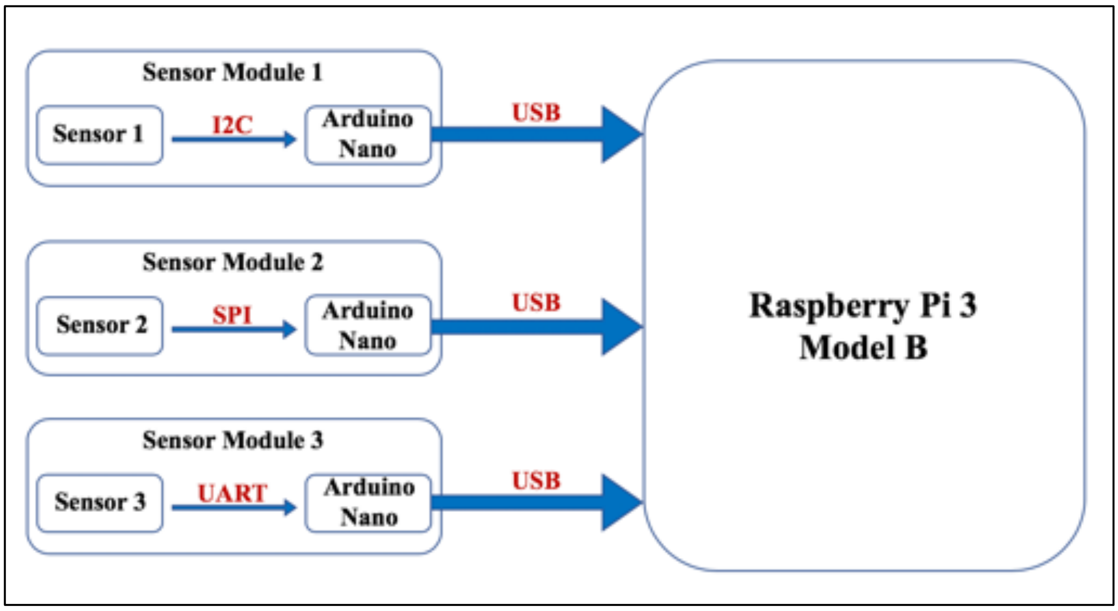


Figure 3-18: USB connection in the system

So, there is an intermediate digital progressing device introduced to the system which is the Arduino Nano to support a variety of sensors connection. Arduino supports I2C, SPI and UART communication protocols. Sensors based on I2C, SPI or UART communication protocols can be connected to the Arduino and then transferred to the Raspberry Pi via USB connection as shown in Figure 3-18.

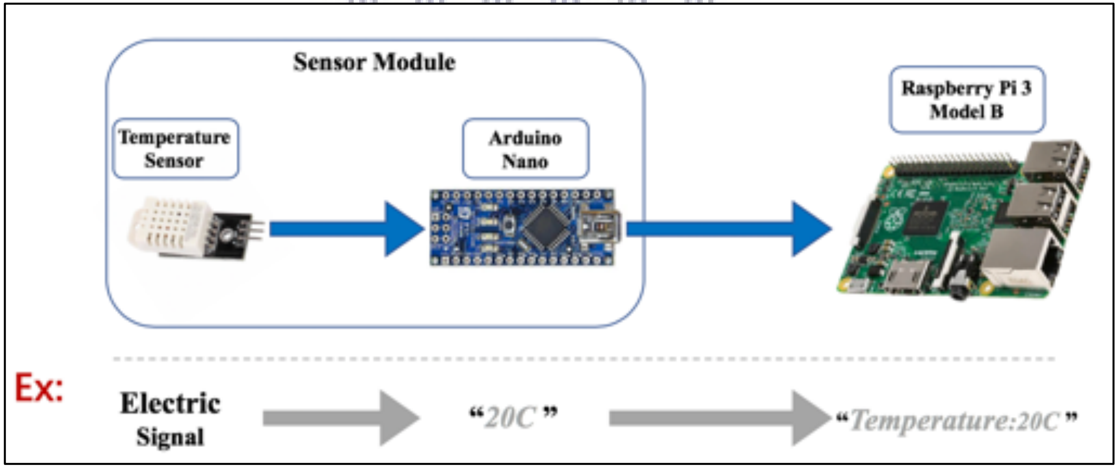


Figure 3-19: Use of Arduino in the system

Arduino's role here is not only to connect sensors and convert the sensor's electrical signals to data, but also to convert the data to a unified format. After Arduino receives the data, Arduino adds an id-like name to the data to help system identify the sensor.

As shown in the Figure 3-19 above, the sensor is connected to Arduino and then connected to the Raspberry Pi. When the sensor generates an electrical signal, it arrives at Arduino, the Arduino will convert the electrical signal into data by referring to the datasheet formula of the corresponding sensor, and then add the corresponding sensor or data name in front of the data.

For example, the electronic signal generated by the temperature sensor in Figure 3-19 is converted to "20C" after the data arrived at Arduino. Then Arduino adds the name of the data “Temperature:” before the data and send it to the Raspberry Pi. The data received by the Raspberry Pi is " Temperature: 20C". The Raspberry Pi is able identify the connected sensor and the type of data based on the name of the received data. The advantage of this design is that the system is modularized to the greatest extent and the hardware part only needs to replace the sensor modules to detect different air pollution data. The same set of equipment can be used for different precisions levels. One only needs to replace the sensor module to meet the needs of environmental or home being monitored.

Data Flowchart Diagram

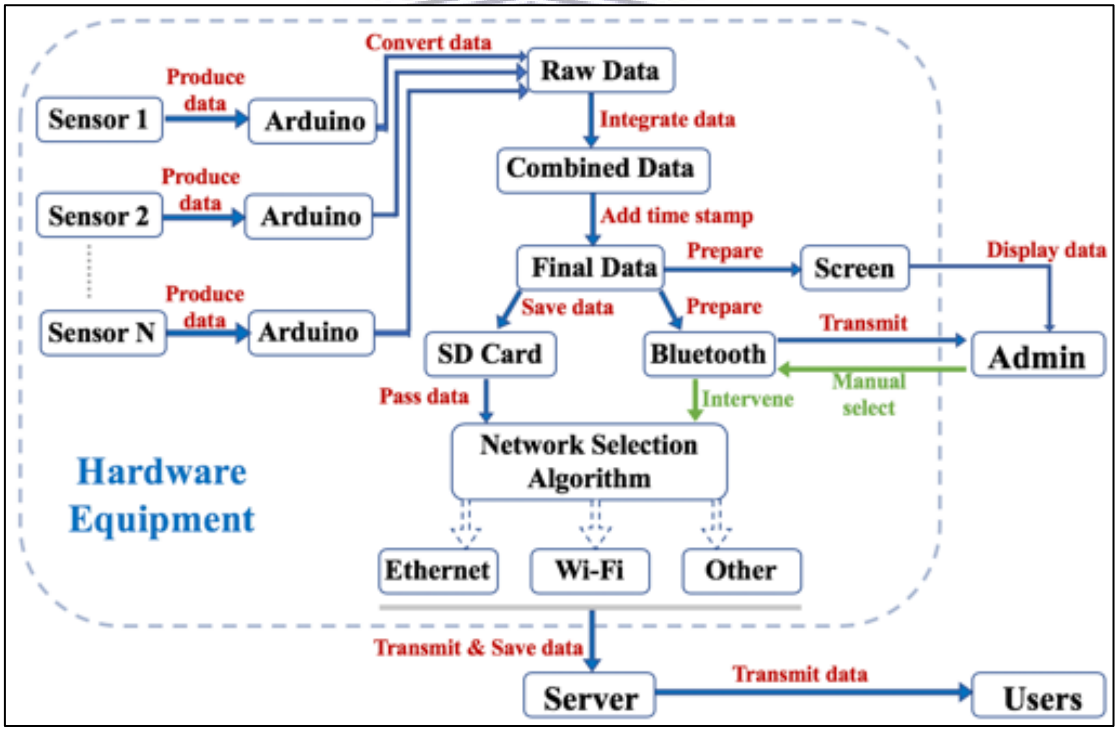


Figure 3-20: Data Flowchart Diagram

As shown in diagram in Figure 3-20, the data from different sensors will be edited by the Arduinos and then integrated together to a set of data. A timestamp will be added to the combined data to record the time of the collected data. Different sensors’ data are separated by number sign

"#" while the data from same sensor are separated by comma ",". Here is an example of an edited data:

```
#Time,2018/08/09-12:05:05#SensorName1,Data1,Data2#SensorName2,Data3
```

The Final data will be temporary saved in the database of the Raspberry Pi which is on its SD memory card. This final data will also be displayed on the screen and transmitted to the admin through Bluetooth. The saved data in SD card will be passed to the network selection algorithm and transmitted to the server through a suitable connection method such Wi-Fi connection. The admin's manual intervention through Bluetooth is actually a supplement to the network selection algorithm, it helps the system to select the best connection to transmit data to the users.

Inner Class Datatypes and Methods

The focus of this research was mainly on the development of the hardware part, thus in designing the database a csv files was used. In future work and upon completion of the main air quality monitoring system, the database could be changed to SQL/No-SQL standard. An example of the datatype that the system may have is shown below. Each hardware equipment has its own device id as the primary key. And the content of the data depends on the sensors connected to the hardware equipment.

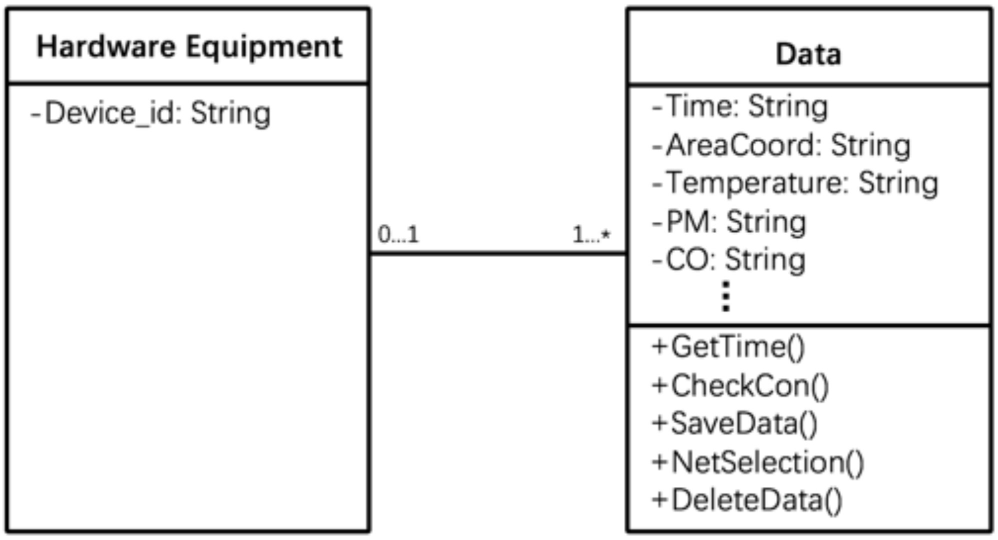


Figure 3-21: Data types and methods

Detailed Object Diagrams

The diagram in Figure 3-22 below shows an example of the data that can be collected by one hardware equipment. The hardware equipment can be placed at difference areas and collect many sets of data.

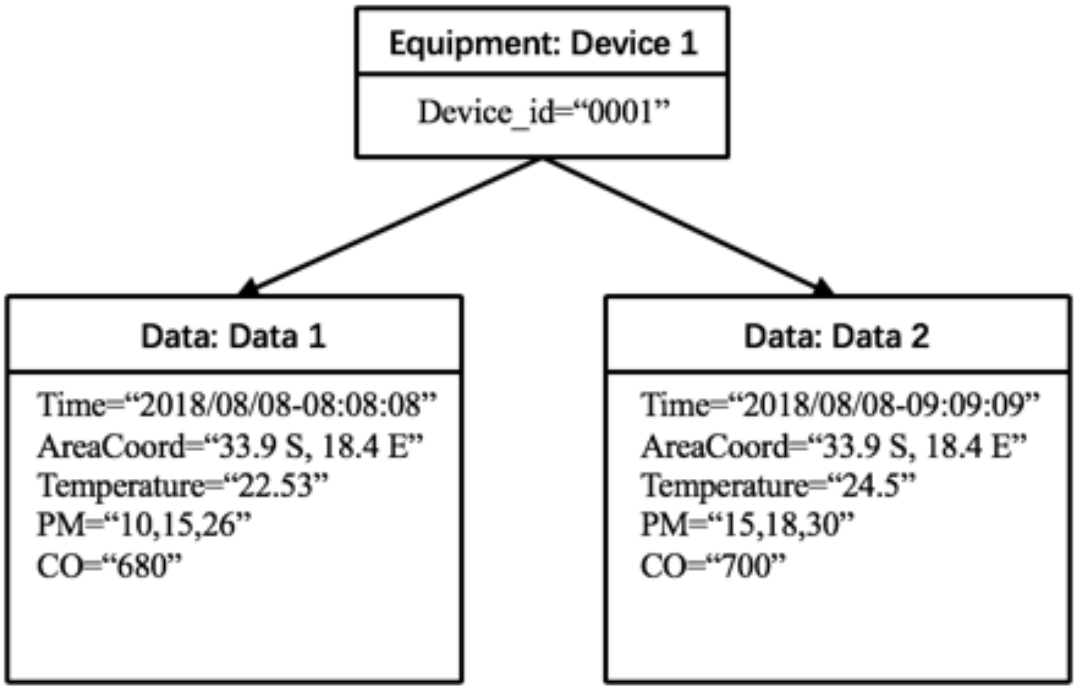


Figure 3-22: Object Diagram

State Diagram

The main algorithm of the hardware part is divided into three parts which are time module algorithm, sensing and saving algorithm and network selection algorithm. Since the Raspberry Pi does not have an RTC module, the time of the Raspberry Pi system will not be synchronized when the power is turned off. The main function of the RTC time module algorithm is to synchronize time across the hardware equipment.

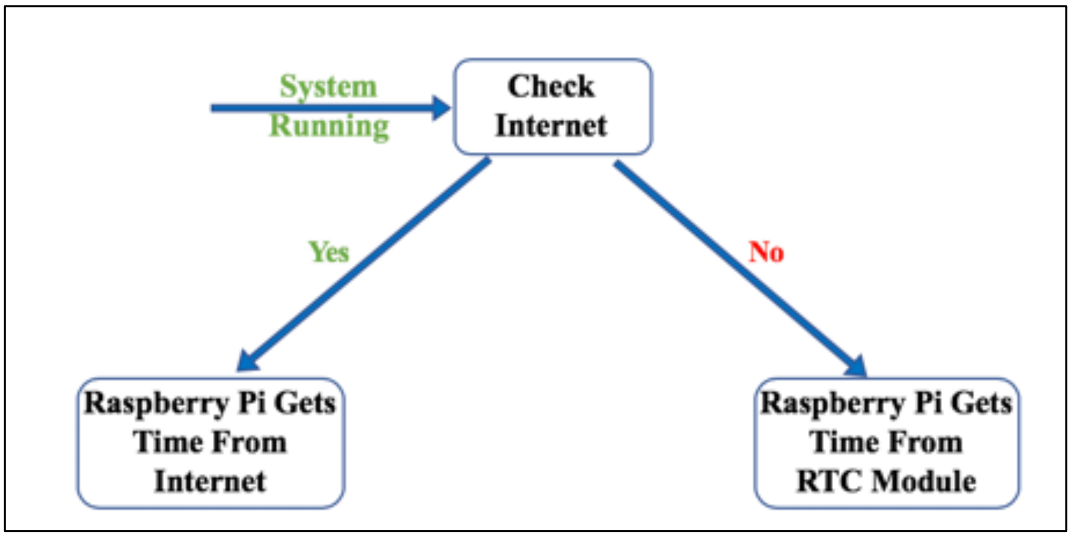

System Running
Check Internet
Yes
No
Raspberry Pi Gets Time From Internet
Raspberry Pi Gets Time From RTC Module

Figure 3-23: Time function diagram

Due to the USB connection method used between the hardware and sensors, the Raspberry Pi can use the serial port to read data from the sensors, but the premise is that Raspberry Pi has to know the port and the rate of the connected device. The sensing and saving algorithm diagram mainly show how the system recognizes the sensor's port and rate information when a new sensor is connected to the hardware. This ensures that the hardware and sensor can communicate and collect data.

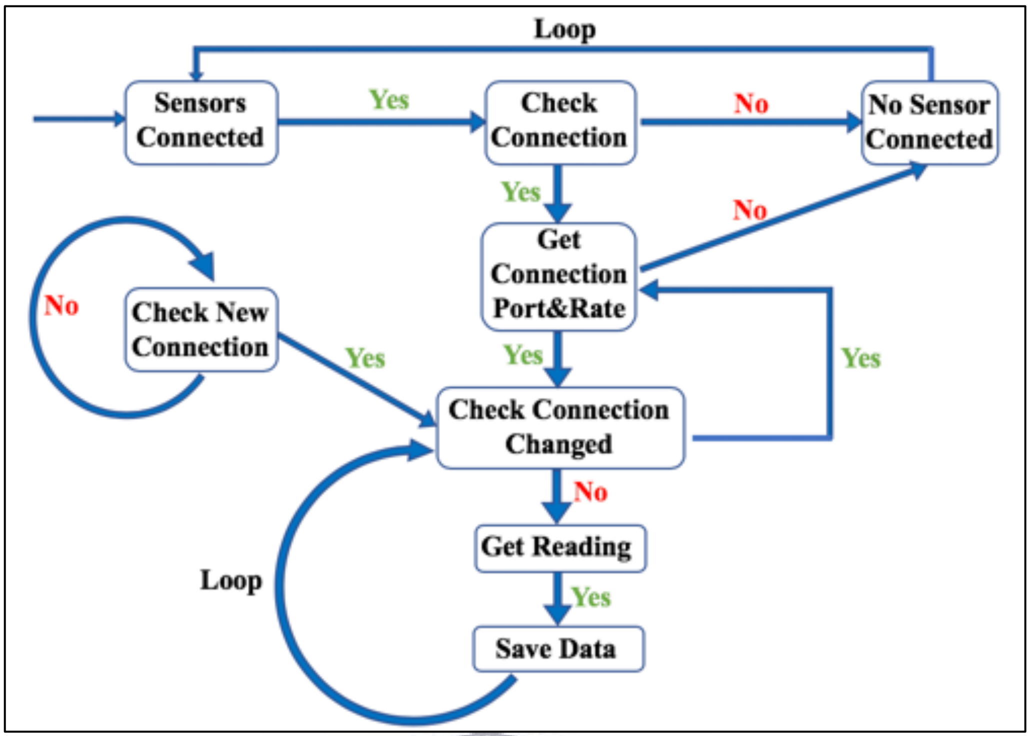


Figure 3-24: Sensing and saving algorithm diagram

The system determines the current best data communication method to transmit the data using the network selection algorithm. However, if the administrator manually selects a communication channel, the network selection algorithm would always use that over any other.

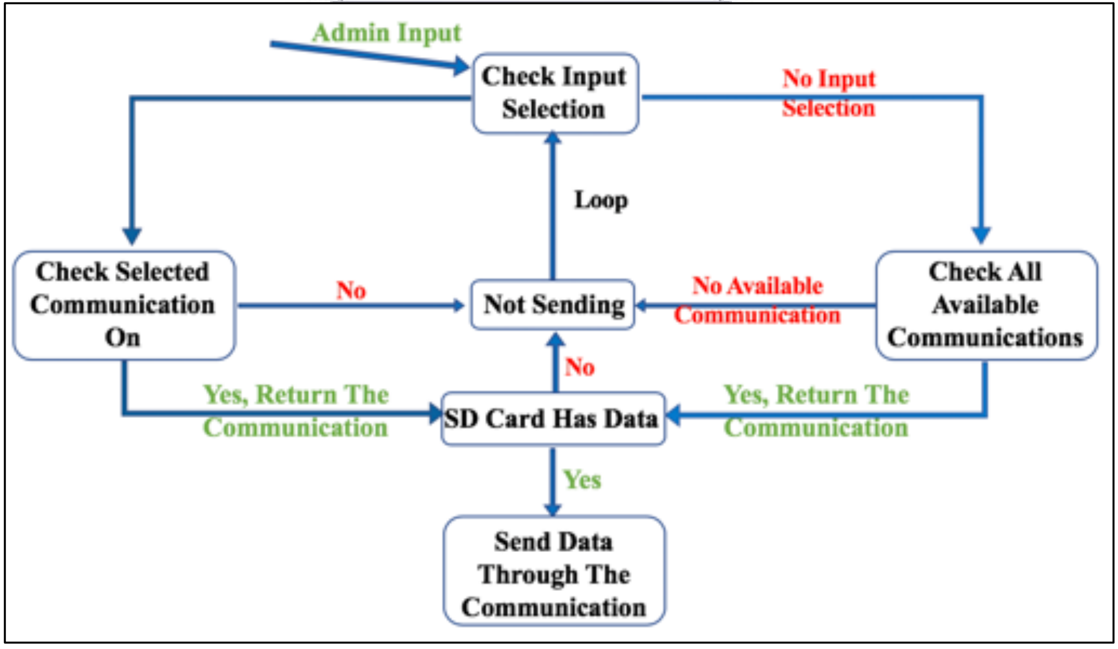

Admin Input
Check Input Selection
No Input Selection
Loop
Check Selected Communication On
No
Not Sending
No Available Communication
Check All Available Communications
No
Yes, Return The Communication
SD Card Has Data
Yes, Return The Communication
Yes
Send Data Through The Communication

Figure 3-25: Network selection algorithm diagram

Pseudo-code

Time module function

```
While hardware system starts running
    Check the hardware Internet connection
        If hardware has internet access
            Raspberry Pi synchronizes time from Internet
            Update RTC module with the synchronized time
        Else hardware has no Internet access
                Raspberry Pi synchronizes time from RTC module
    System runs this every month
```

Sensing and saving algorithm

```
While hardware system starts running
        Check if there are any sensors connected
        If there are sensors connected
                If there is new sensor
                        Get the all connected sensors' port and rate
                Get readings from the sensors
                Save data
    Else hardware does not sense or save data
```

Network selection algorithm

```
While hardware system starts running
        If admin specifies a communication module, select that
        Else select best communication module
            If the selected communication module is connected and can be used for data transmission
                If SD card has data, send data through the communication to server
            Else check current available communication
                If SD card has data, send data through the current most suitable communication to
                server
```

Algorithmic Description

The all three algorithms will be written in the Raspberry Pi board and loop on themselves. When the hardware equipment turned on, the time module will first check the hardware Internet connection and help the hardware sync to the right time. The hardware then uses the sensing algorithm to identify the port and rate of all connected sensors on the Raspberry Pi. After completing the check of sensors' port and rate, the Raspberry Pi will start to establish a serial connection, read the sensor data through serial, and save the data in the SD card. When the connected sensor changes or there is a new sensor added to the hardware, the Raspberry Pi will re-identify the port and rate of all connected sensors on the hardware and re-establish the connection with the new port and rate to read the data.

After data is stored in the SD card, the network selection will firstly check the communication method selected by the admin. If the selected communication method is connected and available, the algorithm will start transmitting data to the server. It follows the 'First In First Out' (FIFO) data structure; the first data added to the SD card will be the first one to be transmitted. But if the communication method is not connected, then the data will stay in the SD card until communication method is connected.

If the admin has not manually selected any transmission communication for the hardware, then the network selection algorithm will automatically choose the transmission method according to the current available communications. The network selection algorithm will preferentially select the transmission communications according to a certain order, the priority order follows Ethernet > Wi-Fi > Zigbee > LoRa > GSM, but the algorithm will also combine the signal strength of the actual communication modules to comprehensively choose the best transmission communication to transmit data.

3.3 Summary

This chapter described some of the application scenarios and specific functions that the system will need, as well as the choice of hardware platform and the design of the software part. The system has been separated into four layers, which are sensing layer, networking layer, middleware layer and application layer. The main task of the sensing layer is to collect data; the networking layer is mainly responsible for the transmission of data to the server; the middleware layer focuses on the analysis of the data at server side while the application is responsible for presenting the data at user side. The various layers of the system cooperation and work together to complete the entire air monitoring system. There are some other designs aspects, such as the admin mobile control app which requires the development of some mobile phone software to help display the data transmitted by the system via Bluetooth. On the server side, the system on completing the process of receiving data saves the data in a CSV files. These sections will all be described in the next chapter.

4. Implementation

4.1 Implementation Environment Requirements

This chapter will focus on how the hardware is connected and explain some of the code. The first one is the code for the hardware device like Raspberry Pi. The code for this part is mainly responsible for the functions of receiving data and sending data to the server (middleware layer). The other aspect of the code is for the server (middleware layer) and focuses on receive data and storage into the database.

4.1.1 Hardware and wiring instructions

Raspberry Pi 3 Model B:

The main platform used to create interactive electronic objects. It is a control centre of data collection and transmission. The Raspberry Pi is also used as a temporary data storage for collected data. The sensor modules will be connected to the Raspberry Pi through USB ports.

Arduino Nano:

Arduino is mainly to help the sensor convert electronic signals into data and pass it to the Raspberry Pi. The advantage of Arduino is that it is easy to connect various sensors. Different sensors will be connected to the Arduinos in their specified ways. Using a USB type A to USB micro-B cable, Arduinos will be connected to the Raspberry Pi**.**

USB to TTL module:

The pins on the communication modules are different from that of the Raspberry Pi and therefore cannot be directly connected together. A TTL-to-USB adapter module is needed to convert the TTL connection of the communication module to be the USB ports of the Raspberry Pi.

SIM800L GSM/GPRS module:

SIM800L module mainly helps the system to send data to the server via GPRS. The module is connected to the USB to TTL convert first then connect to the Raspberry Pi. Module wiring instructions is shown below:

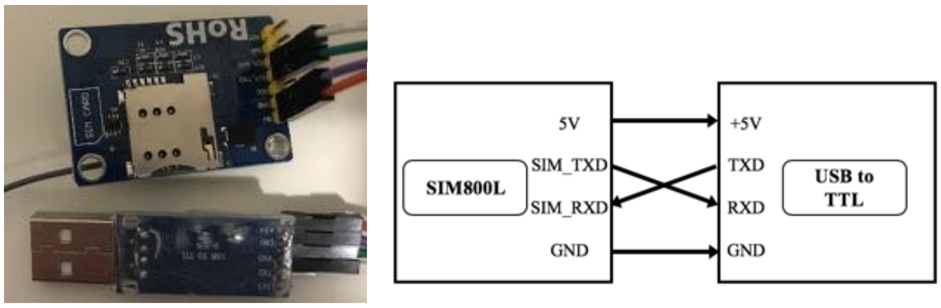


Figure 4-1: Sim800 GSM/GPRS module wiring instruction diagram

DRF1609H Zigbee module:

DRF1609H module mainly helps the system to send data to the server via Zigbee. The module is connected by the TTL to USB cradle that comes with the device then to the USB port of Raspberry Pi.

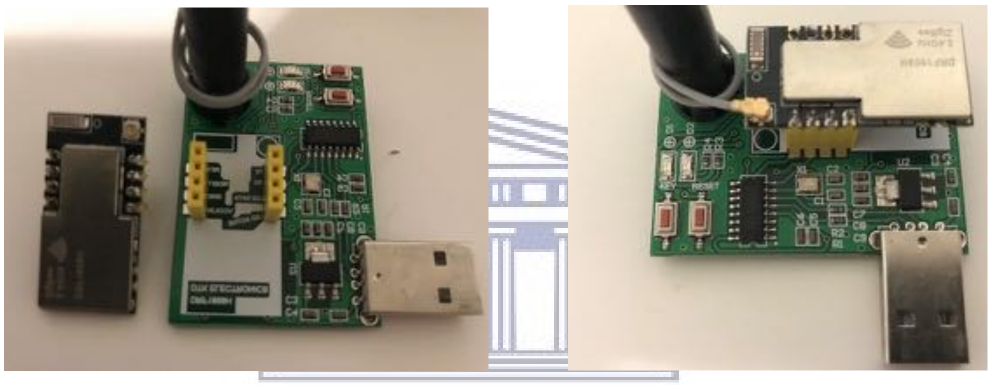


Figure 4-2: Zigbee module wiring instruction diagram

AS32-TTL-1W LoRa module:

AS32-TTL-1W module mainly helps the system to send data to the server in a long distance via LoRa technology. The module is also connected by a TTL to USB cradle that comes with the device then to the USB port of Raspberry Pi.

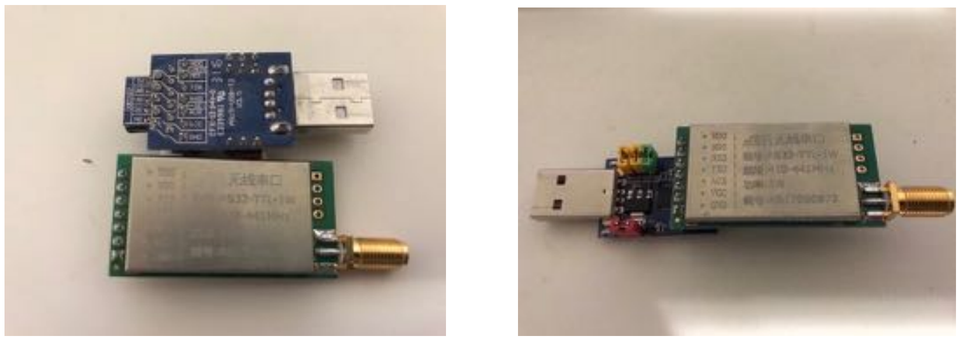


Figure 4-3 : LoRa module wiring instruction diagram

Pioneer600 Screen & RTC module:

Pioneer600 is a versatile extension hat for Raspberry Pi made by WaveShare(Waveshare, 2015). The features of this hat include 0.96" OLED, DS3231 RTC module, five-way joystick, BMP180 temperature/pressure sensor, etc. The OLED will be used as the main display screen of the device and will be connected to the Raspberry Pi using SPI protocol. The DS3231 will serve as the RTC module to provide accurate time when the system is powered off, and this module will connect to the Raspberry Pi through I2C protocol. The device would not use the built-in BMP180 module on the hat as a temperature sensor because the extension hat will be connected directly to the Raspberry Pi GPIO, and the sensor will be close to the Raspberry Pi motherboard during the running process. The detected temperature data may be affected by the heat dissipation from Raspberry Pi. The actual connection method is as shown below.

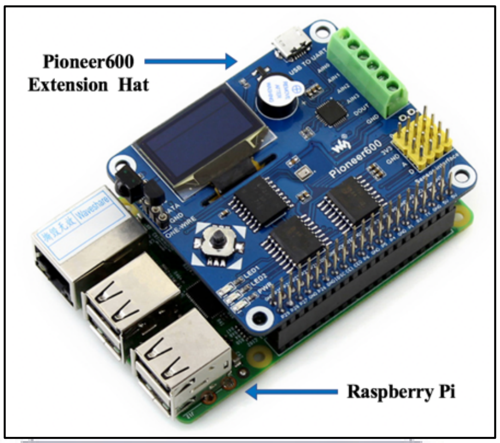


Figure 4-4: Pionner600 Raspberry Pi hat(Waveshare, 2015)

DHT22 Temperature sensor:

DHT22 sensor is a temperature and humidity composite transmission with calibrated digital signal output sensor. DHT22 sensor is widely used in various fields due to its responsiveness, anti-interference ability and high cost performance. The basic parameters of the DHT22 sensor are shown on Table 4-1 below.

Table 4-1: DHTT22 temperature sensor technical specification(T. Liu, 2016)

Sensor Name	Parameters	Range	Precision	Resolution	Com Protocol
DHT22	Temperature	-20~80°C	±0.5 °C	0.1 °C	Single bus digital signal output
	Humidity	0~100%RH	±2%RH	0.1%RH	

Module wiring instruction is shown as following Figure 4-5.

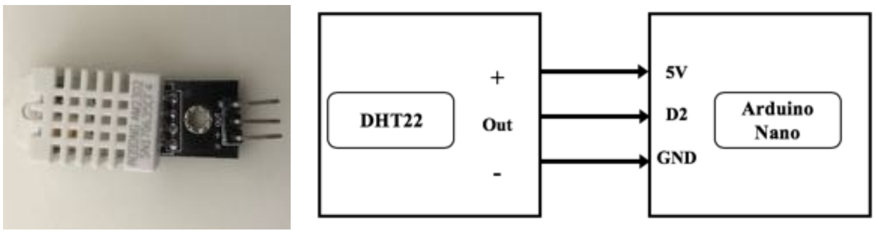


Figure 4-5: DHT22 sensor wiring instruction diagram

PMSA003 PM sensor:

PMSA003 is a particle matter sensor made by Beijing Plantower Co., Ltd. It is based on the principle of laser light scattering. The sensor has an ultra-small size and can be used for portable devices. The sensor can continuously collect and calculate the number of different suspended particles in a certain volume air and then calculate the mass concentration. The sensor is able to produce the $PM_{1.0}$, $PM_{2.5}$ and PM_{10} data at same time. The output data can be sent through the serial port. The basic parameters of the PMSA003 sensor are shown on Table 4-2 below.

Table 4-2: PMSA003 PM sensor technical specification(Yong and Haoxin, 2016)

Sensor Name	Parameters	Range	Precision	Resolution	Com Protocol
PMSA003	PM1.0	0~500ug/m3 (PM2.5 standard value)	±10ug/m3@ 0~100 ug/m3 ±10%@ 100~500 ug/m3	1 ug/m3	UART output
	PM2.5				
	PM10				

Module wiring instruction is shown as following Figure 4-6.

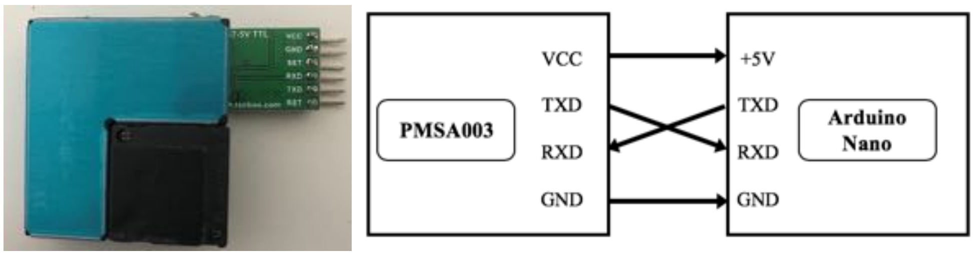


Figure 4-6: PMSA003 sensor wiring instruction diagram

ZE12-CO CO sensor:

ZE12-CO is a electrochemical sensor made by Winsen Electronics Technology Co, China(Winsen, 2016). It can use its chemical properties to detect CO in the air. The same series of the gas sensors are also available to sense other gases such as SO2, NO2, and O3. The ZE12 series of sensors offer the advantages of high sensitivity and high resolution. The sensor has a built-in temperature sensor for temperature compensation and UART data output for ease of use. The basic parameters of the ZE12-CO sensor are shown on Table 4-3 below.

Table 4-3: ZE12-CO sensor technical specification(Winsen, 2016)

Sensor Name	Parameters	Scale	Precision	Resolution	Com Protocol
ZE12-CO	CO	0~12.5ppm	±1.5%FS	≤10ppb	UART output

Module wiring instruction is shown as following Figure 4-7.

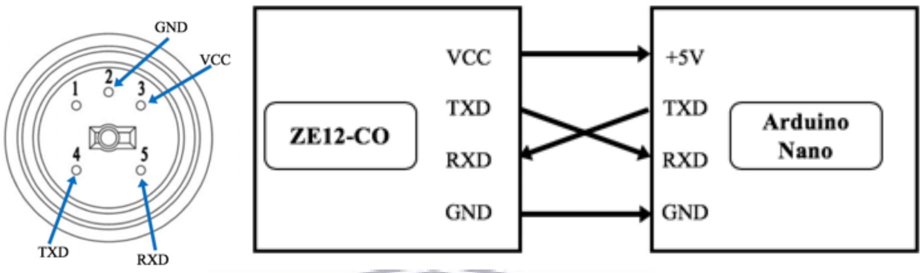


Figure 4-7: ZE12-CO sensor wiring instruction diagram

NEO-6M GPS sensor:

The NEO-6M GPS module has features of high sensitivity, low power consumption, and miniaturization. The module is especially suitable for applications in vehicles, handheld devices, and mobile positioning systems. The basic parameters of the NEO-6M sensor are shown in the Table 4-4.

Table 4-4: NEO-6M GPS module technical specification (U-blox, 2015)

Sensor Name	Parameters	Range	Precision	Resolution	Com Protocol
NEO-6M	Longitude	-180~180 oC	±2.5M	0.000001 oC	UART output
	Latitude	-90~90 oC			

Module wiring instruction is shown in Figure 4-8.

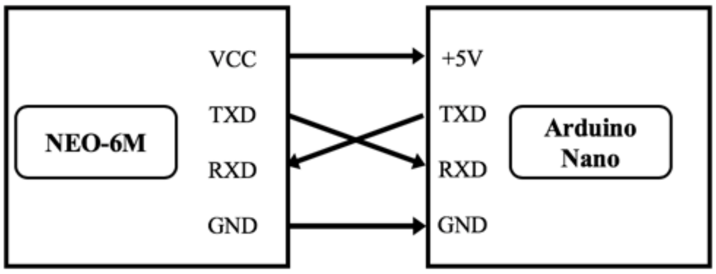


Figure 4-8: NEO-6M module wiring instruction diagram

4.1.2 Software

1. Arduino: The ADK for writing Arduino program and uploading to the Arduino board.
2. Python: The coding language used in Raspberry Pi (hardware equipment) and the server (middleware layer) for receiving and preparing data.
3. MIT App Inventor: Android app develop kit
4. Android System: Used for running our admin control app.
5. Oray NAT service: Used to build and manage server.
6. PubNub

4.2 Software system installation

4.2.1 Installation of Raspbian for Raspberry Pi

The Raspberry Pi is mainly used to collect data and temporarily store same. The official Raspbian system released by Raspberry Pi requires about 4Gb of storage space. An SD card with at least 8 Gb of space is required to store both the Raspbian and captured data. The official Raspbian system can be downloaded at the following URL: Https://www.raspberrypi.org/downloads/raspbian/. After installed the system and powering up, the Raspberry Pi will read the operating system that was previously burned in the SD card, load and run the operating system.

Figure 4-9: Backside view of the Raspberry Pi with SD card

Before powering up, first connect the HDMI video cable to the Raspberry Pi, so that the Raspberry Pi can project the video signal to the display screen. Then connect the mouse and keyboard to the Raspberry Pi through the USB port, which is convenient for the user to input commands.

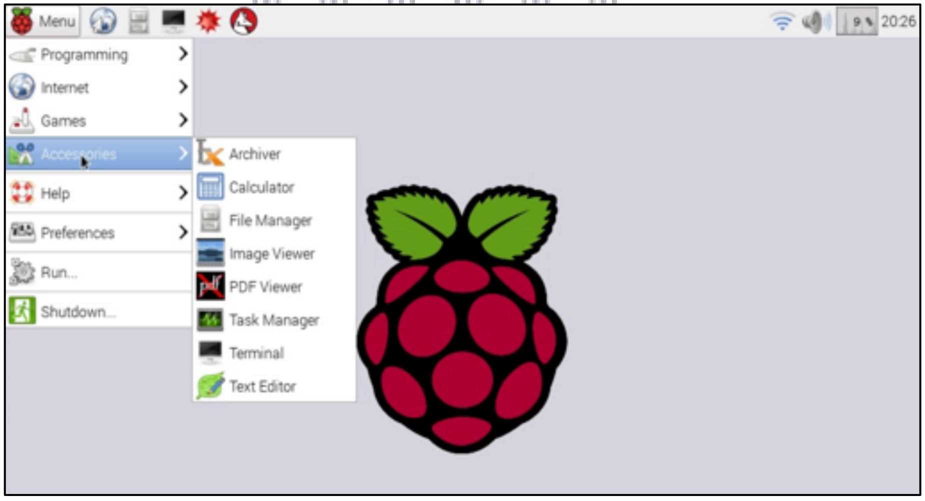
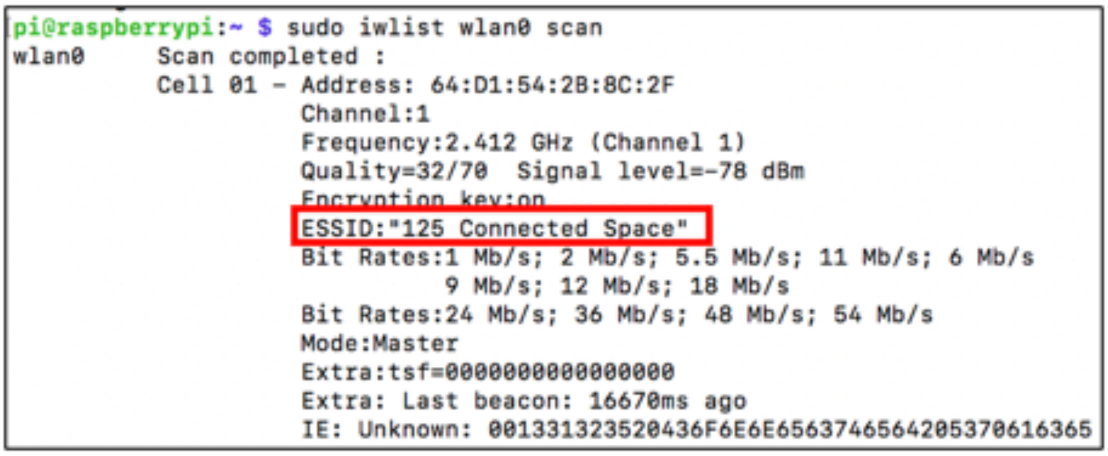


Figure 4-10: Desktop of Raspberry Pi

Once connected to the power source, the Raspberry Pi system is displayed on the screen. The Raspberry Pi is able to connect to the Internet through Ethernet cable or Wi-Fi connection. By using the Wi-Fi connection, users need to click on the Raspberry Pi icon in the upper left corner to enter the menu shown in Figure 4-10 to open the "Terminal", and then enter the following command code in the terminal to scan the available Wi-Fi network nearby.

```
sudo iwlist wlan0 scan
```

The scan result should be similar to that shown in Figure 4-11 below, where ESSID is the name of Wi-Fi network. It also provides the Wi-Fi signal strength and some basic information about the Wi-Fi network.



```
pi@raspberrypi:~ $ sudo iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 64:D1:54:2B:8C:2F
                    Channel:1
                    Frequency:2.412 GHz (Channel 1)
                    Quality=32/70  Signal level=-78 dBm
                    Encryption key:on
                    ESSID:"125 Connected Space"
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s
                    Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                    Mode:Master
                    Extra:tsf=0000000000000000
                    Extra: Last beacon: 16670ms ago
                    IE: Unknown: 00133132352043 6F6E6E656374656420537061 6365
```

Figure 4-11: Wi-Fi scanning results on command line

Entering the following command code would display the Wi-Fi network list that has been saved on the system.

sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

The Raspberry Pi system is able to save the new Wi-Fi connection by adding the following format code to the displayed list. If the Wi-Fi network is an open network without a password, then leave the *psk* line empty and change the *key_mgmt* to *NONE*, as shown below. After completing the addition, press “ctrl+x” to exit and save the changes. When the system reboots next time, the device will be able to connect to the specified Wi-Fi network. It is worth to note that the Wi-Fi network order in the Wi-Fi list will affect the system connection. The system will give more priority to connect the Wi-Fi network with the highest order when it starts up.

```
network={
    ssid="Wi-Fi_Name"
    psk="Wi-Fi_Password"
    key_mgmt=WPA-PSK
}
```

There are other basic settings including **SSH**, **SPI**, **I2C** interface that can be enabled in the Raspberry Pi software configuration tool by entering the command code sudo *raspi-config* in the terminal and then select the **Interfacing Options** as shown below. (Alternatively, it is also able to use the Raspberry Pi software configuration tool to select the **Network Options** to add a new Wi-Fi connection.)

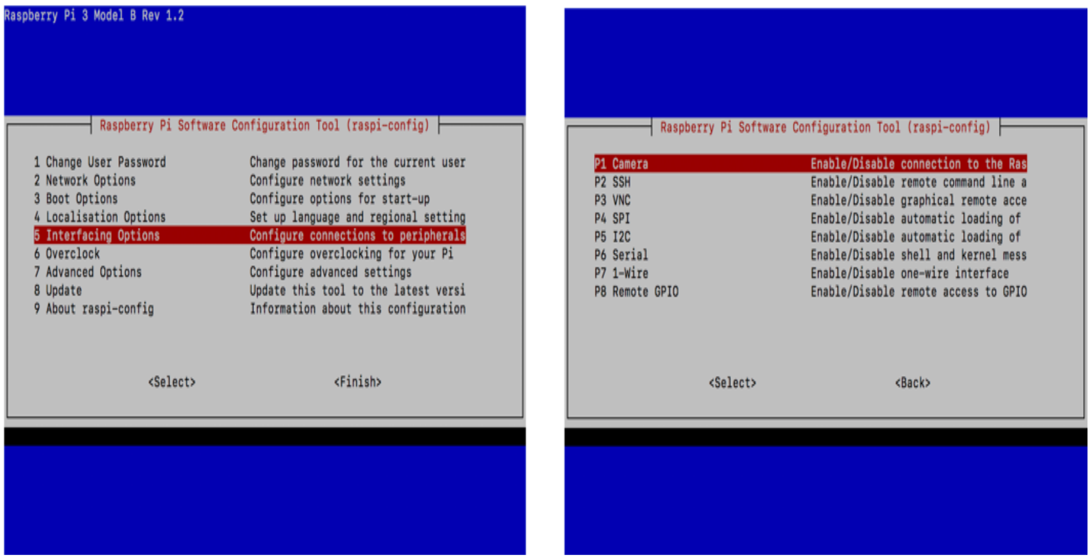
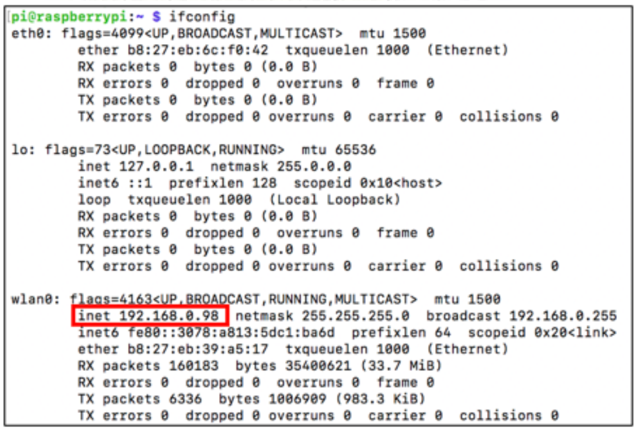

Raspberry Pi 3 Model B Rev 1.2
Raspberry Pi Software Configuration Tool (raspi-config)
1 Change User Password   Change password for the current user
2 Network Options   Configure network settings
3 Boot Options   Configure options for start-up
4 Localisation Options   Set up language and regional setting
5 Interfacing Options   Configure connections to peripherals
6 Overclock   Configure overclocking for your Pi
7 Advanced Options   Configure advanced settings
8 Update   Update this tool to the latest versi
9 About raspi-config   Information about this configuration
<Select>
<Finish>
Raspberry Pi Software Configuration Tool (raspi-config)
P1 Camera   Enable/Disable connection to the Ras
P2 SSH   Enable/Disable remote command line a
P3 VNC   Enable/Disable graphical remote acce
P4 SPI   Enable/Disable automatic loading of
P5 I2C   Enable/Disable automatic loading of
P6 Serial   Enable/Disable shell and kernel mess
P7 1-Wire   Enable/Disable one-wire interface
P8 Remote GPIO   Enable/Disable remote access to GPIO
<Select>
<Back>

Figure 4-12: Raspberry Pi software configuration tool

Enabling SPI interface is mainly used for the connection of the screen module in the Pioneer600 extension hat and enabling I2C interface is mainly used for the support of the RTC module in the Pioneer600 module. The SSH interface of the Raspberry Pi is turned off by default. After SSH is enabled, the external network can be used to remotely operate the Raspberry Pi. To log in to the Raspberry Pi using SSH, the first thing needs to know is the IP address of the Raspberry Pi in current network connection. It is able to enter *ifconfig* in the terminal to get the information of the network connection.



```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether b8:27:eb:6c:f0:42  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.98  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::3078:a813:5dc1:ba6d  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:39:a5:17  txqueuelen 1000  (Ethernet)
        RX packets 160183  bytes 35400621 (33.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6336  bytes 1006909 (983.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4-13: Internet connection information of Raspberry Pi

After got the IP address, users can use the terminals of other computers in the same network to remotely log in to the Raspberry Pi with following command code. The initial username of the login section is pi and the password is *raspberry*.

ssh username@IPaddress

The interface after login is the same as the interface for opening the terminal on the Raspberry Pi. For convenience to log in to the Raspberry Pi with the same address, Raspberry Pi allows the system to set a static IP. After that, there is no need to connect the monitor, mouse and keyboard. It is worth to note that this IP address is the intranet IP. If users want to use the computer of the external network to log in to the Raspberry Pi, users need to use some intranet penetration software accordingly.

4.2.2 Raspberry Pi (hardware) configuration

The previous section has shown how to install the Raspbian system for Raspberry Pi system and how to remotely log in to control the Raspberry Pi. There are still some common settings that need to be changed to allow subsequent program to run in the system. All the following command codes need to be input in the terminal of the Raspberry Pi.

System firework update

The following command codes help Raspberry Pi to update the software source, get the latest software list and upgrade the installed software. It can fix system bugs and help improve stability.

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get dist-upgrade -y
sudo rpi-update
```

Time zone selection

The following command codes is going to bring a time zone list on the screen, select the time zone based on the current region can help the Raspberry Pi to synchronize time according to the time zone when there is an Internet access.

```
sudo apt-get install ntpdate
sudo dpkg-reconfigure tzdata
```

System package and libraries installation

Pip is a package management system that can help the system to install and manage other software packages easily.

```
sudo apt install python-pip
```

Paramiko package allows the python program to use SSH protocol in the system. It will help the system to move data files from the monitoring device to server by using SCP command in the program. Since the server was built on the Raspberry Pi with some free NAT software, so it is able to use SCP to transmit data for current stage (OSXDaily, 2012).

```
sudo apt install python-paramiko
```

The main purpose of installing the Pandas package is to help the system organize temporary csv files.

```
sudo apt-get install python-pandas
```

To install the pyserial package in the system so that the Raspberry Pi can read data from the connected Arduino board via the serial port.

```
sudo pip install pyserial
```

Vim package used to edit files in the terminal.

```
sudo apt-get install vim
```

The following package is used for web server support.

```
sudo apt-get install python-lxml
```

Install Python-imaging and Pillow to support the display screen of the Pioneer600 extension hat and the I2C connection.

```
sudo apt-get install python-imaging
sudo apt-get install python-smbus
sudo pip install Pillow
sudo pip install pynmea2
```

Bluetooth configuration

The main purpose of the Bluetooth connection is that the Admin can obtain data through the mobile phone device and can debug the Raspberry Pi and change some related settings via Bluetooth. The Bluetooth communication uses the Bluetooth module that comes with the Raspberry Pi. So Raspberry Pi Bluetooth needs to do some initial setup in order to detect and start monitoring other Bluetooth devices when they are connected. The first thing is to update Bluetooth firmware.

sudo apt-get install pi-bluetooth

Install the python-bluetooth in order to help the python program to receive data in Raspberry Pi.

sudo apt-get install python-bluetooth

Pairing is required when the Bluetooth device is first time to connect with Raspberry Pi via Bluetooth. Enter the following command to get in the Bluetooth settings(MARTIN, 2016).

sudo bluetoothctl

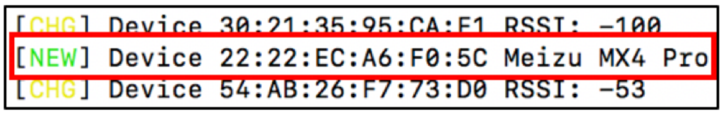
Turn on the Android mobile phone device Bluetooth and keep the device discoverable. Then let Raspberry Pi starts searching the nearby Bluetooth device.

agent on

default-agent

scan on

The unique addresses of all the Bluetooth devices around the Raspberry Pi will appear in the terminal and the result will look like as following Figure 4-14. The left part is the Bluetooth address of the device and followed by the device name on its right.



```
[CHG] Device 30:21:35:95:CA:F1 RSSI: -100
[NEW] Device 22:22:EC:A6:F0:5C Meizu MX4 Pro
[CHG] Device 54:AB:26:F7:73:D0 RSSI: -53
```

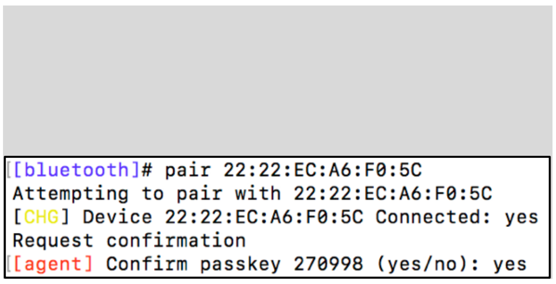
Figure 4-14: New Bluetooth device ID in command line

When the correct device shown on the list, stop searching the new device and then pair the Raspberry Pi with the device.

scan off

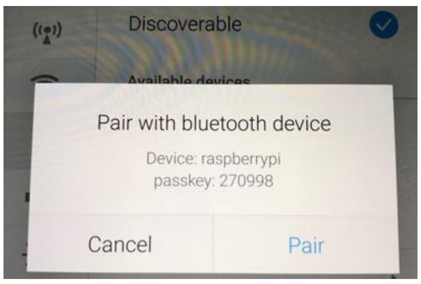
pair BluetoothAddress

Raspberry is going to show a confirm passkey for mobile device to confirm. Enter "yes" in the command and then click "Pair" on the mobile phone device to finish Bluetooth pairing.



```
[bluetooth]# pair 22:22:EC:A6:F0:5C
Attempting to pair with 22:22:EC:A6:F0:5C
[CHG] Device 22:22:EC:A6:F0:5C Connected: yes
Request confirmation
[agent] Confirm passkey 270998 (yes/no): yes
```

(a)



Discoverable

Pair with bluetooth device

Device: raspberrypi
passkey: 270998

(b)

Figure 4-15: Bluetooth pair process Raspberry Pi command line(a), Android mobile device(b)

Then enter the following command code to trust the Bluetooth device, so that the mobile phone device can connect to the Raspberry Pi device without having to enter the pairing password again.

trust BluetoothAddress

After exiting the Bluetooth settings, by entering the following code to display the Bluetooth service file.

sudo nano /etc/systemd/system/dbus-org.bluez.service

Edit the following line in the file

ExecStart=/usr/lib/bluetooth/bluetoothd

to

ExecStart=/usr/lib/bluetooth/bluetoothd -C

ExecStartPost=/usr/bin/sdptool add SP

Save and exit the file. After that, the mobile phone device is able to connect to the Bluetooth of Raspberry Pi and receive data.

Program code uploading and running

Through the SCP command on the terminal of the local computer, the local developed code file can be uploaded to the Raspberry Pi. The following code is an example of how to use scp in the terminal "*scp -r LocalDirection/FileName pi@IPAddree:Direction*".The following code uploads the entire HardwareProgram folder which the programs for the hardware on the local computer to the Raspberry Pi desktop which is at *192.168.0.99* in the local area network. Of course, it requires the password when uploading the files to the Raspberry Pi, the default password is same as the Raspberry Pi login password *raspberry*.

scp -r /Users/chenshu/Desktop/Final-Code/HardwareProgram pi@192.168.0.99:Desktop/

After completing the program upload, enter the following code to add autorun programs.

```
sudo nano /etc/rc.local
```

Add the following commands below the comment in the opened file, but leave the line *exit 0* at the end, then save the file and exit to let the Raspberry Pi can automatically run the following three programs when system starts to facilitate sensor access and data collection. The datils of how each program works will be described in the following section Code Document.

```
python /home/pi/Desktop/Hardware/TimeProgram.py &
python /home/pi/Desktop/Hardware/SensingSavingProgram/ SensingSavingProgram.py &
python /home/pi/Desktop/Hardware/PrepareSendingProgram /PrepareSendingProgram.py &
```

Until this step, the configuration of the Raspberry Pi is basically completed.

4.2.3 External communication modules configuration

The configuration in this section is mainly for the collection hardware, but the same configuration can be also applied to the server part of the next section since the server in this research was also built based on Raspberry Pi. Due to the infrequent changes of the connection of LoRa, Zigbee and GSM communication modules, the system is set to bind the communication modules with the USB ports in order that they communication modules can be recognized by the system when they are connected. And it only works when the communication modules connected to the specified port of Raspberry Pi as designed. So for the configuration, first plug the LoRa, Zigbee and GSM communication module into the USB ports of the Raspberry Pi as shown below.

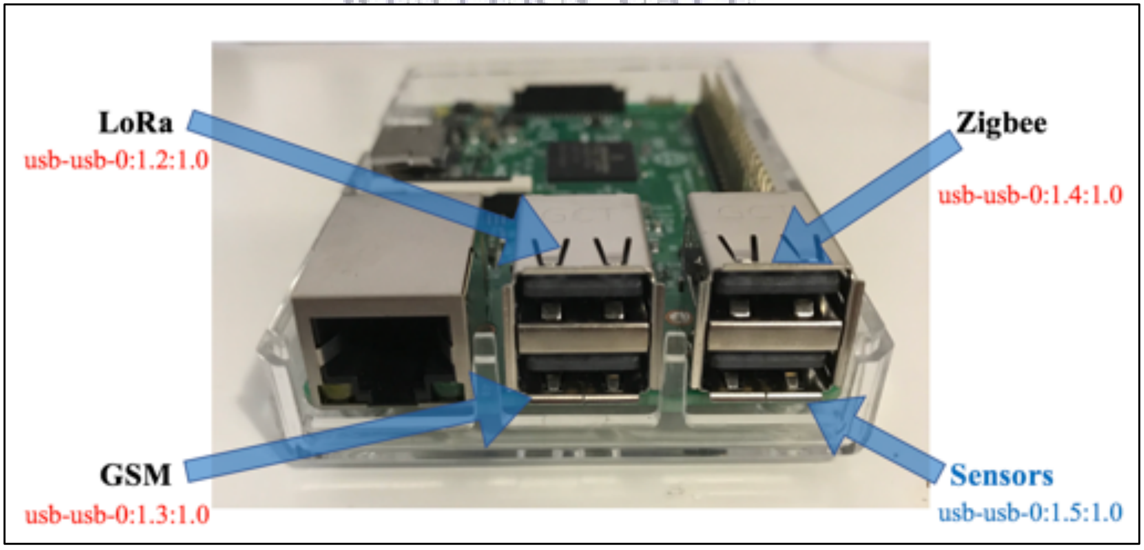
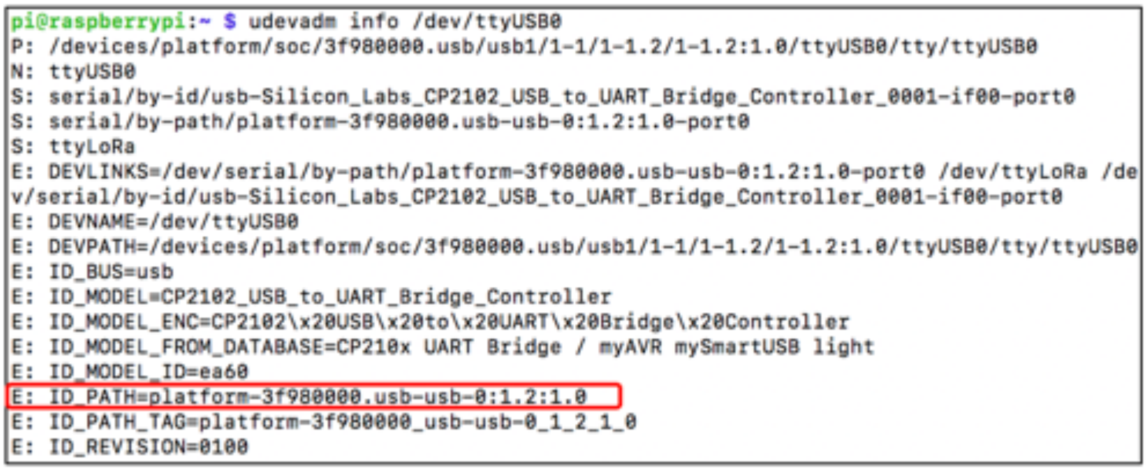


Figure 4-16: Raspberry Pi USB configuration

According to the naming rules of Raspberry Pi for newly inserted USB device, the names of the inserted three device will be ttyUSB0, ttyUSB1 and ttyUSB2. Then enter the following command in the terminal to view the USB device information under the different USB port names(Tu, 2017).

udevadm info /dev/ttyUSB0

It is going to get following information as Figure 4-17 below. The *ID_PATH* shows the real device port number that device actually connects.



```
pi@raspberrypi:~ $ udevadm info /dev/ttyUSB0
P: /devices/platform/soc/3f980000.usb/usb1/1-1/1-1.2/1-1.2:1.0/ttyUSB0/tty/ttyUSB0
N: ttyUSB0
S: serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0
S: serial/by-path/platform-3f980000.usb-usb-0:1.2:1.0-port0
S: ttyLoRa
E: DEVLINKS=/dev/serial/by-path/platform-3f980000.usb-usb-0:1.2:1.0-port0 /dev/ttyLoRa /de
v/serial/by-id/usb-Silicon_Labs_CP2102_USB_to_UART_Bridge_Controller_0001-if00-port0
E: DEVNAME=/dev/ttyUSB0
E: DEVPATH=/devices/platform/soc/3f980000.usb/usb1/1-1/1-1.2/1-1.2:1.0/ttyUSB0/tty/ttyUSB0
E: ID_BUS=usb
E: ID_MODEL=CP2102_USB_to_UART_Bridge_Controller
E: ID_MODEL_ENC=CP2102\x20USB\x20to\x20UART\x20Bridge\x20Controller
E: ID_MODEL_FROM_DATABASE=CP210x UART Bridge / myAVR mySmartUSB light
E: ID_MODEL_ID=ea60
E: ID_PATH=platform-3f980000.usb-usb-0:1.2:1.0
E: ID_PATH_TAG=platform-3f980000_usb-usb-0_1_2_1_0
E: ID_REVISION=0100
```

Figure 4-17: Connected USB ID_PATH

According to the above information to edit the new device name rules by enter the follow code in the terminal in order to edit the rule file.

sudo vim /etc/udev/rules.d/99-com.rules

Add the following code to the opened file. Then press "Esc" and enter: x! to save and exit file. The following codes help the communication modules to bind with the new port name and port number.

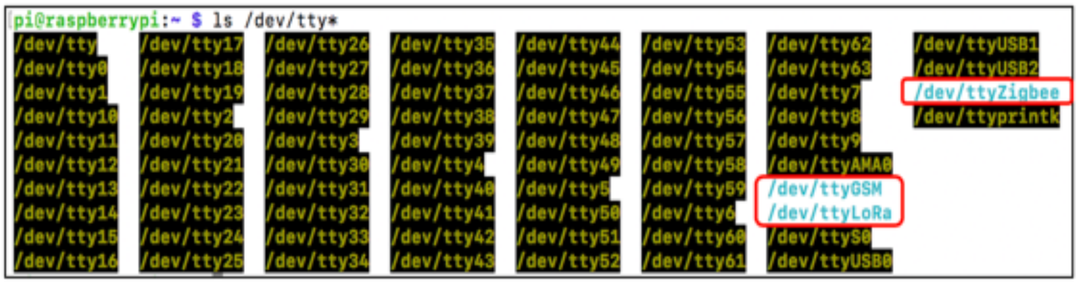
SUBSYSTEM=="tty", ENV{ID_PATH}=="platform-3f980000.usb-usb-0:1.2:1.0", SYMLINK+="ttyLoRa"

SUBSYSTEM=="tty", ENV{ID_PATH}=="platform-3f980000.usb-usb-0:1.3:1.0", SYMLINK+="ttyGSM"

SUBSYSTEM=="tty", ENV{ID_PATH}=="platform-3f980000.usb-usb-0:1.4:1.0", SYMLINK+="ttyZigbee"

After completing the above steps, enter the following code to view the currently connected device, you will find that the port name has been changed. When the device is connected, the Raspberry Pi is able to identify the device based on the port name.

*ls /dev/tty**



```
pi@raspberrypi:~ $ ls /dev/tty*
/dev/tty    /dev/tty17  /dev/tty26  /dev/tty35  /dev/tty44  /dev/tty53  /dev/tty62      /dev/ttyUSB1
/dev/tty0   /dev/tty18  /dev/tty27  /dev/tty36  /dev/tty45  /dev/tty54  /dev/tty63      /dev/ttyUSB2
/dev/tty1   /dev/tty19  /dev/tty28  /dev/tty37  /dev/tty46  /dev/tty55  /dev/tty7       /dev/ttyZigbee
/dev/tty10  /dev/tty2   /dev/tty29  /dev/tty38  /dev/tty47  /dev/tty56  /dev/tty8       /dev/ttyprintk
/dev/tty11  /dev/tty20  /dev/tty3   /dev/tty39  /dev/tty48  /dev/tty57  /dev/tty9
/dev/tty12  /dev/tty21  /dev/tty30  /dev/tty4   /dev/tty49  /dev/tty58  /dev/ttyAMA0
/dev/tty13  /dev/tty22  /dev/tty31  /dev/tty40  /dev/tty5   /dev/tty59  /dev/ttyGSM
/dev/tty14  /dev/tty23  /dev/tty32  /dev/tty41  /dev/tty50  /dev/tty6   /dev/ttyLoRa
/dev/tty15  /dev/tty24  /dev/tty33  /dev/tty42  /dev/tty51  /dev/tty60  /dev/ttyS0
/dev/tty16  /dev/tty25  /dev/tty34  /dev/tty43  /dev/tty52  /dev/tty61  /dev/ttyUSB0
```

Figure 4-18: Connected USB device with new names

The physical device connected with LoRa, Zigbee and GSM modules is shown below. Extra sensor modules can be connected to the Raspberry Pi via a USB hub. Of course, for most scenarios, it is not necessary to always plug three communication modules at the same time. Therefore, when the communication module is not connected, the USB port can still be used by the sensor without affecting the function of the sensor.

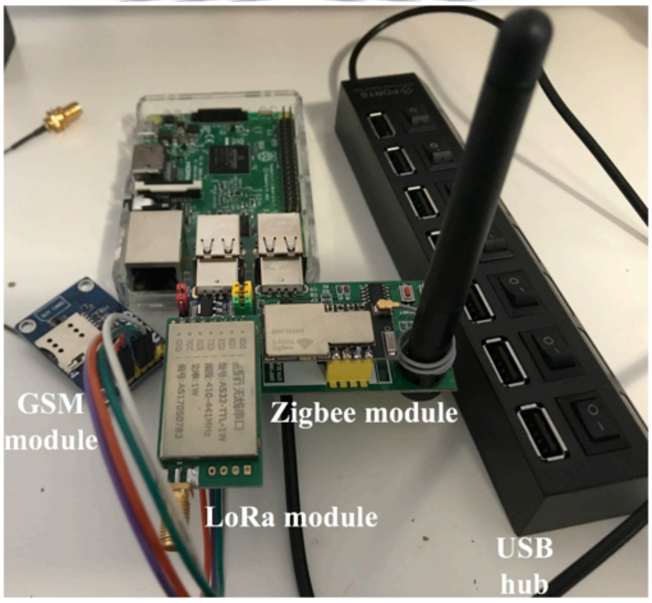


Figure 4-19: Device physical connection with communication modules

4.2.4 Sensor modules configuration

The sensors need to be properly connected to the Arduinos as described in section **4.1.1 "Hardware and wiring instructions"**. Then upload the respective code for each sensor to the connected Arduino board by using the Arduino software on the computer. The respective codes of the sensors are based on the sensor's data output interface and the manufacturer's instructions. The basic principle is to convert the digital signal into data according to the datasheet and output it through Arduino boards. The respective code file for each sensor is shown in the table below. The code will be attached in the appendix.

Table 4-5: Sensor modules code list

Sensor	Arduino code file
DHT22 temperature sensor	DHT22-Temp.ino
NEO-6M GPS sensor	NEO-6M-GPS.ino
PMSA003 PM sensor	PMSA003-PM.ino
ZE12-CO carbon monoxide sensor	ZE12-CO.ino

It has been mentioned in the **Chapter 3 Design Section** that the system is designed according to the modularity, so the sensor modules can be freely changed and added. The sensor modules can also be plugged to any USB port of the Raspberry Pi if there is no other communication module occupying the port. It is only necessary to ensure that the sensor module outputs data according to a certain format in order to be recognized by the Raspberry Pi. The output data format of the module composed of the sensor and Arduino follows the following rules:

The module first prints the "#" symbol and followed by the prompt of the sensor connected.

Example: *"#DHT22 connected"*

The sensor name and sensor data are then cyclically printed, each data is separated by comma.

Example: *"DHT22, Data1, Data2"*

It should be noted that if the sensor and Arduino is connected through UART protocol, which is the TXD pin of the sensor connected to the RXD pin of the Arduino. In the case the connection between the Arduino and the sensor needs to be disconnected when uploading code to Arduino, otherwise an error will occur during uploading progress. The reason is that Arduino cannot receive data from both RXD and USB port at same time.

4.2.5 Server configuration

A Raspberry Pi was used as the server and was connected to using the Oray's NAT service to use another Raspberry Pi in the home network environment combined with Oray's NAT service to build the server. So, for server side, the installation of the Raspberry Pi system is basically the same as the previous section "**Installation of Raspbian for Raspberry Pi**". For the server configuration, the operations of "**System firework update**", "**Time zone selection**", "**System package and libraries installation**" are basically the same as those on the Raspberry Pi (hardware). The only difference is in the section "**Program code uploading and running**". The program folder updated to the server is called MiddleServer and the IP address in the local home network is *192.168.0.101*.

scp -r /Users/chenshu/Desktop/Final-Code/MiddleServer pi@192.168.0.101:Desktop/

Then it is same here after completing the program upload, enter the following code to add autorun programs.

sudo nano /etc/rc.local

Add the following commands to the opened file to them automatically when the server boots up and enable the server receive data from the hardware device.

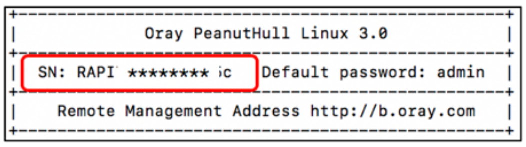
python /home/pi/Desktop/MiddleServer/MiddleServer.py &

python /home/pi/Desktop/MiddleServer/ZigbReceiving.py &

python /home/pi/Desktop/MiddleServer/LoRaReceiving.py &

python /home/pi/Desktop/MiddleServer/GSMReceiving.py &

Then set up the configuration of the Oray's NAT service. For the Oray's NAT service, download the Oray's client software for Raspberry Pi from the following URL: http://hsk.oray.com/download/download?id=25. After the installation is successful, the SN code, default password, and remote management address of this Raspberry Pi will be displayed in the terminal.



```
+------------------------------------------------+
|           Oray PeanutHull Linux 3.0            |
+------------------------------------------------+
| SN: RAPI ******** ic  Default password: admin  |
+------------------------------------------------+
|  Remote Management Address http://b.oray.com   |
+------------------------------------------------+
```

Figure 4-20: Oray local client information

After completing the client installation, it will automatically start running when the system starts. There are also some other related operation command codes for the Oray client shown below:

Table 4-6: Oray client command

Command code	Explanation
dpkg -r phddns	Uninstall the client
phddns start | *stop* | *restart*	Start | Stop | Restart the Oray client
phddns status | *version* | *reset*	Get the information of the Oray client

Then use the SN code as the account name, and *admin* as the initial password to log in to the following website *https://b.oray.com/passport/login* to complete the account binding. Follow the prompts on the web page to log in to the registered Oray account and enter to NAT management interface to add a new NAT mapping.

Table 4-7: Oray NAT mapping list

Application Name	Public IP address	Intranet host IP address
GSM Server	1c**********mypc.cn:15487	192.168.0.101:8080
Middle Server	1c**********mypc.cn:26671	192.168.0.101:22

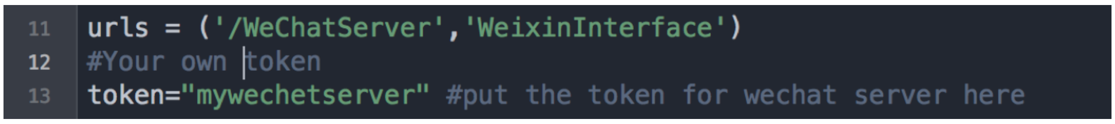
Since the research used the Oray free service, the IP and port of the public network were randomly generated. It should be noted that when setting up the local mapping network, *192.168.0.101* is the IP address of the server in local network and *8080* is the port for the server to establish a TCP connection to receive the data sent by the GSM module. Any other ports can also be set to establish a TCP connection, as long as the same corresponding port is used in the *GSMReceiving.py* program. And for the application Middle Server, since the default SSH port of the Raspberry Pi is *22*, so the mapped port should only be bound to *22*. Of course, the system can also use other NAT software to complete the above step, as long as it can provide a public IP address and port to the intranet server device that can be accessed by the external network.

In the research, another role of the middle server is to push data to the application layer. In the specific implementation process, WeChat public platform was adopted to push data to users. Users can follow the WeChat public channel and then obtain relevant data. This is very similar to the MQTT mode. The specific access method with WeChat public platform refers to the interface guide issued by WeChat officially (WeChat, 2017).

The following step was originally done on the local Raspberry Pi server with the help of the Oray NAT service, and another virtual machine in a Cloud space running Ubuntu was used as connection between WeChat and local Raspberry Pi server. After the configuration is completed, the locally finished program files were also uploaded to the virtual machine by using SCP command and started running the program at port 80 by using the following command.

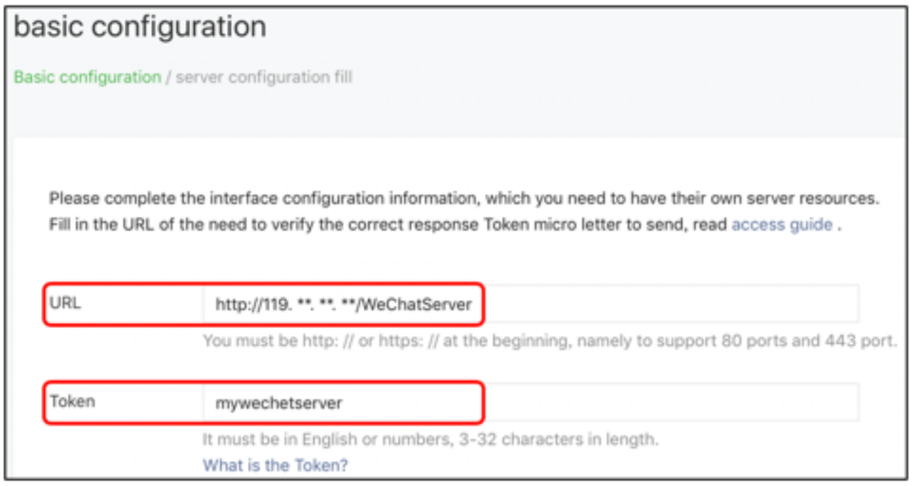
```
sudo python /home/ubuntu/Desktop/WechatServer/WechatServer.py 80
```

Then after completing the registration of the WeChat public platform, enter into the "basic configuration" settings to connect to the virtual machine by using the "urls" and "token" from the program as shown below.



```
urls = ('/WeChatServer','WeixinInterface')
#Your own token
token="mywechetserver" #put the token for wechat server here
```

Figure 4-21: WeChat server information


basic configuration

Basic configuration / server configuration fill

Please complete the interface configuration information, which you need to have their own server resources. Fill in the URL of the need to verify the correct response Token micro letter to send, read access guide .

URL: http://119. **. **. **/WeChatServer

You must be http: // or https: // at the beginning, namely to support 80 ports and 443 port.

Token: mywechetserver

It must be in English or numbers, 3-32 characters in length.

What is the Token?

Figure 4-22: WeChat URL and Token configuration

After completing the docking, the WeChat public channel can be subscribed by searching “AirPollutionMonitoringSystem” in WeChat or scanning QR code like Figure 4-23.

Figure 4-23: An example of WeChat public channel QR code

4.3 Implemented Device

4.3.1 The basic functions of the Device

Figure 4-20 below shows the basic functions that have been implemented in the device. Data is produced by connected sensors, processed and temporarily saved in Raspberry Pi hardware. It is then sent through Ethernet, Wi-Fi, LoRa, Zigbee and GSM module to server. Administrators are able to control the system through Bluetooth and users are able to read data via WeChat and PubNub.

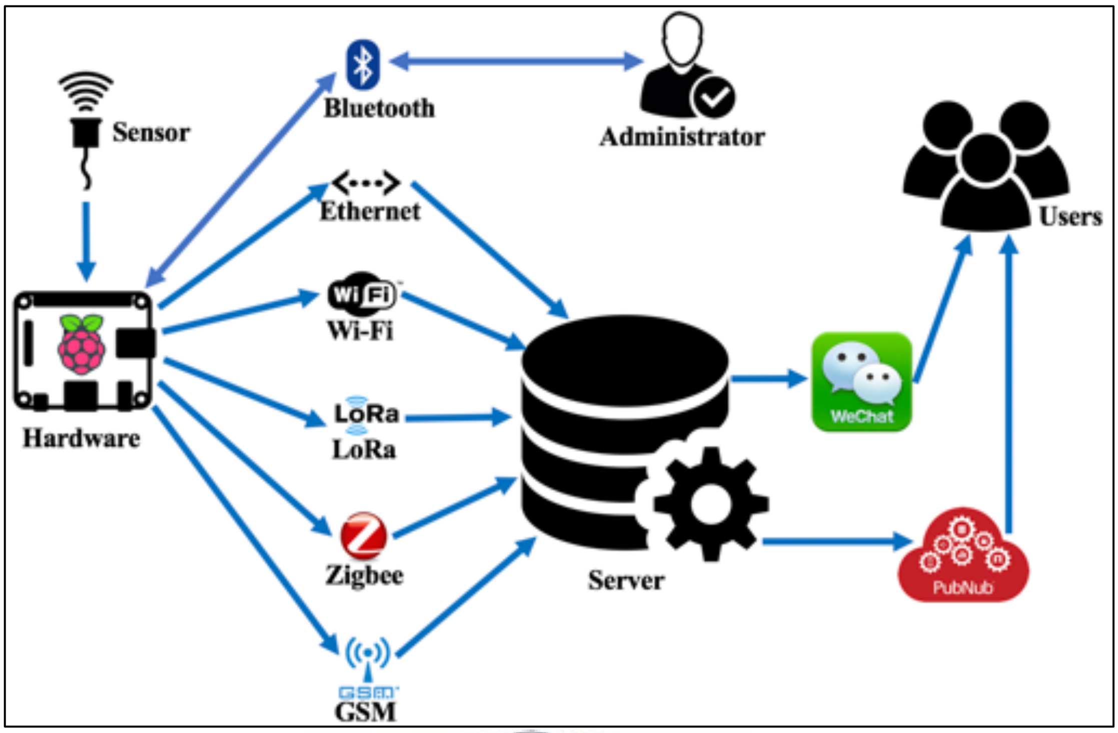


Figure 4-24: The basic functions of the implemented device

4.3.2 Finite State Machine of the Device

When the device is turned on, it will detect the sensor connection, if the sensor is correctly configured and connected, then the device will attempt to save the data with time stamp temporarily to the device except configured otherwise by the administrator. When the device doesn't not collect data, or the sensors are not properly configured / connected, then the progress will be stopped at the red circle "Don't Save Don't Send".

The device also detects the communication modules connected to the device and use the network selection algorithm combine with the selection of admin to get a best communication method for data transmission. If there is no available communication or admin chooses to do don't send data, then the then the process will be stopped at the red circle "Don't send".

When the data has been saved correctly and there is also an available communication, the device will move to data transmission progress. After the data successfully received at the server then users will be able to get the data at the red circle "Received at users". Where in the finite state machine figure, the blue circle is the start state and all the red circles are end states, the process loops from red back to blue.

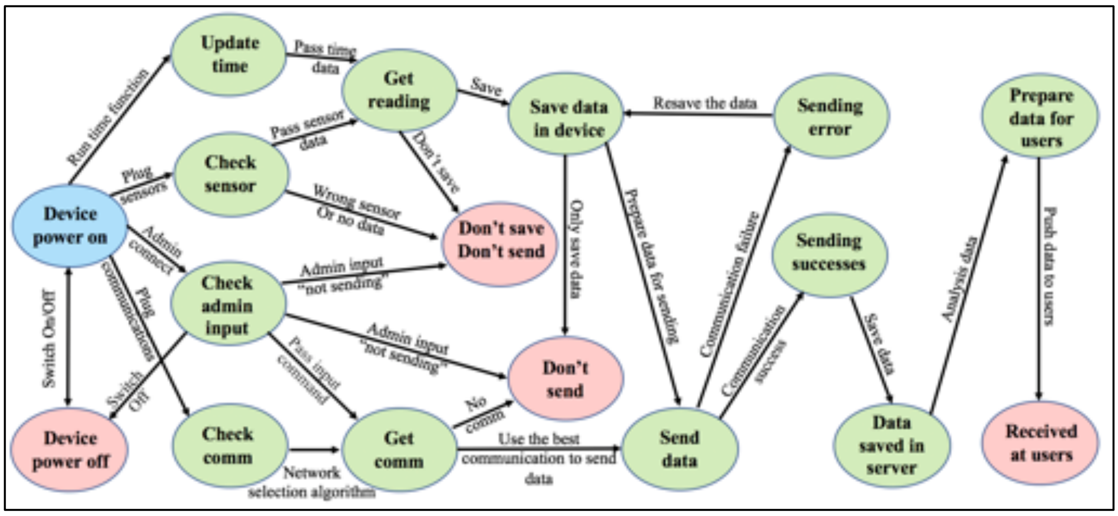


Figure 4-25: Finite state machine of the process

4.3.3 Physical Device

A picture of the finished prototype is shown in Figure 4-26. The hardware component contains of Raspberry Pi, Arduino Nano, Arduino Uno, a screen and sensors. There are four sensors, which are: PMSA003 PM sensor, DHT22 Temperature/Humidity sensor, NEO-6M GPS sensor and ZE12-CO Carbon Monoxide sensor. All the sensor modules are connected to the Raspberry Pi through USB port.

Due to limited space in the box used to house the hardware, this device currently only has Ethernet, Wi-Fi and Bluetooth communication for the data transmission. The LoRa, Zigbee or GSM modules were not installed. However, as described in previous subsections, these modules are replaceable, and the system supports the addition of more sensors in order to detect different gases. In essence, the figure only shows a possible combination this device could have, and other communications will be introduced in the following test section. A fan was included to allow air flow and keep the system cool while also circulating the air in the box to ensure a balance between the gas inside the box and that outside the box.

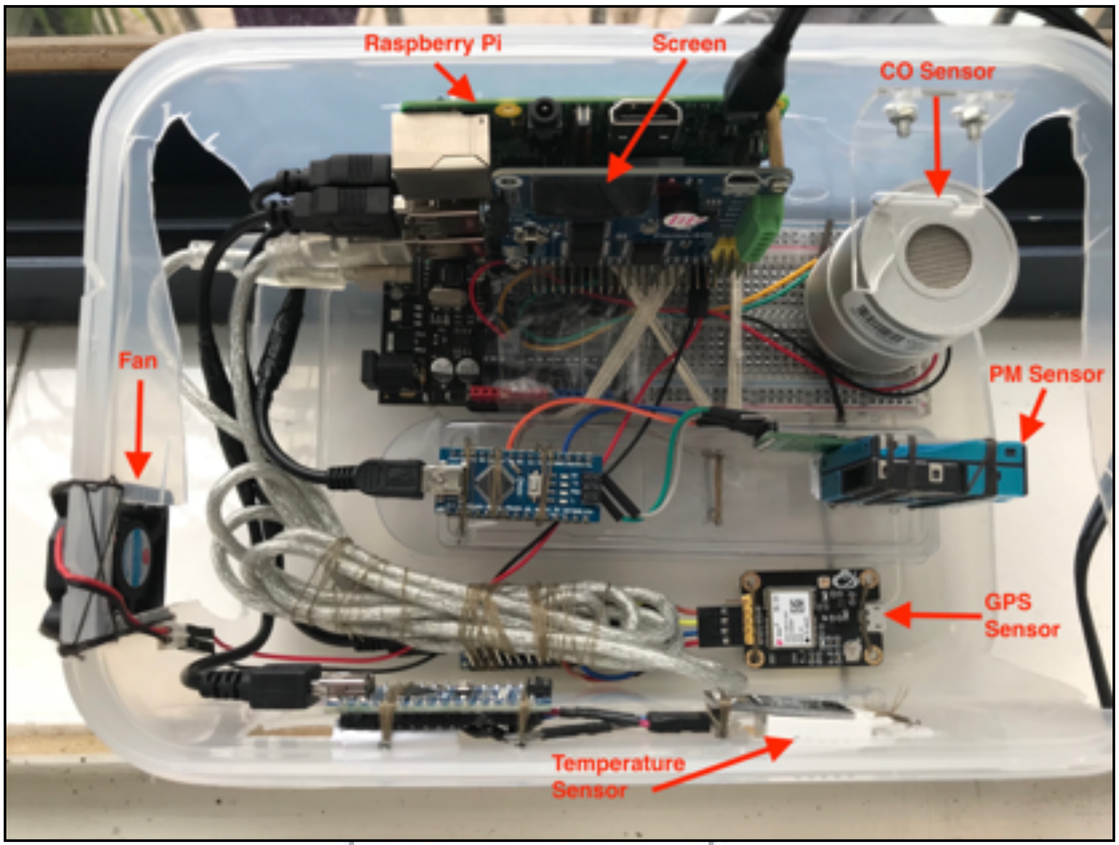


Figure 4-26: Top view of the first prototype

Figure 4-27 shows a zoomed picture of the data collected by the device. The readings are displayed on the screen, to allow users to read the data directly from the device. The data also can be received by Android smart phones through Bluetooth. For this example, reading contains time, humidity/temperature, PM1.0/PM2.5/PM10, GPS data and CO data.

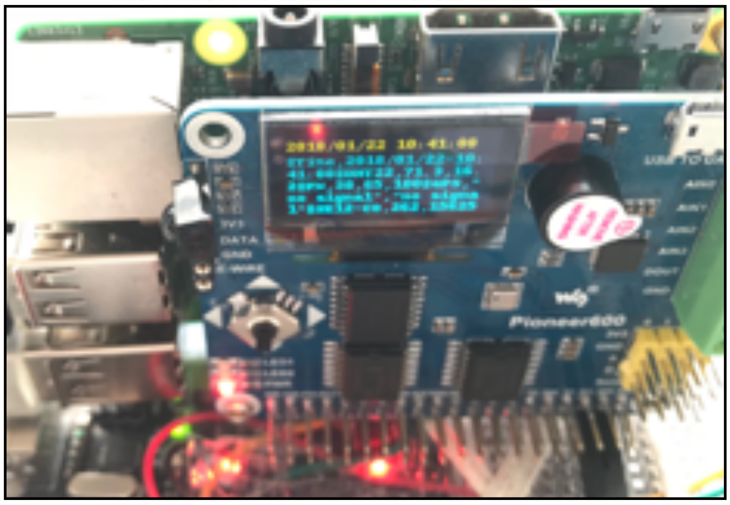


Figure 4-27: Reading example from the screen

The following picture is the outlook of the device. All components were installed in the box with only the extra GPS antenna affixed on the outside of the box. The antenna boosts the GPS sensor signal strength. The device was built small enough portable.

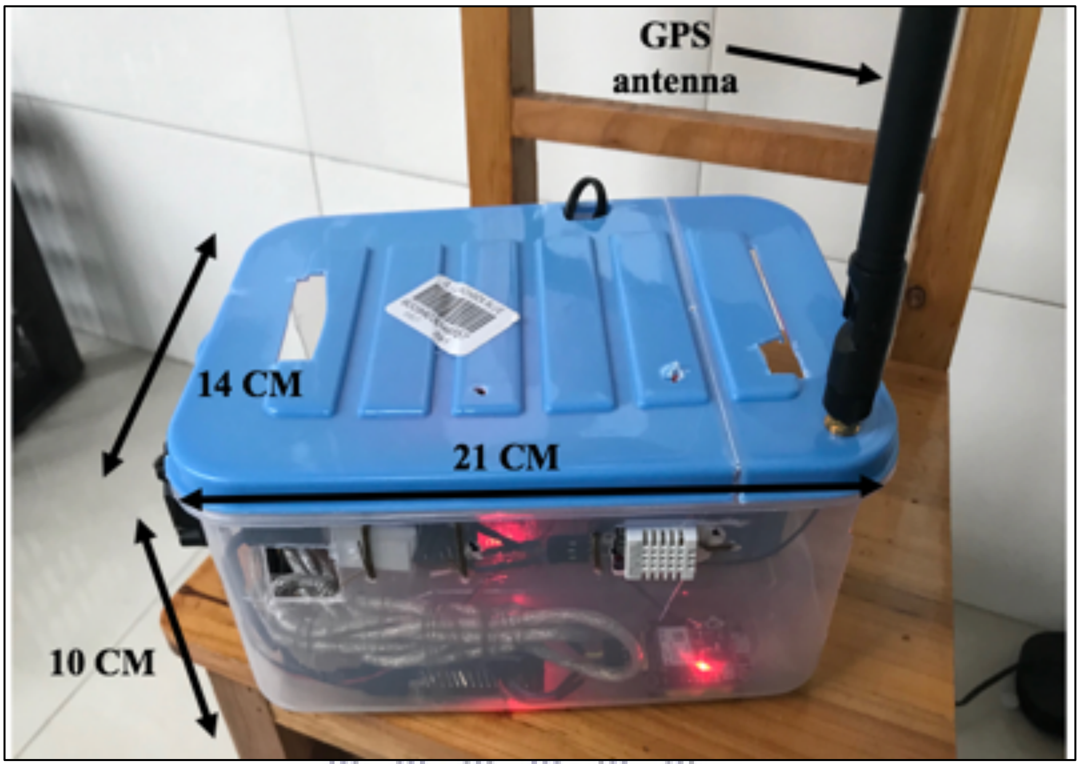


Figure 4-28: Outlook of the device

4.4 Code Document

The code contains four sections, which are: codes for the sensor modules, code for the hardware, middleware server code and code for the admin control App. The codes for sensor modules, hardware and middleware server are on the appendix pages.

4.4.1 Sensor modules code

C++ is the main language that is used to code the Arduinos and the sensor modules. There are different ways of data collection from the different sensors and since different sensors have been used in the system, the different codes for each sensor has to be uploaded on the different Arduinos.

Digital output sensor

A digital output sensor was used in this research work, which is the DHT22 temperature/humidity sensor. For this type of sensor, the output is a digital signal with discontinuous changed in levels. That is either 0, 1 and usually expressed in binary form. The high level is 1 and the low level is 0. On the Arduino, the function "digitalRead()" is be used to read the externally input digital signal and then convert it into data according to the sensor's datasheet.

Analog output sensor

Analog sensors were not used in this research. Arduino boards are equipped with analog pins hence able to read the level value from analog sensors and can convert voltage value between 0~5V into 0~1023 integer. The function "analogRead()" on the Arduino can be used to read analog input data.

Serial output sensor

These serial sensors used in this research work including the PMSA003 PM sensor, NEO-6M GPS and ZE12-CO sensor. These serial output sensors are based on UART communication, which can send signals directly to the RXD pin of Arduino. Input can be read with the Arduino "Serial.read()" function and combined with the sensor datasheet to get data.

4.4.2 Hardware code

The main language that used to code in the Raspberry Pi (hardware) was python. The three main codes are *TimeProgram.py*, *SensingSavingProgram.py* and *PrepareSendingProgram.py*.

Time program

The *TimeProgram.py* is used to help the hardware device synchronize time from either the Internet or RTC module. This ensures that the device has the correct time when collecting data. If the device has Internet access, the device time is mainly synchronized based on the network time, otherwise, the device time is synchronized with the RTC module time. Within the code, the python libraries imported are: *socket*, *time*, *os*, *smbus* and *threading*. Their specific functions are described below.

Time function class

+GetPiTime():

Input: None	Output: The second, minute, hour, week, day, month, year of the current time in an array
This function gets the current time separately in second, minute, hour, week, day, month and year from the system of Raspberry Pi. It then saves the data in hexadecimal format in an array. The function returns the time array.	

+PiSetModuleTime():

Input: None	Output: None
This function uses the GetPiTime() to get the current time on the system in order to update the time on the RTC module	

+GetModuleTime():

Input: None	Output: The second, minute, hour, week, day, month, year of the current time in an array
This function uses bus library to read the current time data from the RTC module then returns a time array similar to that of the GetPiTime() function.	

+ ModuleSetPiTime():

Input: None	Output: None
This function firstly uses the GetModuleTime() function to get the current time from the RTC module then uses the retrieved time to update the time of the Raspberry Pi.	

+isConnectd():

Input: None	Output: Result of Internet access in Boolean
This function is used to check if the device has Internet access. If it has Internet access it returns True, otherwise it returns False.	

+TimeInitial():

Input: None	Output: None
This function runs once at the device power up. The function uses isConnectd() function to check if the device has Internet access or not. If the device has Internet access, it then uses the Internet time to update the RTC module time, if it doesn't, then device will use the RTC module time to update the Raspberry Pi system time.	

+TimeUpdateFunction():

Input: None	Output: None
Prolonged operation without access to the Internet can result in errors within the RTC time. This function helps the RTC module to keep the current time in the case of the device has internet access. For example, every 30 days to use the time from the device to update the RTC module time.	

+TimeNTPRestart():

Input: None	Output: None
When the internet access is lost, this function starts running. This function helps the device to update the time when the Internet is reconnected.	

+main()

Input: None	Output: None
This function runs the TimeInital() function at beginning, then puts the TimeUpdateFunction() function and TimeNTPRestart() function in the thread to multi-tasking run the two functions simultaneously in order to ensure the device has correct time at all times.	

Sensing and saving program

The *SensingSavingProgram.py* is used to help the hardware device to recognize the connected sensors and save the readings from the sensors into the device, and only the configured the sensors can be recognized by the system. The code also contains the functions of reading admin's command from Bluetooth and displaying messages on the screen. In the code, firstly needs to import the necessary libraries for python: *serial*, *datetime*, *time*, *subprocess*, *threading*, *os*, *commands* and *signal*. The specific functions are described below.

Bluetooth functions

+BT_Files_Initial():

Input: None	Output: None
This function is used to initialize the admin's input command. The best communication mode is to set to "Auto" in the Bluetooth files folder. This would enable the system automatically to choose the best communication channel to be used to transmit data to the server. The Bluetooth folder is used to help the *SensingSavingProgram.py* and the *PrepareSendingProgram.py* to synchronous the commands received from the admin. The best communication medium is calculated using both *SensingSavingProgram.py* and *PrepareSendingProgram.py*.	

+BT_Receive():

Input: None	Output: None
This function helps the device to start listening to the data received by the Bluetooth module of the device (Raspberry Pi). After another mobile Bluetooth device connected to the Raspberry Pi, system will start to record the command messages received from the Bluetooth mobile device (Android phone) and subsequently execute the instructions. For example, a "reboot" messages would make the system reboot. If the message received from the admin is to choose the one of the communication methods, then this message will be written into the Bluetooth files folder which would in turn be used by *SensingSavingProgram.py.*	

+BT_Send_Function():

Input: None	Output: None
This function helps the system to send the sensor data and current best communication method (the result of the communication is from the Bluetooth files folder) for data transmission to the connected Bluetooth mobile device.	

+BTmain():

Input: None	Output: None
This function uses threading to simultaneously run the BT_Send_Function() and BT_Receive_Function(). These enable the device to send and receive data to and from the connected Bluetooth mobile device.	

Sensing functions

+getPort(ports)

Input: a list contains possible names for the port in string	Output: a list of the port name that sensors connected in string
This function uses the possible port names which appear in the system (such as "/dev/ttyUSB*") to list all sensors and communication modules connected on the USB port of the device. The function returns a list of port name that connected.	

+getRate(port,rates)

Input: specific port name in string and a list of the possible rates in integer	Output: the rate the sensor uses in integer
This function uses the input port name and tries with different rate numbers to connect with the sensor. For different sensors, it may use different rate for data communication. When the device connects to the sensor with the correct rate, the sensor prints out the right data. If the rate was wrong, random codes would be read from the sensor. Then the function will return the rate, but if the function could not find a correct rate for the sensor, the function then returns 0; which means the sensor data format is either not configured correctly according to the rules in the design or the sensor is not connected correctly.	

+getPortsRates(ports,rates)

Input: a list contains possible names for the port in string and a list of the possible rates in integer	Output: a matrix of the port names and rate numbers
This function uses the *getPort(ports)* to get the a list of the port name that are connected to sensors, then uses the *getRate(port,rates)* to get their corresponding rate numbers. The function returns a matrix of the port names and their counterpart rate numbers.	

+NewConnectionsCheck(ports)

Input: a list contains possible names for the port in string	Output: Boolean true or false
The function uses the function *getPort(ports)* to get a temporary connected port name list to compare with the ports in from the *getPortsRates(ports,rates)* in order to judge whether if the connected ports have changed. If the port has been changed which means either there is new sensor connected, or the current connected has been replaced or removed. The function returns if the sensor connection has been changed, true for yes, false for no.	

+NewConnectionsThread(ports,rates,tim)

Input: a list contains possible names for the port in string, a list of the possible rates in integer and a time in integer	Output: None
The function puts the function *NewConnectionsCheck(ports)* in a loop to check when the ports changes. The time in the input is the function running frequency.	

Saving functions

+connnectAll(PortsRates)

Input: a matrix of the port names and rate numbers	Output: a list of serials
The function uses port names and their counterpart rate numbers from the input matrix to contact all modules connected through serial, then returns a list of the connected serials.	

+readAll(serialList)

Input: a list of the connected serials	Output: readings from sensors in string
The function reads data from each connected serial sensor and combines them into a long string. The function returns the combined data in string format from the different sensors separated by the symbol "#"	

+closeAll

Input: a list of the connected serials	Output: None
The function closes all the serial connections in the input list.	

+ReadingThread(ports,rates,tim):

Input: a list contains possible names for the port in string, a list of the possible rates in integer and a time in integer	Output: None
The function uses the *getPortsRates(ports,rates)* function to get the sensor serial port and rate matrix, then uses the *connnectAll(PortsRates)* function to connect to all the sensor through serial. After the function got the readings from sensors, it will add a time stamp of current time and the ID number of the device to the data. It then saves the data in a csv file in the TempData folder on the device. The data is displayed on the LCD screen of the Pioneer600 Raspberry Pi extension hat and also sent to any connected Bluetooth mobile device using the Bluetooth functions. The data reading frequency is dependent on the input time.	

+main():

Input: None	Output: None
This function uses threading to run the function *BTmain()*, *NewConnectionsThread (ports,rates,time)* and *ReadingThread(ports,rates,time)* at simultaneously in order the device is able sensing the new connected sensors, also can read and save data from the sensors.	

Prepare and sending program

The *PrepareSendingProgram.py* is used to help the hardware device to prepare and organize the data saved in the device. The code can also detect the connected communication modules and use the network selection algorithm to choose the best communication method to transmit data to the server. The network selection algorithm will not only consider the current situation of the connections of the communications, but also according to the admin's input command to determine the current most suitable data transmission method. In the code, the imported libraries include: *paramiko*, *time*, *datetime*, *shutil*, *os*, *csv*, *socket*, *subprocess*, *re* and *urllib2*. The specific functions are described below.

Prepare functions

+DelFile(Dir,FileName):

Input: file direction in string and file name in string	Output: None
This function is used to delete data files after the data file has been uploaded to the server.	

+MoveFiles(FromFileDir,ToFileDir)

Input: the old folder direction in string and the new folder direction in string	Output: None
This function is used to move all the files from the temporary data folder to the local data folder. The temporary data folder is used by the *SensingSavingProgram.py* to save temporary data while the local data folder is used to save the merged data.	

+PutDataTogether(FileDir)

Input: folder direction	Output: None
The function checks all the data files in the input folder direction, then merge the data in different files into one file and delete the files have been merged.	

Network selection algorithm functions

+ checkBTUserInput():

Input: None	Output: None
This function checks the input of admin's selected communication from the Bluetooth files folder and update the admin's input to the algorithm. The default admin input is "Auto'.	

+checkIntInterface(IntInterface)

Input: internet interface name in string	Output: Boolean in true or false
This function is used to check whether the system has internet access for eth0 (ethernet) and wlan0 (Wi-Fi) interface. The function returns true if the interface has internet access, otherwise returns false.	

+NetworkSelection()

Input: None	Output: the communication name in string
This function takes precedence over the communication selected by admin, if the admin selected "Auto" which means admin will let the algorithm to decide which communication is the most suitable for the current situation. The function will start checking the connection of each communications in the following order Ethernet>Wi-Fi>LoRa>Zigbee>GSM, after the communication module connected device, the device will send some data to the server though this communication, only if the device receives a success message back from the server, then this communication module can be judged as a suitable communication method. The function will not only consider the connection of the communications, but also the signal strength of each communication. The communication with low signal strength will be past to ensure the stability of data transmission; when there are no suitable communications for data transmission, the function will return "Save" in string, which means just save data, doesn't transmit data. In the case of admin selected one communication and the communication is connected with good signal strength then the function will return the communication name as the best communication meted in string; when the admin selected communication is not available, then the function will return "Save" in string as well.	

Sending functions

+SCPSend (ServerIp, ServerPort, ServerUsername,ServerPassword,LocalDataFileDir, ServerDataFileDir)

Input: server IP address in string, server port number in integer, server username in string, user password in string, local data folder direction in string and server data folder direction in string	Output: Boolean true or false
The function will first check if the system has internet access then use SCP instruction to transmit all the data files in the local data folder to the server data folder. After all the files have been transmitted successfully, the function will delete the local data files and return true, otherwise return false.	

LoRaSend(LoRaPort,LoRaRate,LocalDataFileDir)

Input: port name of the LoRa module connected in string, port rate of the LoRa module connected in integer and local data folder direction in string	Output: Boolean true or false
The function will first check the connection of the LoRa module, connect with the module using the port name and rate. The function then reads and sends each line of the data files from the local data folder to the LoRa module and then to the server. The server also sends a reply message confirming that data has been received. After all the data have been sent to the server, the function will delete the local data files and return true, otherwise return false.	

ZigbSend(ZigbPort,ZigbRate,LocalDataFileDir)

Input: port name of the Zigbee module connected in string, port rate of the Zigbee module connected in integer and local data folder direction in string	Output: Boolean true or false
The use of the this function is similar to that of the *LoRaSend()* function. The function checks for the connection of the Zigbee module and then send each line of the data files from the local data folder to the Zigbee module and then to the server. After all the data have been sent to the server, the function will delete the local data files and return true, otherwise return false.	

+GSMSend(GSMPort,GSMRate,GSMServerIp,GSMServerPort,LocalDataFileDir)

Input: port name of the GSM module connected in string, port rate of the GSM module connected in integer, GSM server IP address in string, GSM server port number in integer and local data folder direction in string	Output: Boolean true or false
All the codes for the GSM module are written using AT command. The function will first check the connection of the GSM module and connect with the module using the port name and rate. This is similar to the *LoRaSend()* and *ZigbSend()* function. It read and send the data line by line from the data files to the TCP client at the server side. The function then deletes the data files in the local data folder. Function returns true if data have been sent successfully, otherwise it returns false.	

+main():

Input: None	Output: None
The main function firstly uses the prepare functions to prepare data and move the data from the temporary data folder to the local data folder. Then uses the network selection function to get the current suitable communication and uses the sending functions to transmit data to the server.	

4.4.3 Middle server

The main language that used to code in the server is python. The program consists mainly of two parts, the first one is the middle server program and the second one is the WeChat server program.

Middle server program

The *MiddleServer.py* is used to help the server to prepare the data received at the server side. The program has the function to send the prepared data to the WeChat server (in the research, the middle server and the WeChat server were not deployed in the same computer, so if the two servers are deployed at same place, then this sending function is not necessary). Except the prepare function and sending function, there are also several additional functions for receiving data, such GSM receiving function, LoRa receiving and Zigbee receiving function. In the code, firstly needs to import the necessary libraries for python: *sys*, *csv*, *operator*, *os*, *time*, *paramiko*, *datetime*, *shutil*, *glob* and *pandas*. The specific functions are described below.

Prepare and sending functions

The functions in the prepare section and the sending section basically are basically the same as the prepare functions and sending functions in *PrepareSendingProgram.py* of the hardware device.

+DelFile(Dir,FileName):

Input: file direction in string and file name in string	Output: None
This function is used to delete the data files after the data file has been uploaded to the server.	

+MoveFiles(FromFileDir,ToFileDir)

Input: the old folder direction in string and the new folder direction in string	Output: None
This function is used to move all the files from the temporary data folder to the local data folder. The temporary data folder is for receiving the data from the monitoring devices and the local data folder is used a database for the server.	

+PutDataTogether(FileDir)

Input: folder direction	Output: None
The function checks all the data files in the input folder direction, then merge the data in different files into one file and delete the files have been merged.	

+SCPSend (ServerIp, ServerPort, ServerUsername,ServerPassword,LocalDataFileDir, ServerDataFileDir)

Input: WeChat server IP address in string, WeChat server port number in integer, WeChat server username in string, WeChat user password in string, local data folder direction in string and WeChat server data folder direction in string	Output: Boolean true or false
The function will first check if the system has internet access then use SCP instruction to transmit all the data files in the local data folder to the WeChat server data folder. After all the files have been transmitted successfully, the function will delete the local data files and return true, otherwise return false.	

Receiving functions

+LoRaReceiving(LoRaPort,LoRaRate,TempDataFileDir)

Input: port name of the LoRa module connected in string, port rate of the LoRa module connected in integer and temporary data folder direction in string	Output: None
The function will first check the connection of the LoRa module and connect with the module using the LoRa port name and LoRa rate. The port name depends on the USB port that LoRa module connected. The function then reads the date received at the LoRa module and saves it to the temporary data folder in the server.	

+ZigbReceiving(ZigbPort, ZigbRate,TempDataFileDir)

Input: port name of the Zigbee module connected in string, port rate of the Zigbee module connected in integer and temporary data folder direction in string	Output: None
The function will first check the connection of the Zigbee module and connect with the module using the Zigbee port name and Zigbee rate. The function then reads the date received at the Zigbee module and saves it to the temporary data folder in the server.	

+GSMReceiving(HostNamee, GSMPort, ,TempDataFileDir)

Input: TCP host name in string, TCP port number in integer and temporary data folder direction in string	Output: None
The function uses TCP communication protocol to build a client to receive the data sent from GSM modules. After the TCP server receives the data, the function saves the data to the temporary data folder in the server.	

WeChat server program

The *WechatServer.py* is used to interface with the WeChat public platform. When a user requests data through WeChat, WeChat sends the request to the program. The program then sends the requested data to the WeChat platform, for it to be pushed back to the user through WeChat. In the code, the necessary libraries for python include: *os*, *csv*, *web*, *time*, *hashlib* and *lxml*. The specific functions are described below.

+getData(userInput)

Input: area name of user input in string	Output: data set in string
The function checks the area name of the user input and then prepares the latest data for the area. The function returns the data for the input area in string	

+GET(self)

Input: object	Output: self variables
The function uses function web.input() to get the input parameters and returns the parameters.	

+POST(self)

Input: object	Output: None
The function gets the parameters from the post and performs XML parse to get the content input from the user. It then replies to the user with corresponding data according to the user's input.	

4.4.4 Admin control app

The admin app was built by using MIT App Inventor2 for Android phones. The 'Designer' and 'Blocks' of the app are shown below.

Designer

This is the design of interface for the app. The functionality of this app is to help the admin to control the device. Admin can start and stop the data transmission and also manually choose communication method.

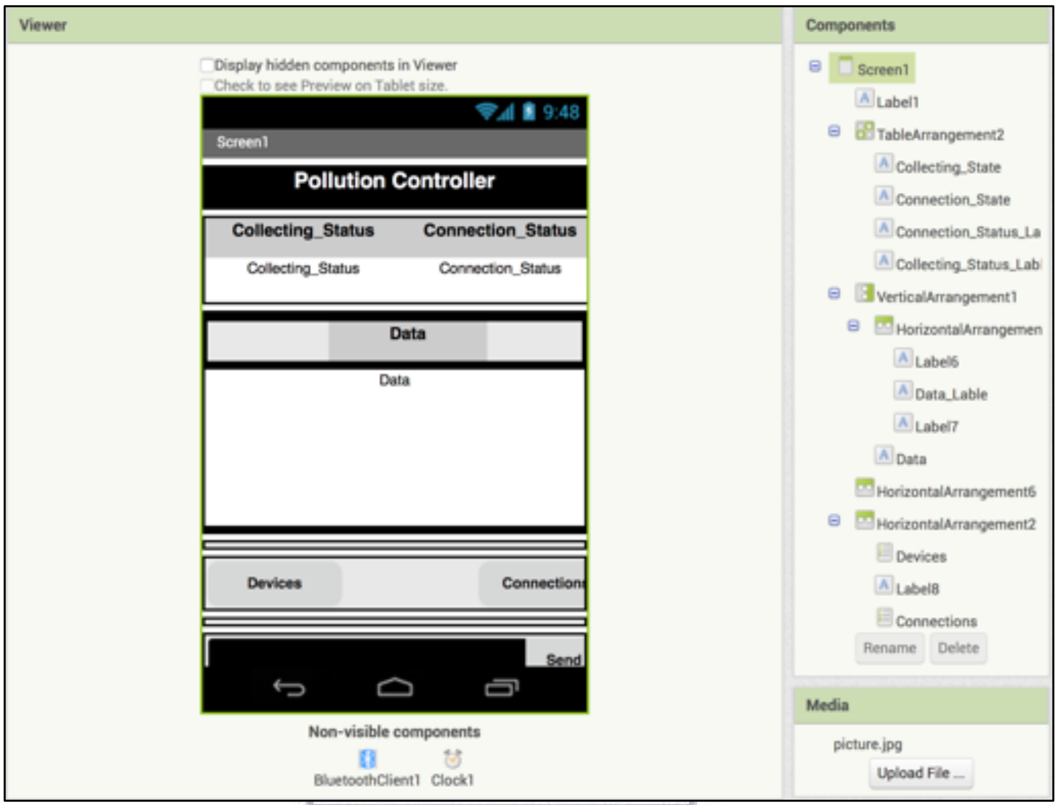
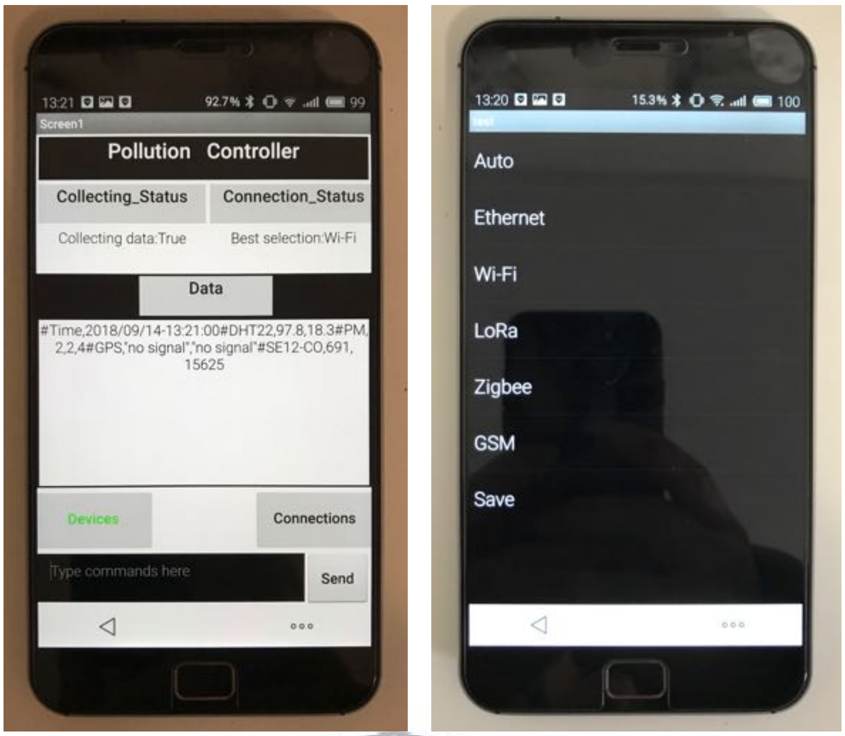


Figure 4-29: Admin control app interface

After the admin app connected to the monitoring device through Bluetooth, the current collecting status, connection status and data will be displayed on the screen as shown in Figure 4-30 below. Admin user is also able to change the communication method through the "Connection" button or from the user command input box.



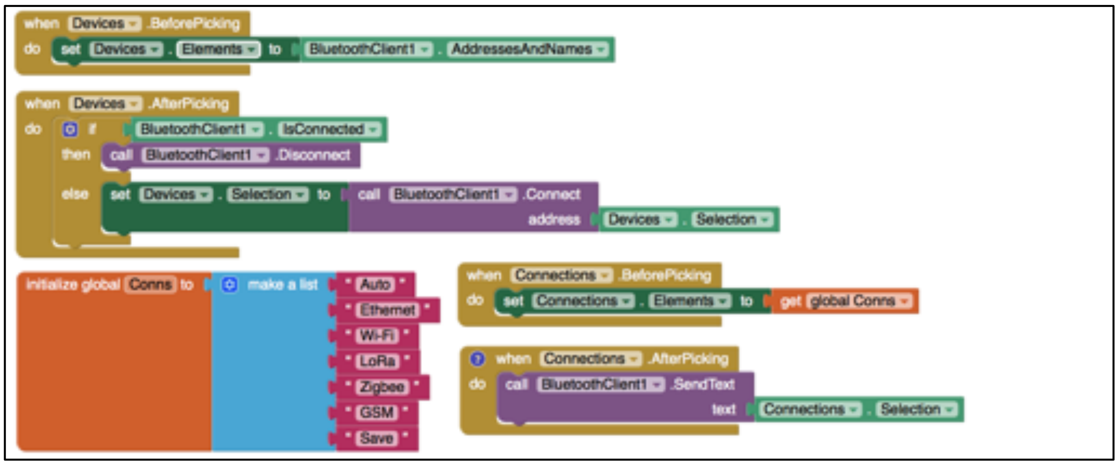
```
#Time,2018/09/14-13:21:00#DHT22,97.8,18.3#PM,
2,2,4#GPS,"no signal","no signal"#SE12-CO,691,
15625
```

(a) (b)

Figure 4-30: App interface on device (a), Communication screen (b)

Blocks

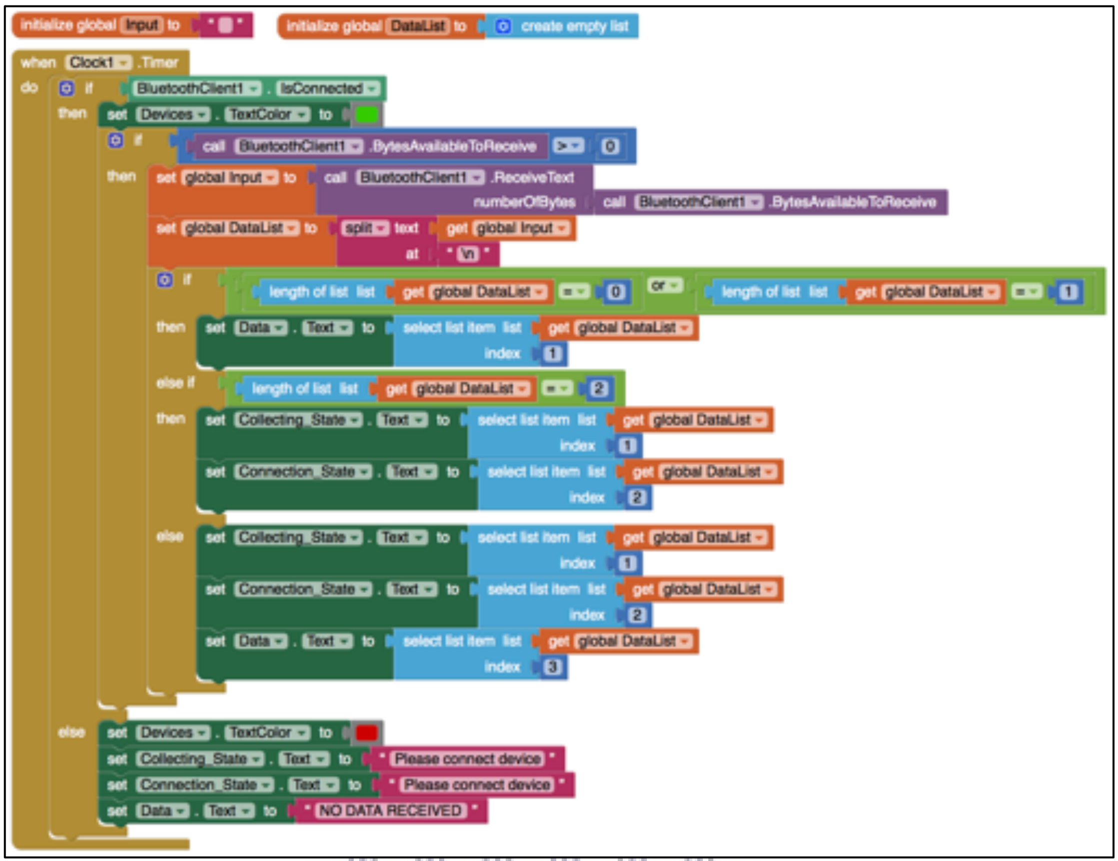
These blocks show the functions in this app. When the admin types into the app or the app receives messages from the monitoring device, the app will change the text in the interface to achieve the corresponding feedback. The following code corresponds to the Bluetooth client calls and communication list. It creates a communication list that can be selected by admin user.



```
when Devices .BeforePicking
do  set Devices . Elements to BluetoothClient1 . AddressesAndNames

when Devices .AfterPicking
do  if BluetoothClient1 . IsConnected
    then call BluetoothClient1 .Disconnect
    else set Devices . Selection to call BluetoothClient1 .Connect
                                                 address Devices . Selection

initialize global Conns to make a list " Auto "
                                        " Ethernet "
                                        " Wi-Fi "
                                        " LoRa "
                                        " Zigbee "
                                        " GSM "
                                        " Save "

when Connections .BeforePicking
do  set Connections . Elements to get global Conns

when Connections .AfterPicking
do  call BluetoothClient1 .SendText
                         text Connections . Selection
```

Snippet 4-1: Blocks of the app built by MIT App Inventor (a)

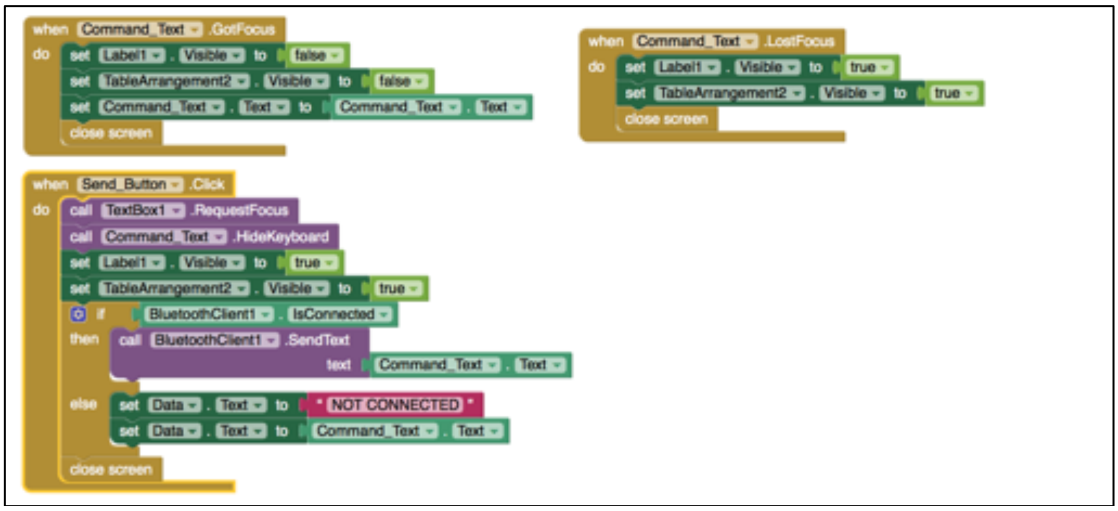
The following code is corresponds to receiving data from the Bluetooth module of the monitoring device and display the data it on the screen.



```
initialize global Input to " "        initialize global DataList to  create empty list

when Clock1 .Timer
do  if  BluetoothClient1 . IsConnected
    then  set Devices . TextColor to 
          if  call BluetoothClient1 .BytesAvailableToReceive  >  0
          then  set global Input to  call BluetoothClient1 .ReceiveText
                                         numberOfBytes  call BluetoothClient1 .BytesAvailableToReceive
                set global DataList to  split text  get global Input
                                              at  " \n "
                if  length of list list  get global DataList  =  0  or  length of list list  get global DataList  =  1
                then  set Data . Text to  select list item list  get global DataList
                                                          index  1
                else if  length of list list  get global DataList  =  2
                then  set Collecting_State . Text to  select list item list  get global DataList
                                                                      index  1
                      set Connection_State . Text to  select list item list  get global DataList
                                                                      index  2
                else  set Collecting_State . Text to  select list item list  get global DataList
                                                                      index  1
                      set Connection_State . Text to  select list item list  get global DataList
                                                                      index  2
                      set Data . Text to  select list item list  get global DataList
                                                          index  3
    else  set Devices . TextColor to 
          set Collecting_State . Text to  " Please connect device "
          set Connection_State . Text to  " Please connect device "
          set Data . Text to  " NO DATA RECEIVED "
```

Snippet 4-2: Blocks of the app built by MIT App Inventor (b)

The following code corresponds to the user command input box, and it is used by the user to send command to the monitoring device through Bluetooth.



```
when Command_Text .GotFocus
do  set Label1 . Visible to  false
    set TableArrangement2 . Visible to  false
    set Command_Text . Text to  Command_Text . Text
    close screen

when Command_Text .LostFocus
do  set Label1 . Visible to  true
    set TableArrangement2 . Visible to  true
    close screen

when Send_Button .Click
do  call TextBox1 .RequestFocus
    call Command_Text .HideKeyboard
    set Label1 . Visible to  true
    set TableArrangement2 . Visible to  true
    if  BluetoothClient1 . IsConnected
    then  call BluetoothClient1 .SendText
                                  text  Command_Text . Text
    else  set Data . Text to  " NOT CONNECTED "
          set Data . Text to  Command_Text . Text
    close screen
```

Snippet 4-3: Blocks of the app built by MIT App Inventor (c)

4.5 Summary

In this section details of the software installation process, hardware connectivity, device installation process, server setup and working status of the completed system were presented. The complete air monitoring system was constructed as described in this chapter and now able to detect $PM_{2.5}$, PM_{10}, temperature, humidity, CO and GPS data. It is also able to upload data to the server through Ethernet, Wi-Fi, LoRa, Zigbee or GSM communications. Testing and data capturing processes are discussed in the next section.

5. Test and Results

The basic components, features and functions of the monitoring system were discussed in the previous chapter. This chapter will thus focus on the testing of the hardware part and how the system compares to the existing system; how users get stared with the system and errors configuration will be descripted in the User’s Guide.

5.1 Testing Document

5.1.1 Description of the completed system

The major function of the system is collection of real time air pollution data. In order to facilitate the use threshold and learning costs of the system, also the system has tried to simplify the operation procedure during the measurements. As long as the hardware is connected with right sensors and put a right place then it is able to monitor the environment around the device. The only process that requires manual intervention is the administrator’s choice of date transmission method, other functions were completed by the internal algorithm of the system. As described in the implementation the Bluetooth module is used to control the device, Ethernet, Wi-Fi, LoRa, Zigbee and GSM module are used as data transmission support. WeChat is used as client platforms to present the data to the users. For the future, the research will try to apply Pubhub and develop a mobile app and website to display the data as well and integrate more different sensors into the system.

5.1.2 Test 1: Basic functions test

This section is mainly to test the basic functions of the device such as mobile phone control through Bluetooth, sensors identification, data transmission of network selection algorithm, data reception of WeChat and the transmission range of the communication modules

Test environment

The main simulation scenario of the test is one in which the hardware device has been used to collect air quality information in a home environment. The communication module connected to the device include Wi-Fi, LoRa, Zigbee, GSM modules and the connected sensors include DHT22 temperature sensor and PM sensors, as shown in the Figure 5-1.

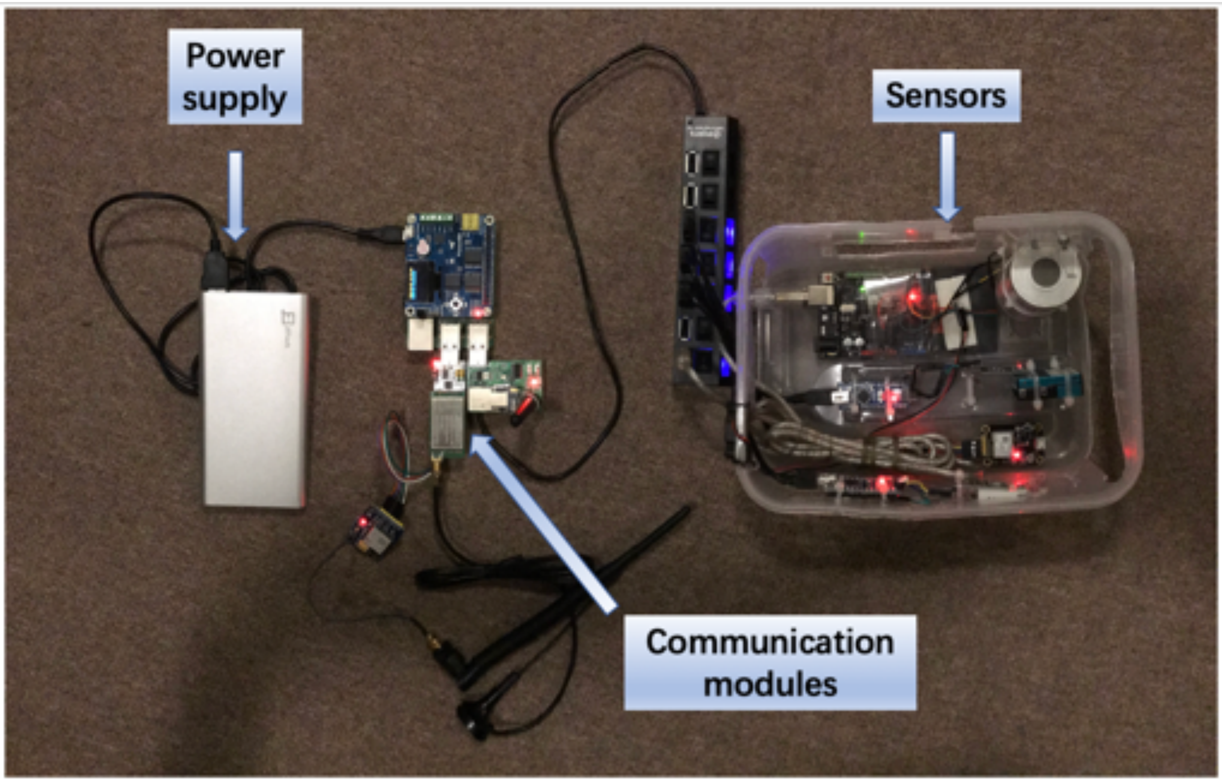


Figure 5-1: Device connected with communication modules and sensors

The other requirements used in the test include:

Admin control phone: Meizu MX4 Pro with system Android OS, v4.4.4, Bluetooth: V4.0
Home router: TP-Link Wireless ADSL2+ Modem Router (TD-W8968) with WLAN: 2.4GHz, IEEE802.11b/g/n
WeChat: Version 6.7.3
Smart phone (for WeChat client): Apple iphone6s with system iOS 10.1.1, Carrier: Cell C
USB current/voltage/power test instrument

Test execution

Before the test started, to connect the device through the USB current/voltage/power test instrument first then to the power supplier, in order that the power consumes of the device could be recorded by calculating the changes of the current and voltage goes through the device. Then to connect the hardware device to the home router via Ethernet cable, turn on the device and add the Wi-Fi information into the hardware device so that it can connect to the network through Wi-Fi. To connect the admin control app with the device and then do the following steps to observe the operation of the system: disconnecting Ethernet connection, switching off the router, disconnecting LoRa module, disconnecting Zigbee module, disconnecting GSM module and switching on the router. At the same time, the test tried to disconnect and reconnect the sensor to test whether the system can work properly. Finally, to check the data collected in real time on the WeChat side on smart phone.

The second part of the test was to establish the stable operating range when the system is in motion. This involves getting the maximum working distance under different communications. For this test, the server was connected with Wi-Fi, LoRa and Zigbee modules and placed in an apartment. The antennas were then placed outside the window. The hardware device was also connected with LoRa, Zigbee, GSM module and then carried in a car. The device was placed in the car to ensure the safety of the device, while the antennas were sucked on the roof of the car to allow for good signal for data transmission. From the apartment fixed, the car was driven away at low speed to observe the data transmission status of the device. The Google Maps on a smartphone was used to record the longest transmission point.

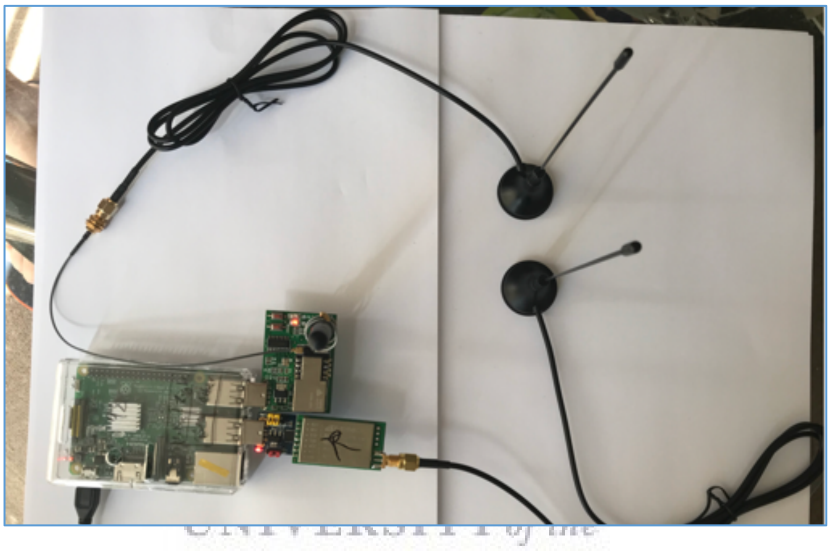


Figure 5-2: Raspberry Pi server placed at the apartment with LoRa and Zigbee modules

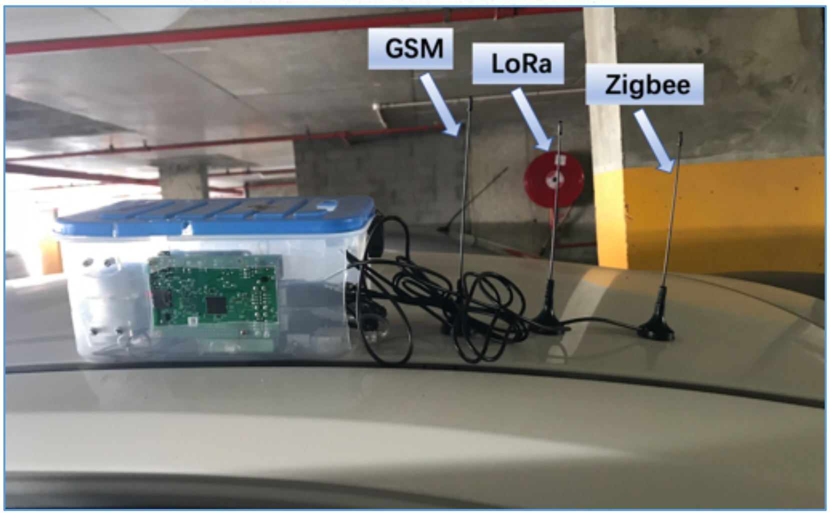
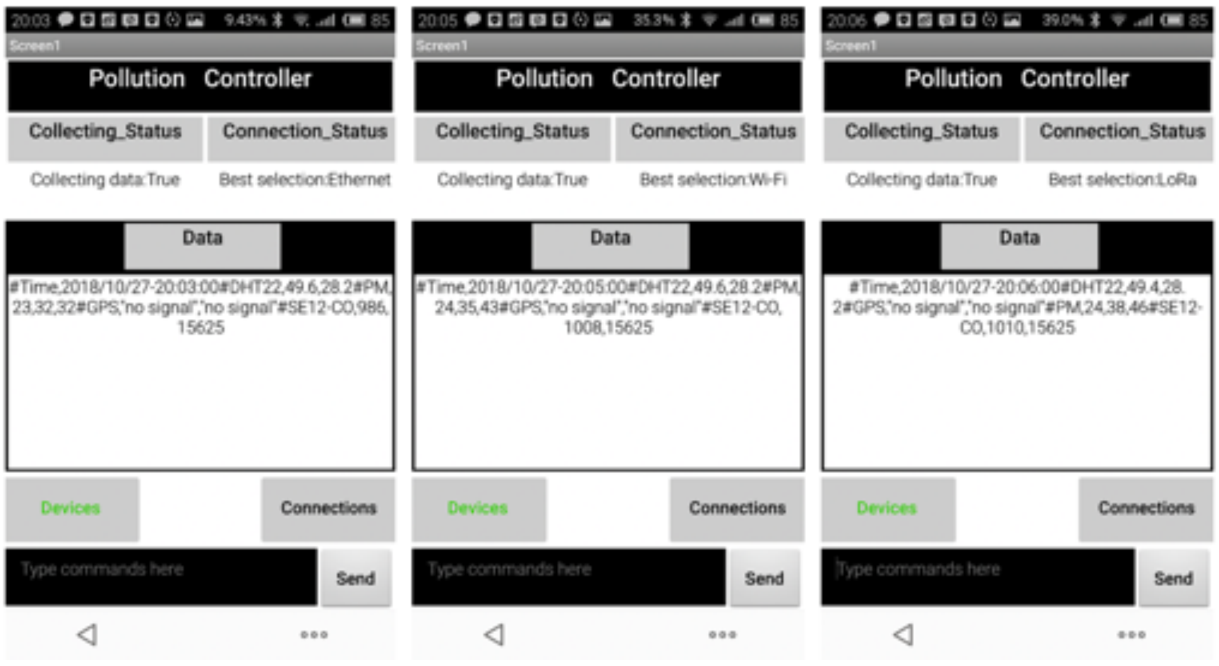


Figure 5-3: The device with GSM, LoRa and Zigbee module on the car

Result and analysis

Figure 5-2 and 5-3 show that the system can meet the requirements of system design. The default communication setting is "Auto", so the data was sent through the first priority communication method which is Ethernet. When the Ethernet cable was disconnected, the network selection algorithm automatically switches to the second priority communication method for data transmission which was Wi-Fi. When the router was switched off (both Ethernet and Wi-Fi disconnected), the device switched to the next priority communication method LoRa, Zigbee and GSM. When none of the communications were available, the system stored data within the device and waited until any communication links was reconnected.

In addition, the functions of the admin control app also worked well. The admin was able to use the admin control app to select any communication methods for data transmission for the device. If the selected communication method was unavailable then the device just saved the data. The device information was sent to the admin control app via Bluetooth. Screen shots of the control app taken during the test shown are shown in Figures 5-4 and 5-5. The data transmission was very stable, and the admin control app was also ran smoothly during the test.



Collecting data:True Best selection:Ethernet

#Time,2018/10/27-20:03:00#DHT22,49.6,28.2#PM,23,32,32#GPS,"no signal","no signal"#SE12-CO,986,15625

Collecting data:True Best selection:Wi-Fi

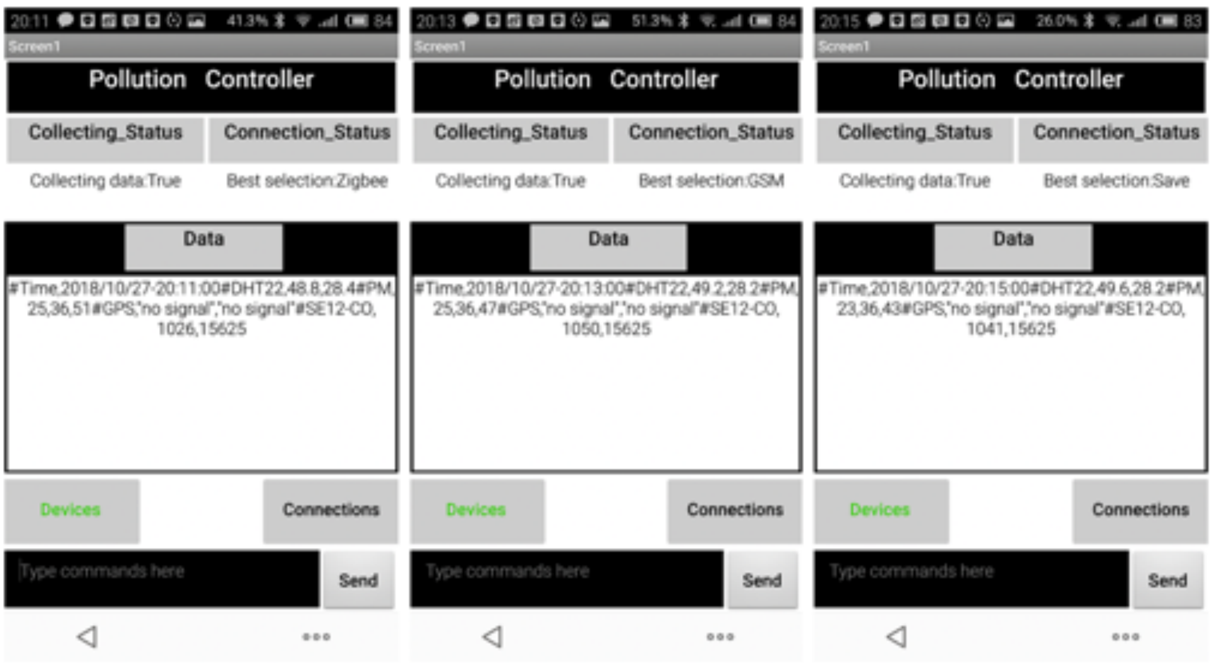
#Time,2018/10/27-20:05:00#DHT22,49.6,28.2#PM,24,35,43#GPS,"no signal","no signal"#SE12-CO,1008,15625

Collecting data:True Best selection:LoRa

#Time,2018/10/27-20:06:00#DHT22,49.4,28.2#GPS,"no signal","no signal"#PM,24,38,46#SE12-CO,1010,15625

(a) (b) (c)

Figure 5-4: Communications of the control app, Ethernet(a), Wi-Fi(b), LoRa(b)



(a)
Collecting data:True
Best selection:Zigbee

```
#Time,2018/10/27-20:11:00#DHT22,48.8,28.4#PM,
25,36,51#GPS,"no signal","no signal"#SE12-CO,
1026,15625
```

(b)
Collecting data:True
Best selection:GSM

```
#Time,2018/10/27-20:13:00#DHT22,49.2,28.2#PM,
25,36,47#GPS,"no signal","no signal"#SE12-CO,
1050,15625
```

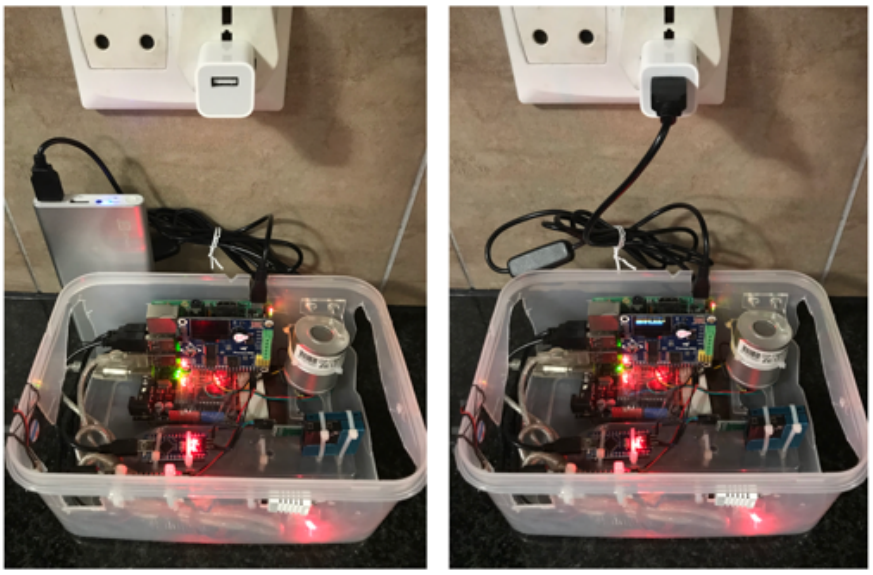
(c)
Collecting data:True
Best selection:Save

```
#Time,2018/10/27-20:15:00#DHT22,49.6,28.2#PM,
23,36,43#GPS,"no signal","no signal"#SE12-CO,
1041,15625
```

(a) (b) (c)

Figure 5-5: Communications of the control app, Zigbee(a), GSM(b), Save(c)

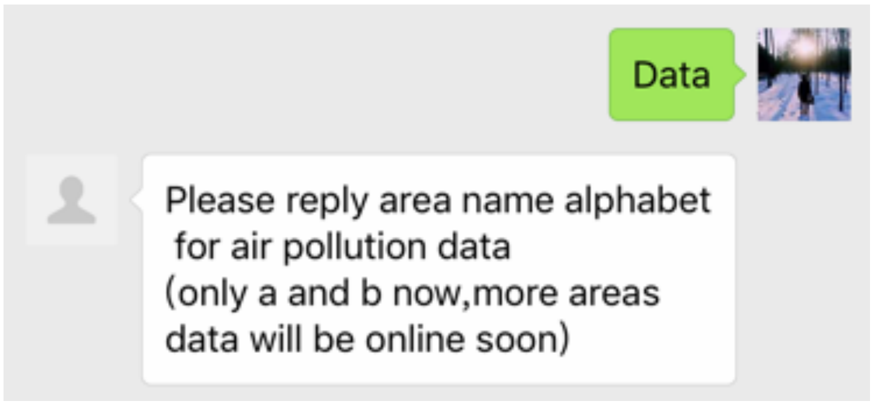
Figure 5-6 below show that the device is very flexible and can be used in both indoor and outdoor situations. When powered with a 10000 mAh battery power bank, the device worked continuously for 10 hours.



(a) (b)

Figure 5-6: Different power supplies, power bank supply(a), AC power supply(b)

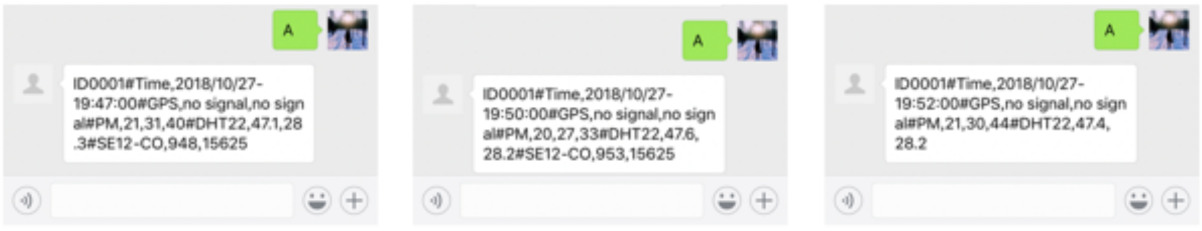
After the device collects data, then from the test of the users' view, which is WeChat app, typing any character on the WeChat app public channel to get the following feedback information. Users can obtain the required data by following the prompt information operation.


Data

Please reply area name alphabet
for air pollution data
(only a and b now,more areas
data will be online soon)

Figure 5-7: WeChat app response on mobile phones

The following Figure 5-8 (a), (b) and (c) are the data that WeChat app public channel returns after the user actively requested from the server during the test. It can be seen that the change of the communication module during the test did not affect the data collection; the addition and deletion of the sensors to the hardware device during the test did not affect the operation of the system and the data collection of other sensors.


A

ID0001#Time,2018/10/27-19:47:00#GPS,no signal,no signal#PM,21,31,40#DHT22,47.1,28.3#SE12-CO,948,15625

A

ID0001#Time,2018/10/27-19:50:00#GPS,no signal,no signal#PM,20,27,33#DHT22,47.6,28.2#SE12-CO,953,15625

A

ID0001#Time,2018/10/27-19:52:00#GPS,no signal,no signal#PM,21,30,44#DHT22,47.4,28.2

(a) (b) (c)

Figure 5-8: WeChat app response after plugging four sensors(a), changing communications(b), removing one sensor(c)

For the second part of the test, after the device left the apartment and entered to the parking lot, it lost its Wi-Fi connection and automatically switched to Zigbee data transmission. It can be seen that the Wi-Fi connection range is very limited. Then during the car's driving, the device automatically switched to LoRa transmission and then GSM transmission when the device was no longer in LoRa's range. While driving the back to the starting point, immediately the car entered the connection range of the LoRa module, the device was reconnected to LoRa communication. Two similar points were left on the same road segment and they can be used to calculate the average transmission range. The location points of each time the device automatically switched communication modules were used as a mark on the Google map. This is as shown in Figure 5-9.

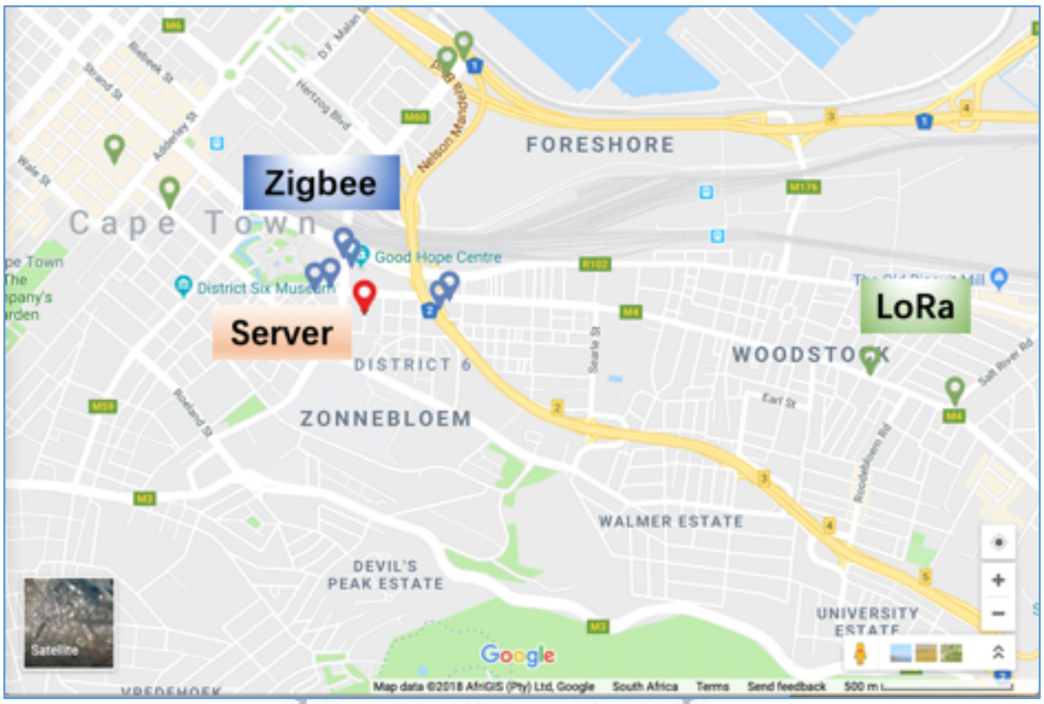


Figure 5-9: Transmission distance on the Google Map

The red point is the server, placed in the apartment. The blue points are the locations when the device switched from Zigbee to LoRa, and the green points are the locations when the device switched from LoRa to GSM. By measuring the distance between each point and the server on the map, combining this with the scale bar in the lower right corner of Google Map, the distance of each point from the server could be approximately calculated. The obtained results are shown in the Table 5-1 below

Table 5-1: The transmission distance for Zigbee and LoRa

Transmission Distance	Zigbee (m)	LoRa (m)
Location 1	300	1950
Location 2	350	2325
Location 3	200	975
Location 4	250	1075
Location 5	200	850
Location 6	225	1125
Average	254	1383

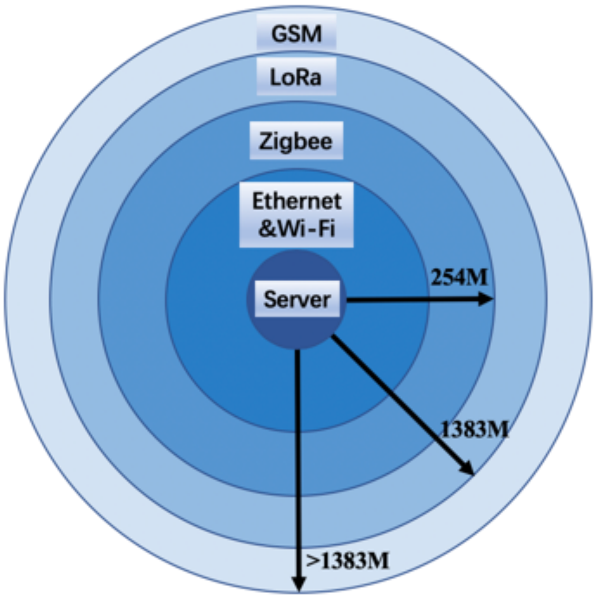


Figure 5-10: Modules transmission range

By using the average result of Table 5-10, a modules transmission range figure could be drawn as shown in Figure 5-10. The Ethernet and Wi-Fi transmission range are the shortest since Wi-Fi signal is not suitable for long distance transmission and Ethernet connection requires cables. The GSM transmission range is the longest, because it uses communication company's base stations for signal transmission, basically it can work at anywhere the mobile phone can work. For other modules, it can be seen from the Table 5-10 that in the actual test, the longest transmission distance of Zigbee was only 350 meters, while the longest transmission distance of LoRa was only 2325 meters. These measurements are far less than the theoretical value of 1,600 and 8,000 meters for Zigbee and LoRa respectively provided by the module dealers that tested in a very empty space. However, since the test server was placed on the first floor of a large building, the data signal actually needs to pass through the entire building during transmission. At the same time, the test environment was in an urban area, there are many high-rise blocks on the transmission path. All these factors can greatly reduce the transmission distance of the wireless transmission modules. If the server was placed at a higher altitude, it is believed that the transmission distance of the wireless modules will be greatly increased. However, it is not difficult to find that our system has great advantages in transmission distance and can basically cover all transmission distances to meet the needs of various scenarios.

5.1.3 Test 2: Long period outdoor test in China

The purpose of this test was to verify that the equipment was able to cope with the need to collect data for long periods of time for outdoor monitoring and to improve the prototype by comparing the data collected by the device with the data from the gas monitoring station.

Test environment and execution

The test attempted to collect air quality information data in a city called Yingtan in China. The sensors that were used on the device in this test included the temperature sensor, PM sensor, GPS sensor and CO sensor. The setup was similar to that shown in previous tests with similar communication modules. However, since the main purpose of the test was to collect data, not push it to the users, and the test area did not have a network connection, therefore during the test data were saved on the device.

There are five gas monitoring stations in the city, with names ranging from 2357A to 2361A. The device was set was about 100 meters away from the 2359A station (measurements were based on the coordinates information on the government's website).

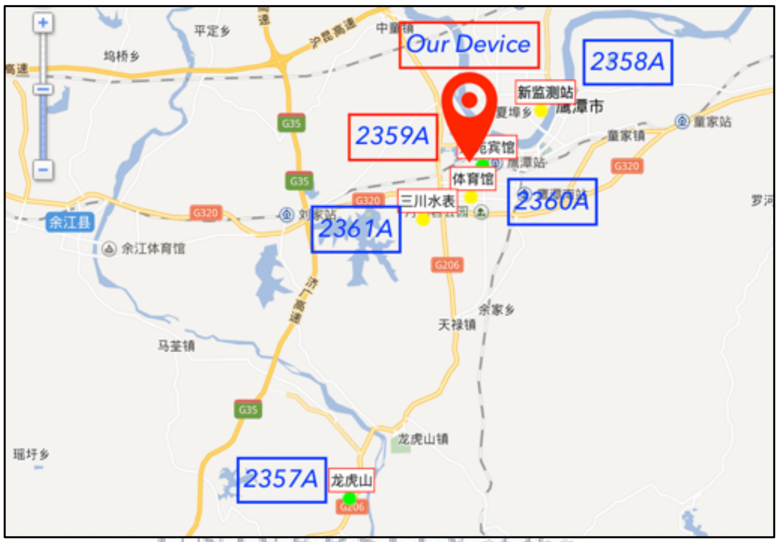


Figure 5-11: Air monitoring stations map in Yingtan, China

For the environment of the test, the device was placed on the top of a building close to the gas monitoring station 2359A and powered by a long power plug. The device was blocked up by a basket and covered by a big board to keep the device away from rain and dust. The test environment is shown in Figure 5-12.

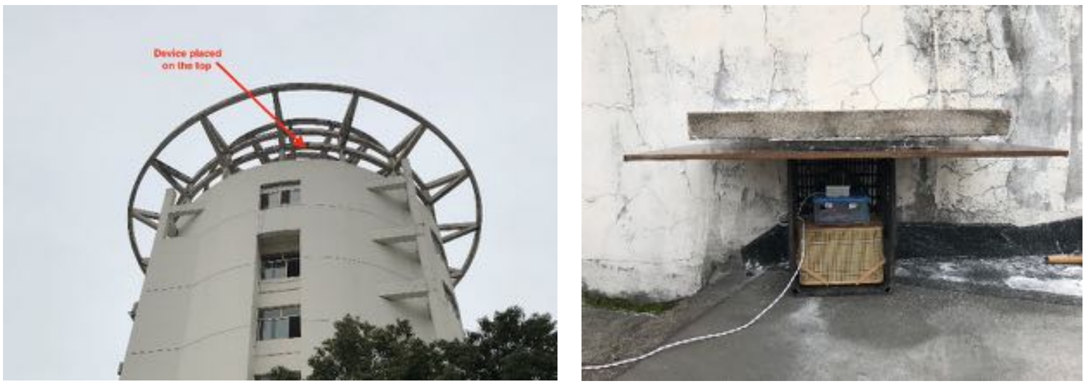
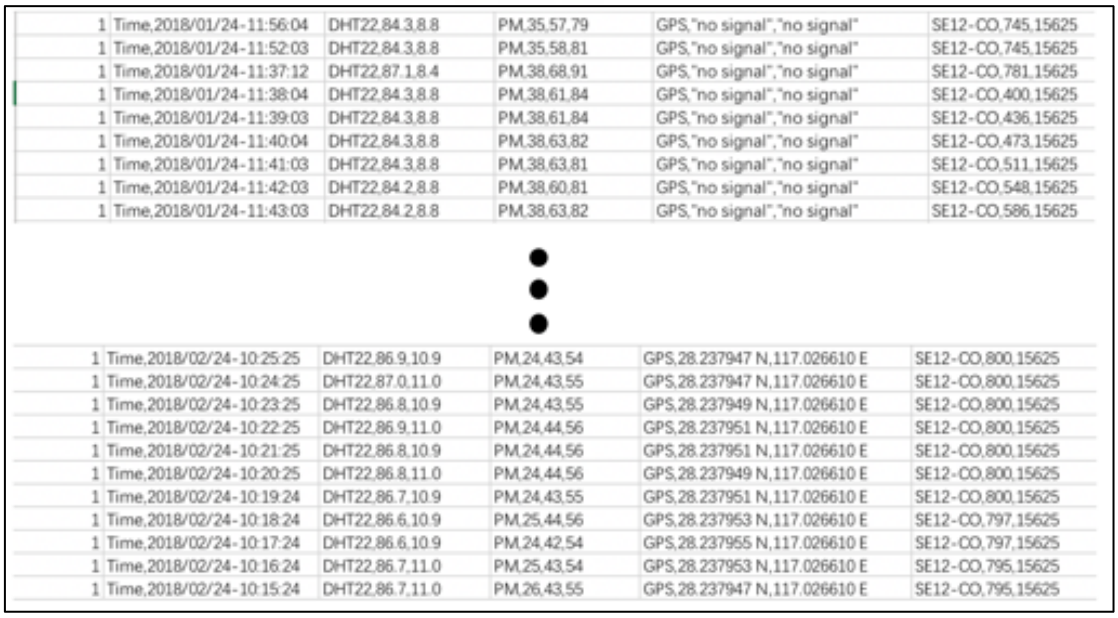


Figure 5-12:Test environment of the second experiment

Result

Figure 5-13 shows the raw data collected by using our hardware device, and the test was running from 24/01/2018 to 24/02/2018, which is about a 30 days duration. The data from left to right shows device ID, time, humidity/temperature, PM, GPS and CO information.



1	Time,2018/01/24-11:56:04	DHT22,84.3,8.8	PM,35,57,79	GPS,"no signal","no signal"	SE12-CO,745,15625
1	Time,2018/01/24-11:52:03	DHT22,84.3,8.8	PM,35,58,81	GPS,"no signal","no signal"	SE12-CO,745,15625
1	Time,2018/01/24-11:37:12	DHT22,87.1,8.4	PM,38,68,91	GPS,"no signal","no signal"	SE12-CO,781,15625
1	Time,2018/01/24-11:38:04	DHT22,84.3,8.8	PM,38,61,84	GPS,"no signal","no signal"	SE12-CO,400,15625
1	Time,2018/01/24-11:39:03	DHT22,84.3,8.8	PM,38,61,84	GPS,"no signal","no signal"	SE12-CO,436,15625
1	Time,2018/01/24-11:40:04	DHT22,84.3,8.8	PM,38,63,82	GPS,"no signal","no signal"	SE12-CO,473,15625
1	Time,2018/01/24-11:41:03	DHT22,84.3,8.8	PM,38,63,81	GPS,"no signal","no signal"	SE12-CO,511,15625
1	Time,2018/01/24-11:42:03	DHT22,84.2,8.8	PM,38,60,81	GPS,"no signal","no signal"	SE12-CO,548,15625
1	Time,2018/01/24-11:43:03	DHT22,84.2,8.8	PM,38,63,82	GPS,"no signal","no signal"	SE12-CO,586,15625
			● ● ●		
1	Time,2018/02/24-10:25:25	DHT22,86.9,10.9	PM,24,43,54	GPS,28.237947 N,117.026610 E	SE12-CO,800,15625
1	Time,2018/02/24-10:24:25	DHT22,87.0,11.0	PM,24,43,55	GPS,28.237947 N,117.026610 E	SE12-CO,800,15625
1	Time,2018/02/24-10:23:25	DHT22,86.8,10.9	PM,24,43,55	GPS,28.237949 N,117.026610 E	SE12-CO,800,15625
1	Time,2018/02/24-10:22:25	DHT22,86.9,11.0	PM,24,44,56	GPS,28.237951 N,117.026610 E	SE12-CO,800,15625
1	Time,2018/02/24-10:21:25	DHT22,86.8,10.9	PM,24,44,56	GPS,28.237951 N,117.026610 E	SE12-CO,800,15625
1	Time,2018/02/24-10:20:25	DHT22,86.8,11.0	PM,24,44,56	GPS,28.237949 N,117.026610 E	SE12-CO,800,15625
1	Time,2018/02/24-10:19:24	DHT22,86.7,10.9	PM,24,43,55	GPS,28.237951 N,117.026610 E	SE12-CO,800,15625
1	Time,2018/02/24-10:18:24	DHT22,86.6,10.9	PM,25,44,56	GPS,28.237953 N,117.026610 E	SE12-CO,797,15625
1	Time,2018/02/24-10:17:24	DHT22,86.6,10.9	PM,24,42,54	GPS,28.237955 N,117.026610 E	SE12-CO,797,15625
1	Time,2018/02/24-10:16:24	DHT22,86.7,11.0	PM,25,43,54	GPS,28.237953 N,117.026610 E	SE12-CO,795,15625
1	Time,2018/02/24-10:15:24	DHT22,86.7,11.0	PM,26,43,55	GPS,28.237947 N,117.026610 E	SE12-CO,795,15625

Figure 5-13: Sample data collected of the second test

It can be seen that the frequency of data collection of the device was in minutes. After converting the data into the hourly average and plotting the PM2.5/PM10 and CO data chronologically onto the XY axis, the result shown in Figure 5-14 was obtained. The x-axis is time and the y-axis is the value of gas pollution.

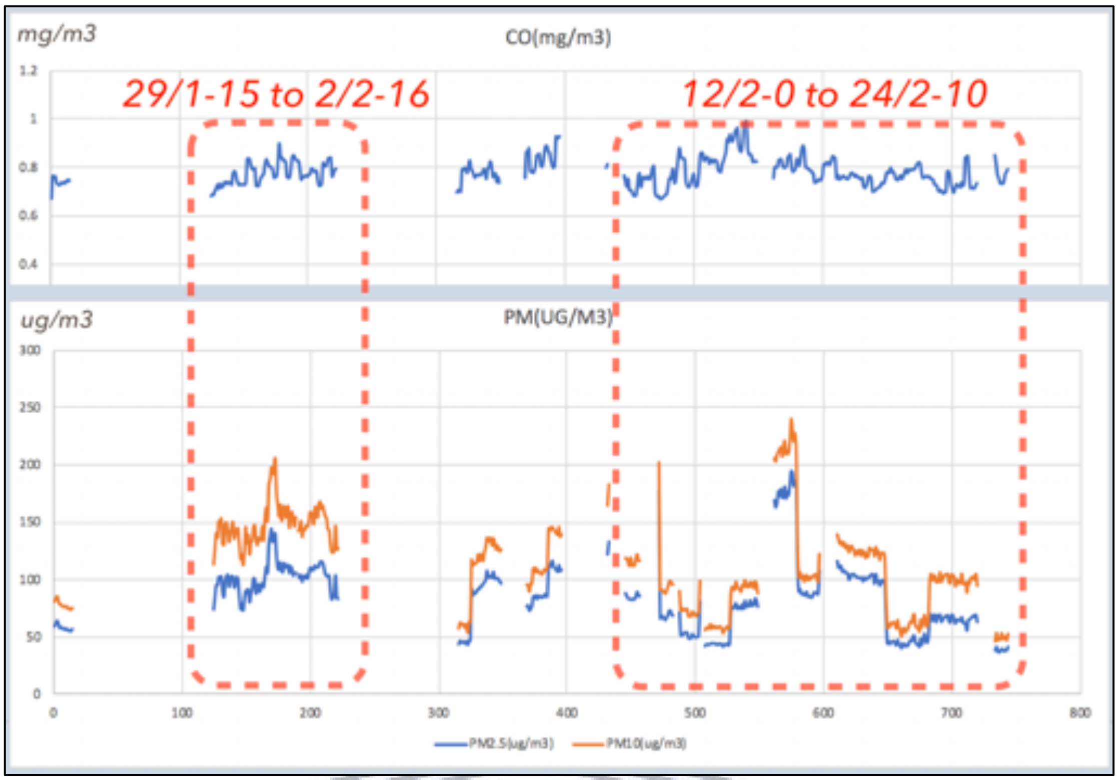


Figure 5-14: CO and PM results of test 2

It can be seen from above Figure 5-14 that the device actually stopped few times during the test, thus the data is not continued. The reason may be because of the power connection line was not stable or the device algorithm was not well developed properly. From the figure, it can be concluded that the two consecutive parts are from 15:00 clock of 29/02/2018 to 16:00 clock of 2/2/2018, and 0:00 clock of 12/02/2018 to 10:00 clock of 24/02/2018. Therefore, by combining the two consecutive part we have about 20 days of continuous data to compare with the corresponding data in the same period from the gas monitoring station. The comparison is shown in Figure 5-15 below.

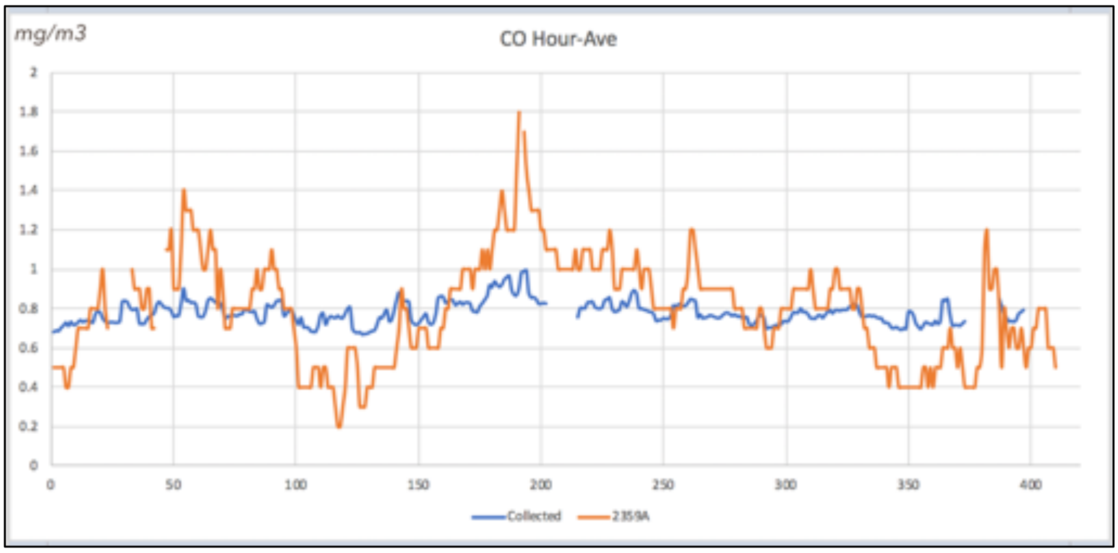


Figure 5-15: CO result comparison in hour average with the station in China

The blue line is the data collected by our device, and the red line is the data from gas station 2359A. The only thing can be told from the Figure 5-15 is that our data is in the range of the station's data. In order to further compare the data from our device and the data from the gas station, the two sets data were converted from hourly average to daily average (24 hours). These results are shown in Figure 5-16 to 5-18. Similarly, the blue line is the data from our device and the red line the data from the station as shown below.

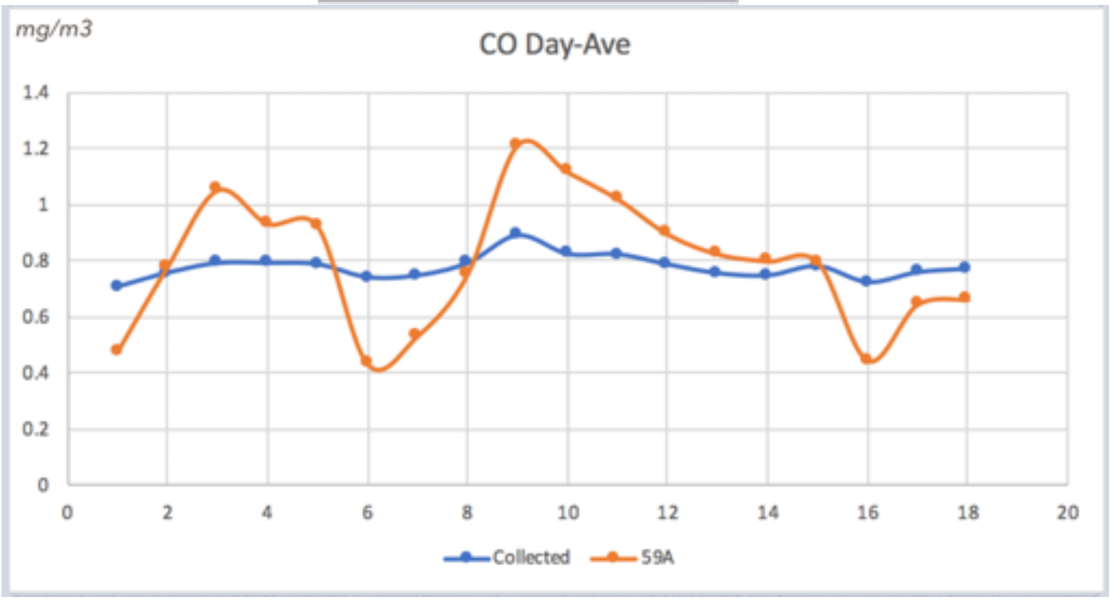


Figure 5-16: CO result comparison in day average with the station in China

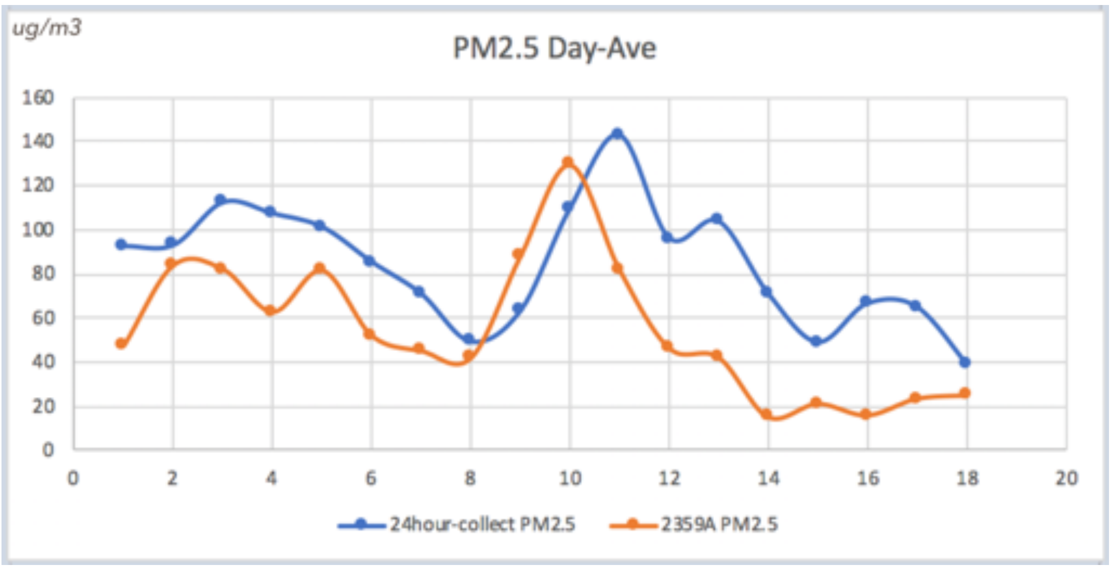


Figure 5-17: PM2.5 result comparison in day average with the station in China

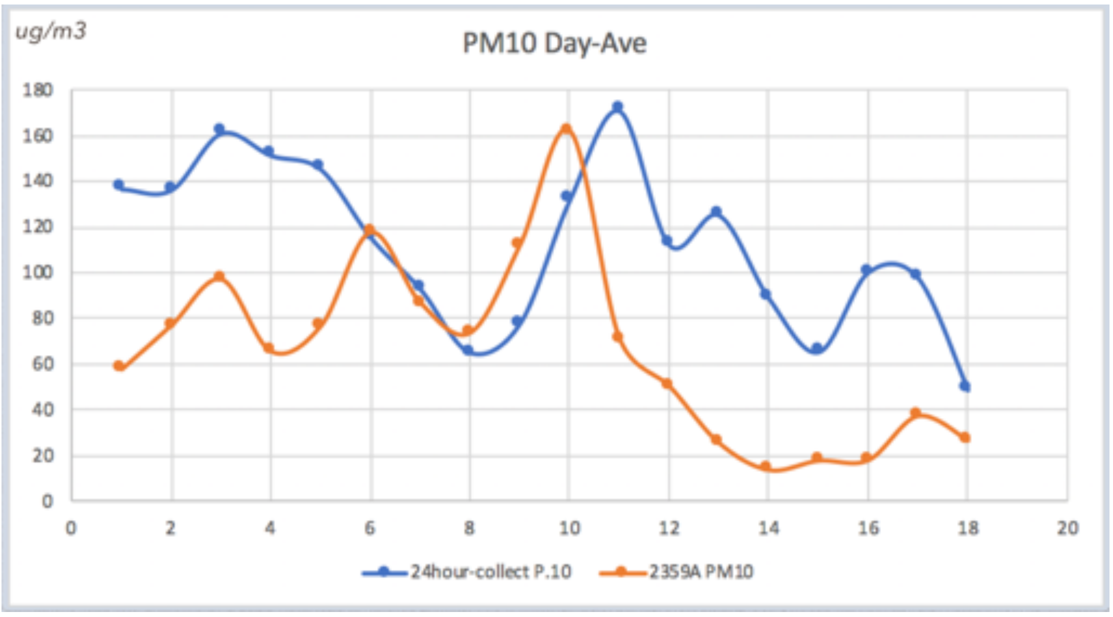


Figure 5-18: PM10 result comparison in day average with the station in China

From the above figures some similarities are observed in the data trend, but at the same time there is a large difference. For example, for the data of CO from point 1 to point 3 both values increased, and from point 6 to 9, both values are going up and reached the highest value at point 9, then decrease till point 14. Both data had a trough at point 16. Similar trends can be observed for the PM data. Though the shape of our data is close to that of the station, each part has significantly large difference. Therefore, the correlation coefficient is introduced here to judge the similarity between the two sets of data with comparison results shown on Table 5-2 and 5-3.

Table 5-2: The correlation coefficient between the device data and station data

CC	CO	PM2.5	PM10
Hour Ave	0.7224916	0.49330678	0.16192435
Day Ave	0.87848581	0.63049177	0.34770864

Table 5-3: Meaning of correlation coefficient

CC	0-0.2	0.2-0.4	0.4-0.6	0.6-0.8	0.8-1
Correlation	Extremely Weak	Weak	Moderate	Strong	Extremely Strong

Usually the correlation coefficient (a value between -1 and +1) tells how strongly two variables are related to each other. If the value goes to 1, it means the two data are very close or similar to each other. From the day-average result, it can be seen that our CO and $PM_{2.5}$ data are strongly related to the station's data. For the PM10 data, there was very weak correlation to the station's data. However, it is easy to see that the PM data has a more obvious fluctuation trend than the CO data to the station; and the PM data has a high degree of similarity in the peak part, but in different time periods. Our data was thus shifted one day left to do another compression with the station data again and the obtained results are shown in Figure 5-19 below.

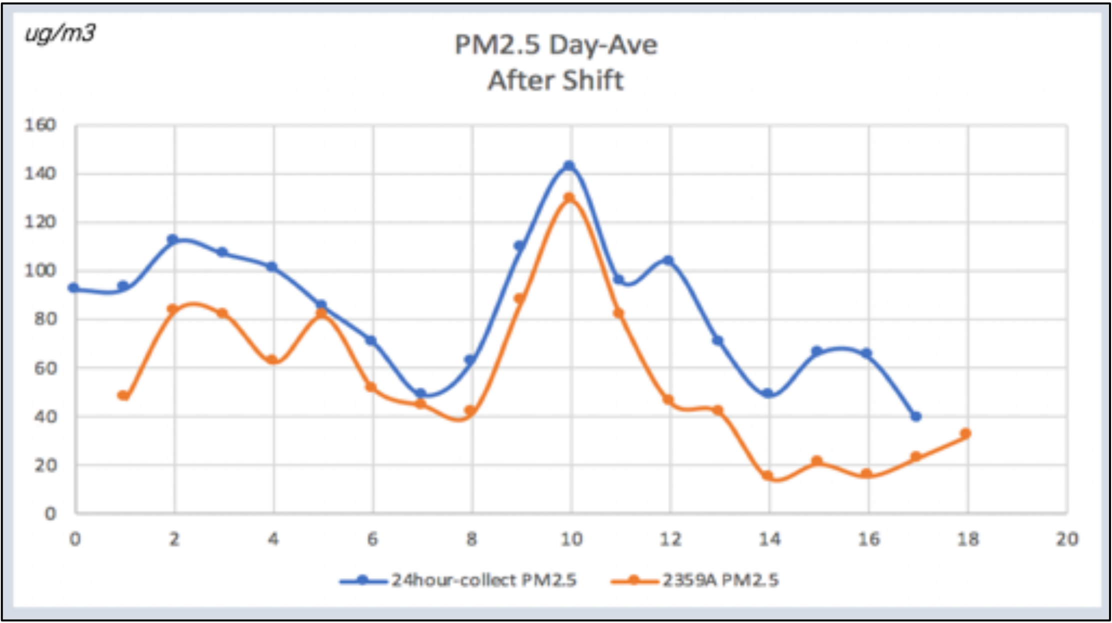


Figure 5-19: PM2.5 result comparison in day average after shift

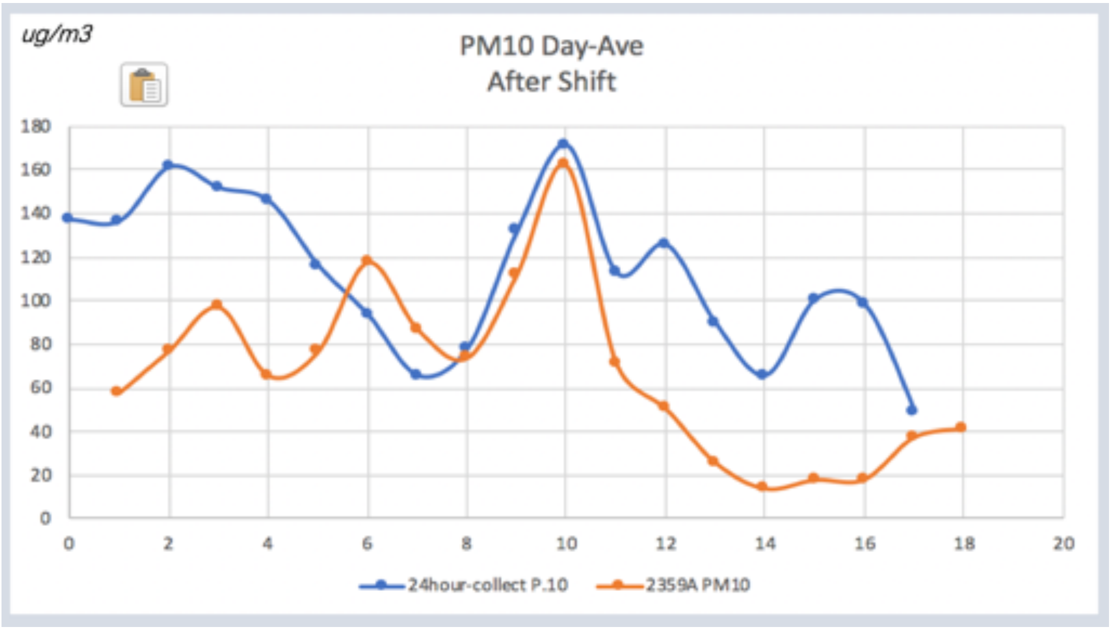


Figure 5-20: PM10 result comparison in day average after shift

Table 5-4: The correlation coefficient compression after shift

CC	PM2.5	PM10
Day Ave	0.86453037	0.53070565

It is not difficult to see that the similarity between the two sets of data has been greatly improved. Both sets of data reached the peak value at the point 10, and the overall fluctuation trends are much closer. The correlation coefficient of day-average of $PM_{2.5}$ has improved from 0.63049177 to 0.86453037 and the PM10 has improved from 0.34770864 to 0.53070565.

Analysis

For temperature data, the station did not provide data on the website, so there was no way to compare the temperature data. But for GPS data, the coordinates of the gas monitoring station are 28.2380 N, 117.0266 E, it is accessible from the official website (Quot, 2019). The coordinates collected by our device were 28.237947 N, 117.026610 S, these two values are very close to each other. It can be said that the data collected by the GPS sensor is very ideal. According to the correlation coefficient of CO, and $PM_{2.5}$, the result could be said that it is a good, but the result of PM10 was less than ideal. A possible explanation for this might be that the device was not deployed in complete outdoor environment for the test. This could also explain why the data collected by the device compared to the station data was quite smooth and had less fluctuations. Though the room used for test did not have a front wall, but it had a roof, side walls and back wall, the air circulation in the room was much slower than the outside. This might have affected the accuracy of the data collected, and large amount of the air that was detecting by the device could not be the real-time air same as the outdoor environment. As in the test, the device was covered by a big board to prevent the influence on the result from rain and smoke, as well as dust. A picture of the fan installed in the device after the 30-day test is shown in Figure 3-21, with lot of dust accumulated on it. This is another important factor that could have affected the test.

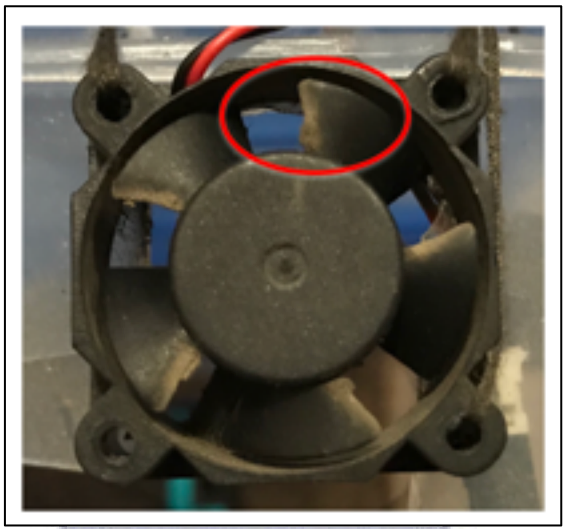


Figure 5-21: The fan on the device after the test

In addition, the device stopped a few times during the test because of the unstable power plug. Subsequent tests would strive to address these shortcomings to ensure a more accurate comparison

5.1.4 Test 3: Monitoring test with ARA N-FRM sampler

Since the results of the previous experiment were quite different from the gas stations in China, the system was improved by modifying aspects of the code. This experiment was then re-tested in communities in South African. The purpose this experiment was to verify the accuracy of the device in comparison with professional equipment.

Test environment and execution

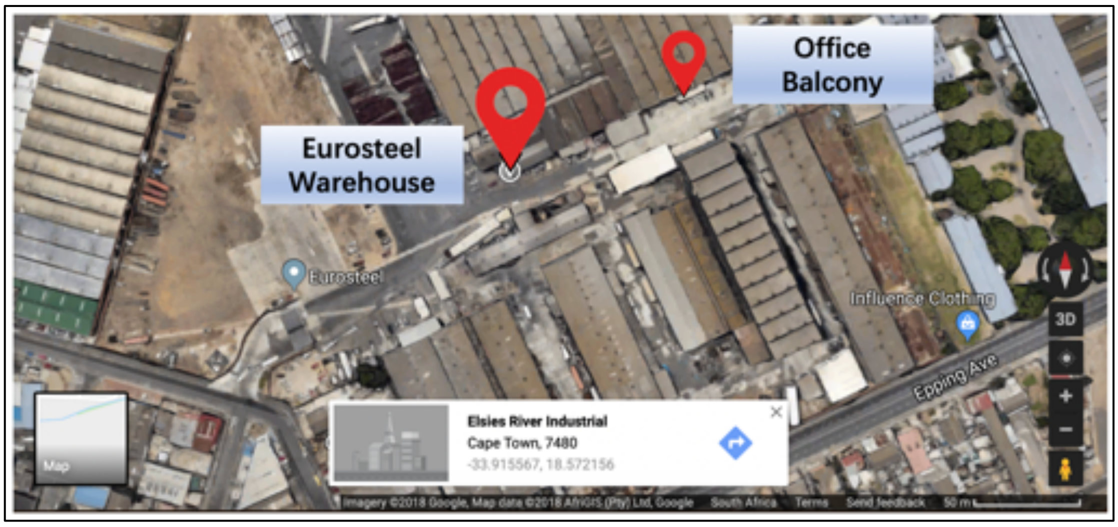


Figure 5-22: Google map of Eurosteel (Google, 2018)

The entire experiment conducted in two areas of Eurosteel, (a manufacturing company in Cape Town). The first was to monitor air conditions in indoor environments; the device was deployed within the warehouse of Eurosteel. The second part was to monitor outdoor air conditions and the equipment was placed on the balcony of an office. In the first part of the experiment, the hardware used was the same prototype as the previous experiment in test 2. The functions within the prototype were temperature, humidity, PM, GPS and CO, sensing. The system was compared with a portable air sampler - ARA N-FRM Sampler made by ARA Instruments. This device has the advantages of a simple and convenient installation process and works with both external and battery power supply. Additional accessories could also can be installed to collect $PM_{2.5}$, PM10 and wind speed data.

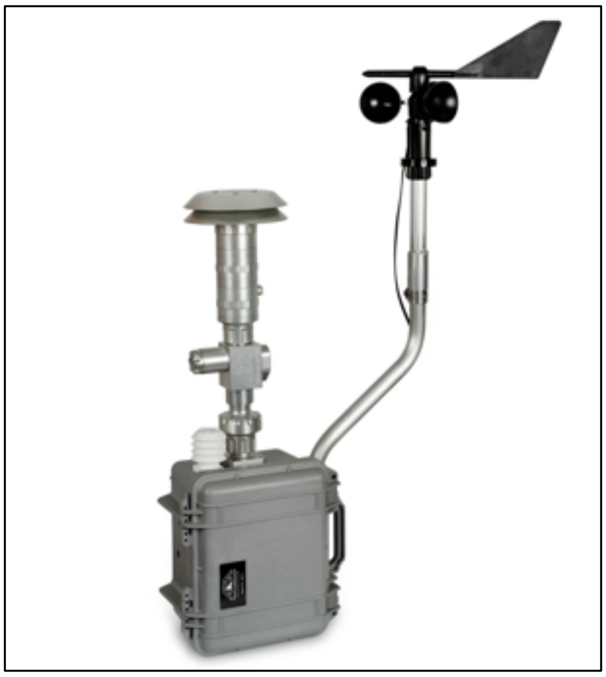
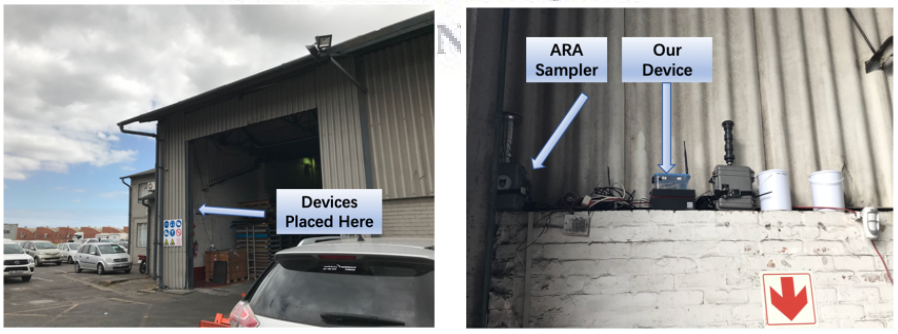


Figure 5-23: ARA N-FRM sampler(ARA, 2016)

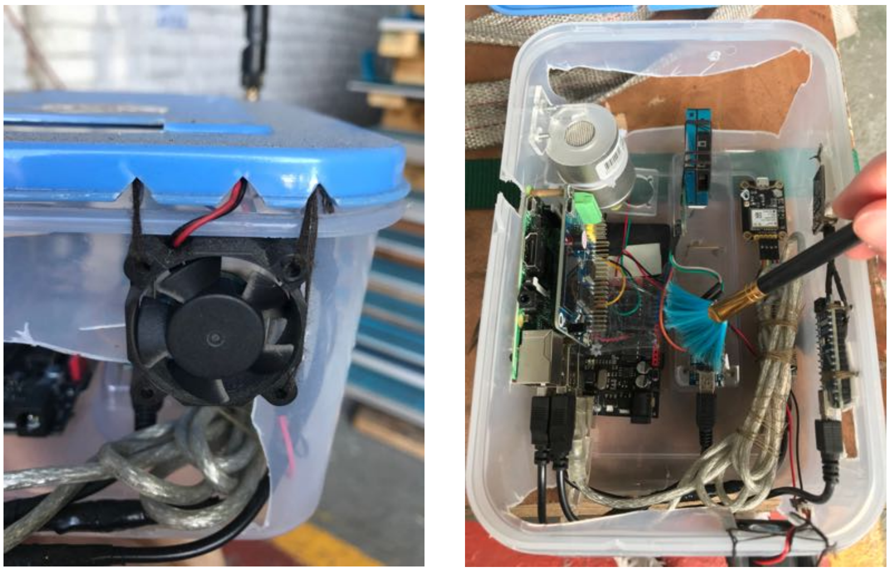
The ARA N-FRM Sampler was provided by Johanna, a post-doctor from the Chemistry Department in UCT. Using the ARA N-FRM Sampler availed us the opportunity to compare our prototype with a professional equipment. In the first part of the experiment, the two equipment were placed in a fence near the gate of the warehouse. Since the air monitoring was in an indoors environment, so there was no need to consider the effects of rain.



(a) (b)

Figure 5-24: Test environment of the warehouse outside(a), inside(b)

Considering the fan dusting problem that occurred during the long-term test in the previous experiment, that affected the accuracy of the test data. To avoid this, the fan was cleaned with a brush on a weekly basis reduce the impact of accumulation dust on data accuracy.



(a) (b)

Figure 5-25: Device cleaning fan(a), inside(b)

At the beginning of the experiment, the data obtained from our device was compared with the data from the ARA equipment and there were noticeable differences between the two data. The suspected cause was probably the distance between the two equipment. The two equipment were moved close to each other and our prototype raised to reduce the impact of dust. This is as shown in Figure 5-26.

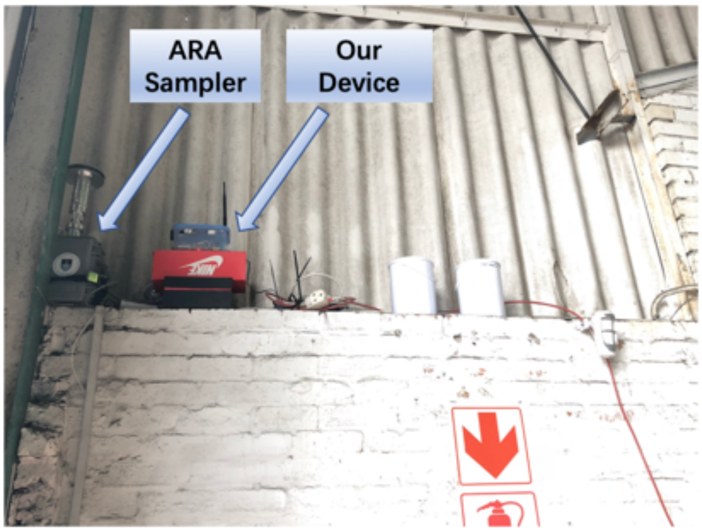


Figure 5-26: After moving the two devices closer

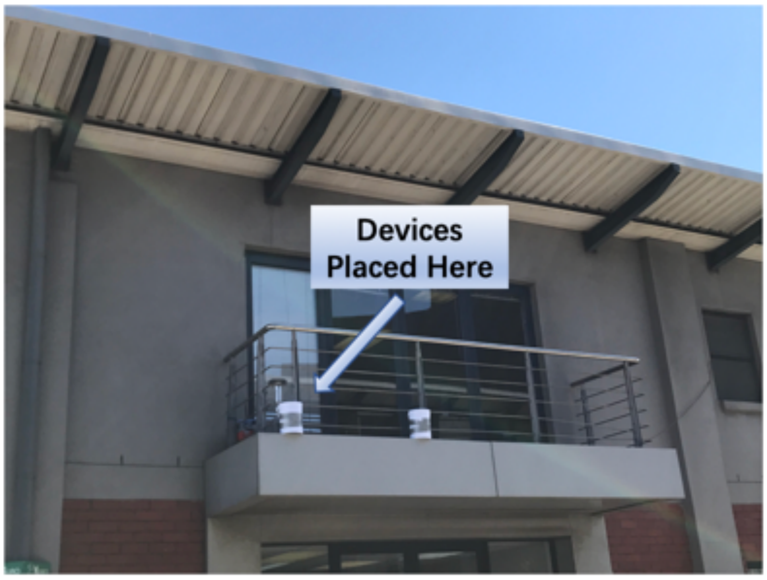
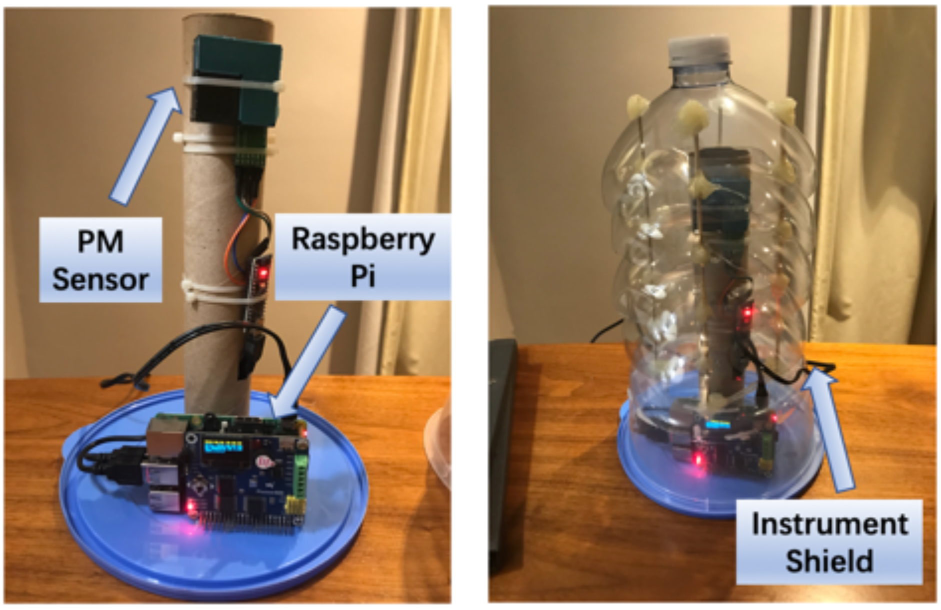


Figure 5-27: Outdoor test environment in the balcony

The second part of the experiment was carried out for outdoor monitoring, in a covered balcony in the Eurosteel industrial zone. The same power cord was used to power both equipment. Being a completely outdoor test, there was the possibility of rain. To cater for this, a louvered design used in the "ARA air sampler and Goleta air monitoring project" in (Zymbit, 2016) was adapted. A simple instrument shield was produced for our device. Ideally, the inner and outer parts of the louvered shield should be white, however due to certain limitation a water bottle was used as a raw material for building the instrument shield. This helped reduce the influence of strong winds and rains on the equipment and the sensors. We also ensured that the prototype was well ventilated in order for it to efficiently sense the changes in ambient air temperature and humidity. However, due to limited space of the louvered shield built for our prototype, we could only connect the PM and temperature sensors; and compared these with the ARA equipment.



(a) (b)

Figure 5-28: Rebuild the device with shield, device part(a), shield part(b)

The newly designed equipment was then placed on the balcony alongside the ARA equipment for a week to monitor the air conditions.

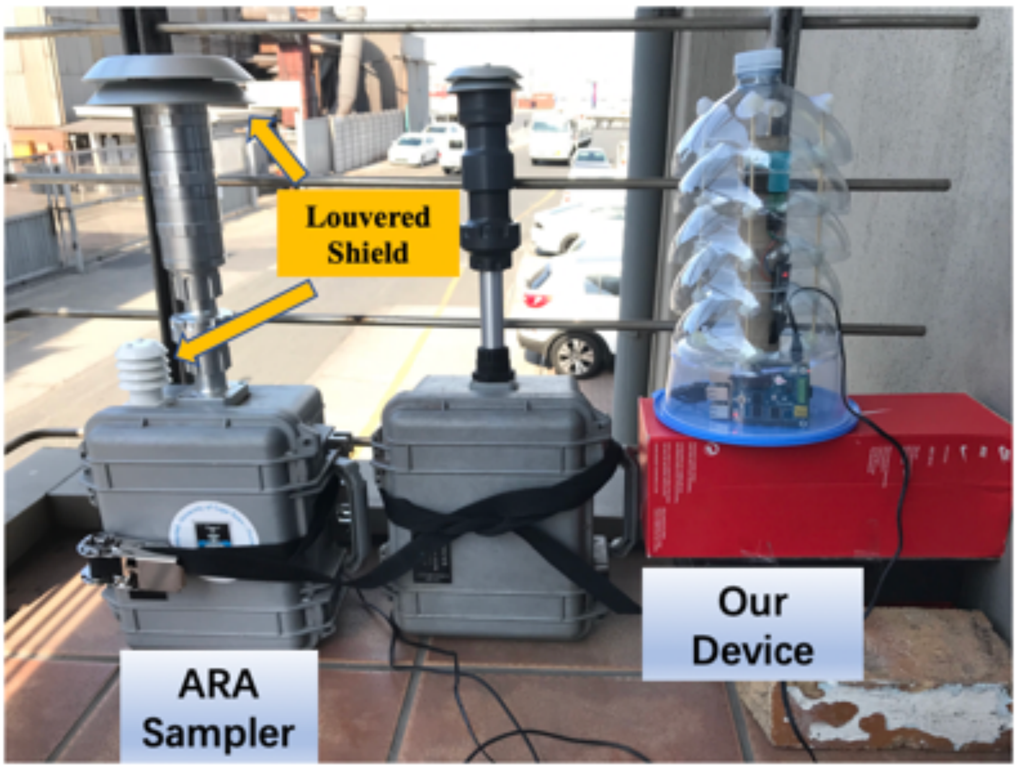
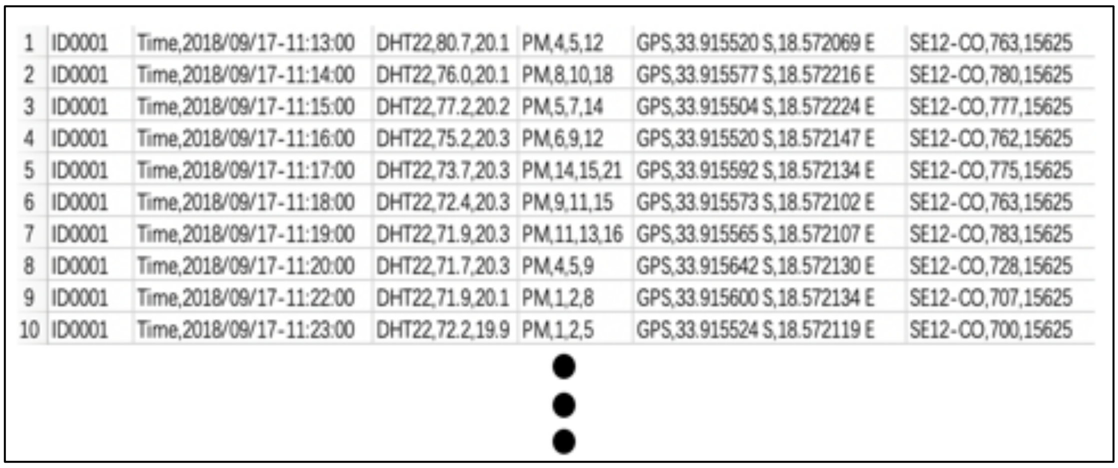


Figure 5-29: Outdoor test on the balcony with ARA sampler

Result

The Figure 5-30 shows some example data collected by our device, the data from left to right shows device ID, time, humidity/temperature, PM, GPS and CO information. Presented in the same format as in previous experiments.

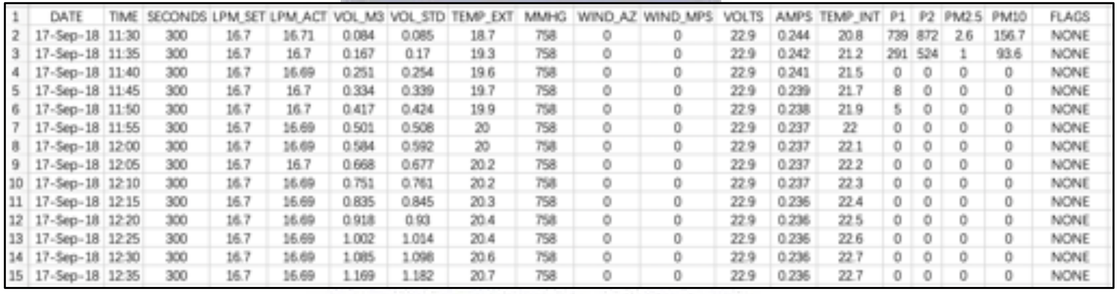


1	ID0001	Time,2018/09/17-11:13:00	DHT22,80.7,20.1	PM,4,5,12	GPS,33.915520 S,18.572069 E	SE12-CO,763,15625
2	ID0001	Time,2018/09/17-11:14:00	DHT22,76.0,20.1	PM,8,10,18	GPS,33.915577 S,18.572216 E	SE12-CO,780,15625
3	ID0001	Time,2018/09/17-11:15:00	DHT22,77.2,20.2	PM,5,7,14	GPS,33.915504 S,18.572224 E	SE12-CO,777,15625
4	ID0001	Time,2018/09/17-11:16:00	DHT22,75.2,20.3	PM,6,9,12	GPS,33.915520 S,18.572147 E	SE12-CO,762,15625
5	ID0001	Time,2018/09/17-11:17:00	DHT22,73.7,20.3	PM,14,15,21	GPS,33.915592 S,18.572134 E	SE12-CO,775,15625
6	ID0001	Time,2018/09/17-11:18:00	DHT22,72.4,20.3	PM,9,11,15	GPS,33.915573 S,18.572102 E	SE12-CO,763,15625
7	ID0001	Time,2018/09/17-11:19:00	DHT22,71.9,20.3	PM,11,13,16	GPS,33.915565 S,18.572107 E	SE12-CO,783,15625
8	ID0001	Time,2018/09/17-11:20:00	DHT22,71.7,20.3	PM,4,5,9	GPS,33.915642 S,18.572130 E	SE12-CO,728,15625
9	ID0001	Time,2018/09/17-11:22:00	DHT22,71.9,20.1	PM,1,2,8	GPS,33.915600 S,18.572134 E	SE12-CO,707,15625
10	ID0001	Time,2018/09/17-11:23:00	DHT22,72.2,19.9	PM,1,2,5	GPS,33.915524 S,18.572119 E	SE12-CO,700,15625

●
●
●

Figure 5-30: Example data collected by our device

The Figure 5-31shows some example data collected by the ARA equipment.



1	DATE	TIME	SECONDS	LPM_SET	LPM_ACT	VOL_M3	VOL_STD	TEMP_EXT	MMHG	WIND_AZ	WIND_MPS	VOLTS	AMPS	TEMP_INT	P1	P2	PM2.5	PM10	FLAGS
2	17-Sep-18	11:30	300	16.7	16.71	0.084	0.085	18.7	758	0	0	22.9	0.244	20.8	739	872	2.6	156.7	NONE
3	17-Sep-18	11:35	300	16.7	16.7	0.167	0.17	19.3	758	0	0	22.9	0.242	21.2	291	524	1	93.6	NONE
4	17-Sep-18	11:40	300	16.7	16.69	0.251	0.254	19.6	758	0	0	22.9	0.241	21.5	0	0	0	0	NONE
5	17-Sep-18	11:45	300	16.7	16.7	0.334	0.339	19.7	758	0	0	22.9	0.239	21.7	8	0	0	0	NONE
6	17-Sep-18	11:50	300	16.7	16.7	0.417	0.424	19.9	758	0	0	22.9	0.238	21.9	5	0	0	0	NONE
7	17-Sep-18	11:55	300	16.7	16.69	0.501	0.508	20	758	0	0	22.9	0.237	22	0	0	0	0	NONE
8	17-Sep-18	12:00	300	16.7	16.69	0.584	0.592	20	758	0	0	22.9	0.237	22.1	0	0	0	0	NONE
9	17-Sep-18	12:05	300	16.7	16.7	0.668	0.677	20.2	758	0	0	22.9	0.237	22.2	0	0	0	0	NONE
10	17-Sep-18	12:10	300	16.7	16.69	0.751	0.761	20.2	758	0	0	22.9	0.237	22.3	0	0	0	0	NONE
11	17-Sep-18	12:15	300	16.7	16.69	0.835	0.845	20.3	758	0	0	22.9	0.236	22.4	0	0	0	0	NONE
12	17-Sep-18	12:20	300	16.7	16.69	0.918	0.93	20.4	758	0	0	22.9	0.236	22.5	0	0	0	0	NONE
13	17-Sep-18	12:25	300	16.7	16.69	1.002	1.014	20.4	758	0	0	22.9	0.236	22.6	0	0	0	0	NONE
14	17-Sep-18	12:30	300	16.7	16.69	1.085	1.098	20.6	758	0	0	22.9	0.236	22.7	0	0	0	0	NONE
15	17-Sep-18	12:35	300	16.7	16.69	1.169	1.182	20.7	758	0	0	22.9	0.236	22.7	0	0	0	0	NONE

Figure 5-31: Example data collected by ARA

The data collected by the ARA equipment includes time, temperature, operating voltage, $PM_{2.5}$ and PM10 data. However, since the ARA N-FRM Sampler mainly targets PM for air pollution, this experiment will mainly focus on the PM data. The device also only had the PM10 filter installed, thus PM2.5 data we not compared. In essence, this experiment focused on the comparison between the temperature and PM10 data of both devices. It can be seen that the frequency of data collection of the device was in minutes and the ARA was in 5 minutes. After converting the data into the hourly average and daily average and plotted the monitoring data chronologically onto the XY axis to obtain the graph in Figure 5-30. The x-axis is time and the y-axis are the value of the data.

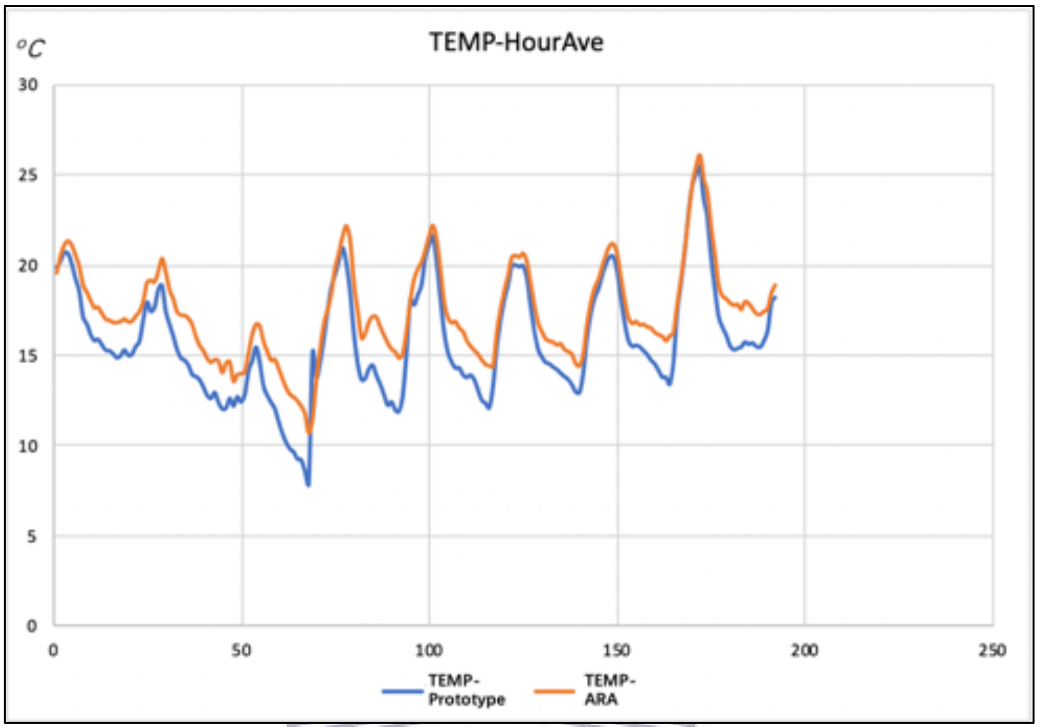


Figure 5-32: Temperature result comparison in hour average with ARA sampler (1)

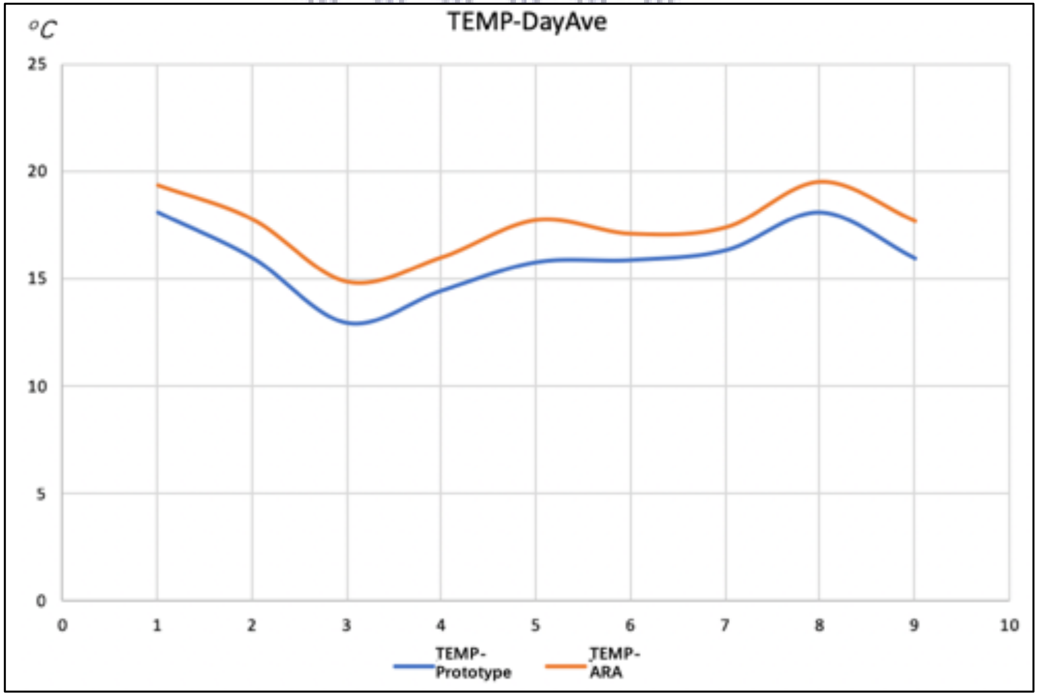


Figure 5-33: Temperature result comparison in day average with ARA sampler (1)

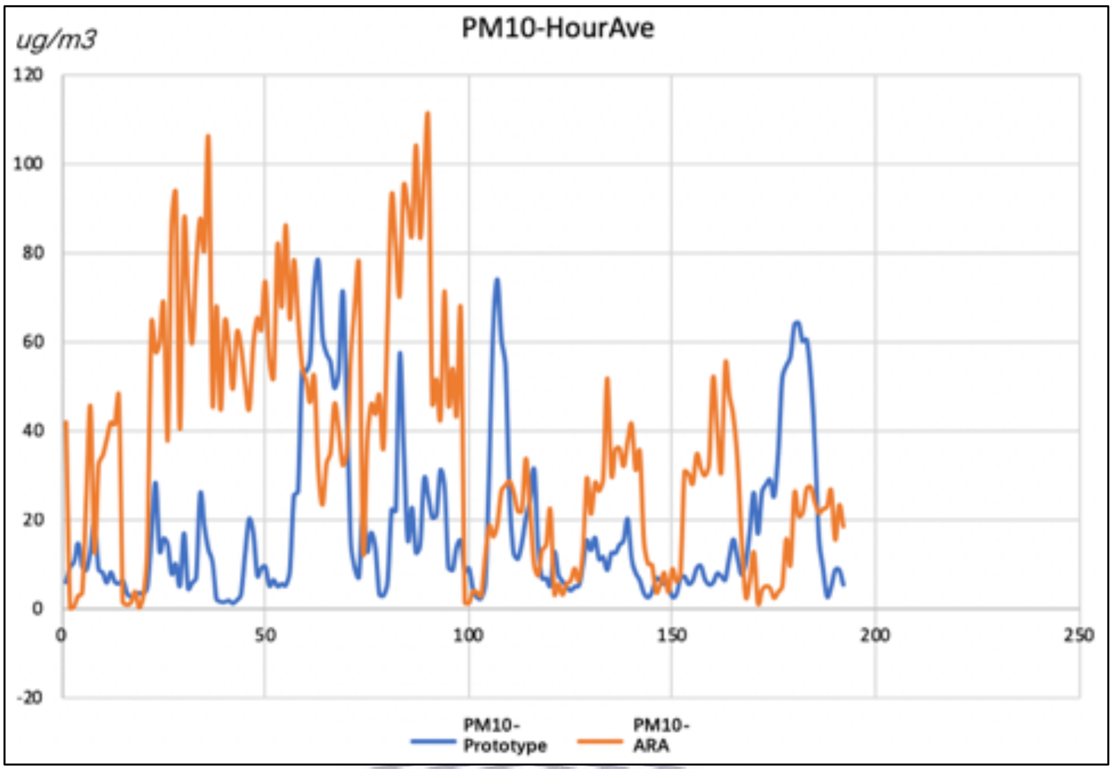


Figure 5-34: PM10 result comparison in hour average with ARA sampler (1)

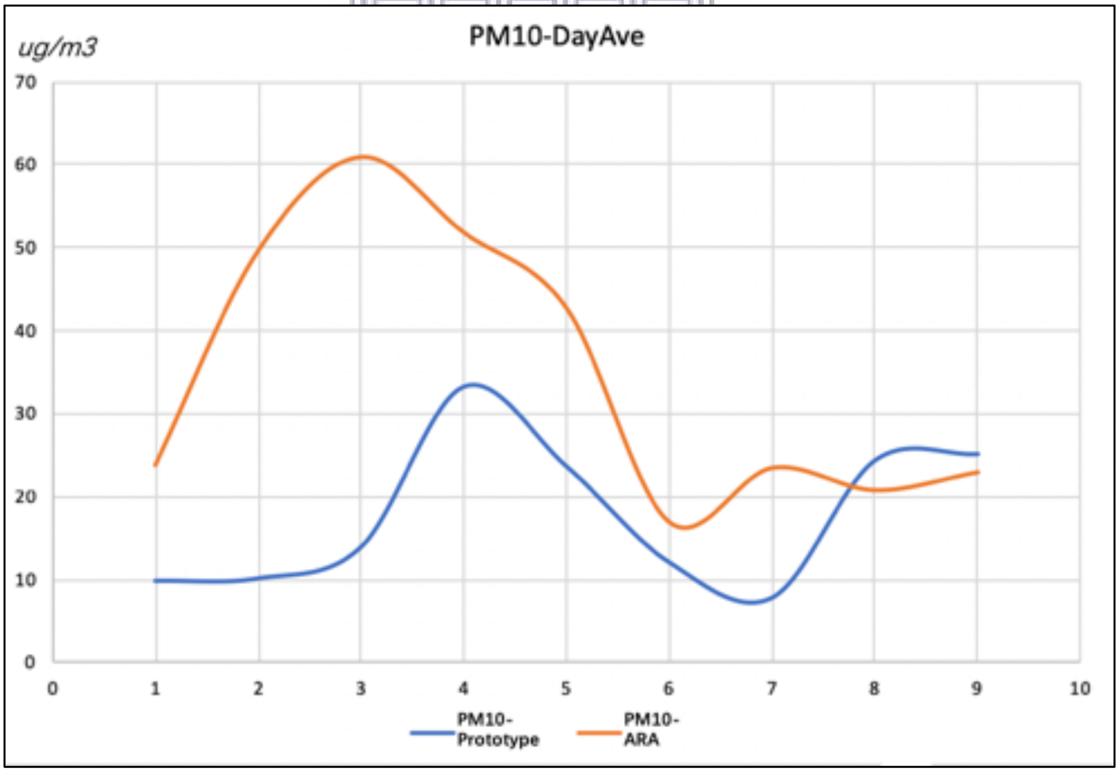


Figure 5-35: PM10 result comparison in day average with ARA sampler (1)

The data shown above was from the first part of the experiment. The experiment process was run for 8 days, from 17/09/2018 to 25/09/2018 and the two devices were far apart at the beginning of the experiment. The blue line is the data from our device and the red line is the data of the ARA equipment. However, it is not difficult to see that there is a good similarity in the temperature data of the two equipment, whether it is the temperature change trend or the actual temperature value, but the data of PM10 are quite different. Therefore, the correlation coefficient is also introduced here to judge the similarity of data between the equipment. The comparison results are shown in the following table.

Table 5-5: The correlation coefficient between our device data and ARA data (1)

CC	TEMP	PM10
Hour Ave	0.9643591	0.0103854
Day Ave	0.98195839	0.18580183

It can be seen that the result of temperature data is very good, the correlation coefficient is very close to 1, indicating that the two data are very close to each other, but the result of PM10 data is not good. Therefore, in the following experiments, the two devices were moved as close as possible to reduce the error caused by different environments. After converting the data into the hourly average and daily average and plotted the monitoring data chronologically onto the XY axis, the graph shown in Figure 5-34 was obtained. Time is on the x-axis, while time is on y-axis.

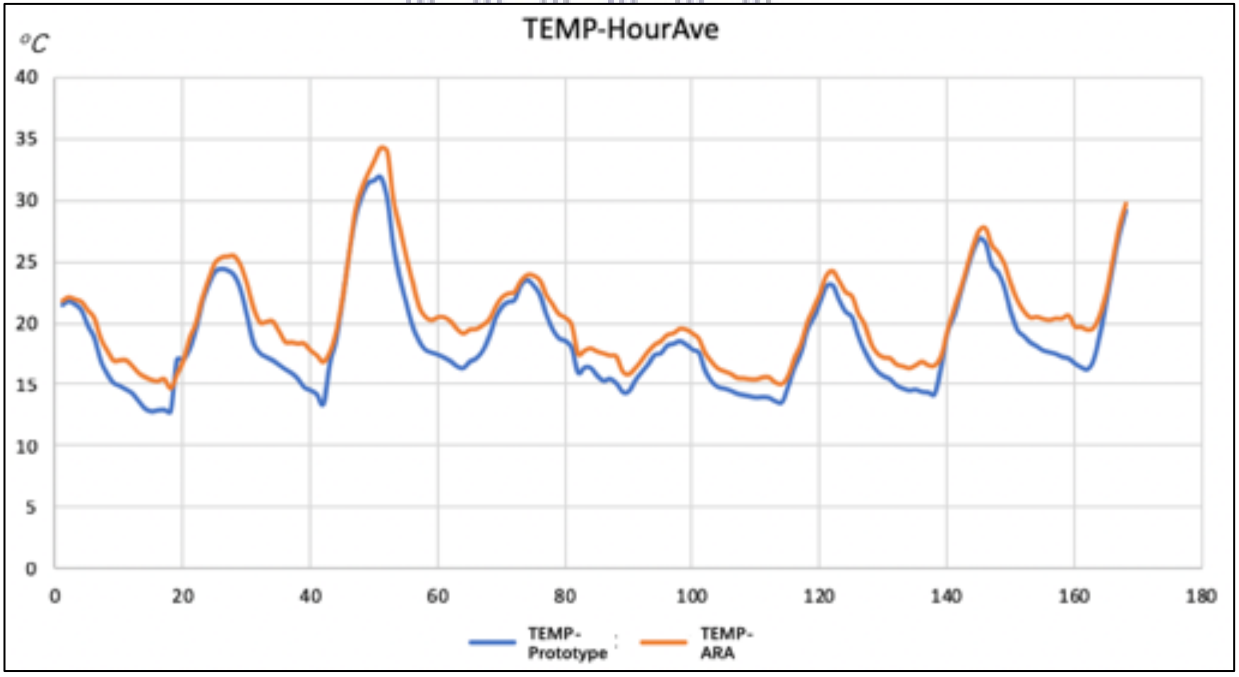


Figure 5-36: Temperature result comparison in hour average with ARA sampler (2)

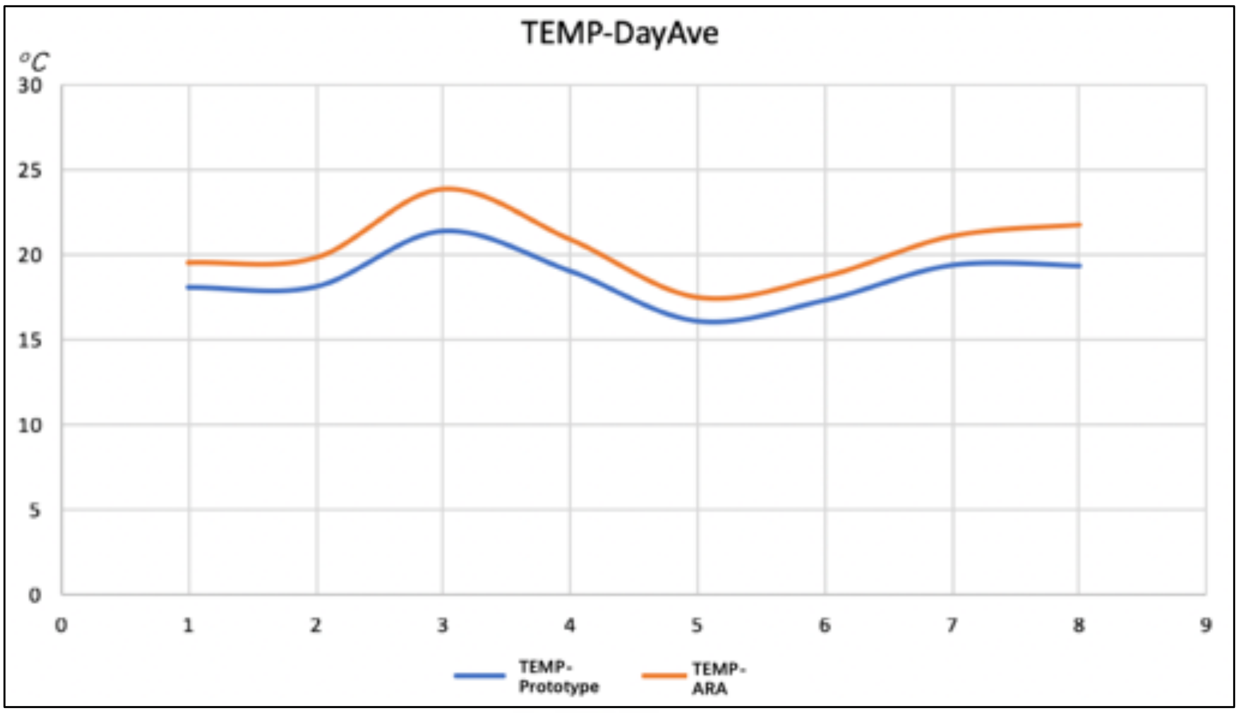


Figure 5-37: Temperature result comparison in day average with ARA sampler (2)

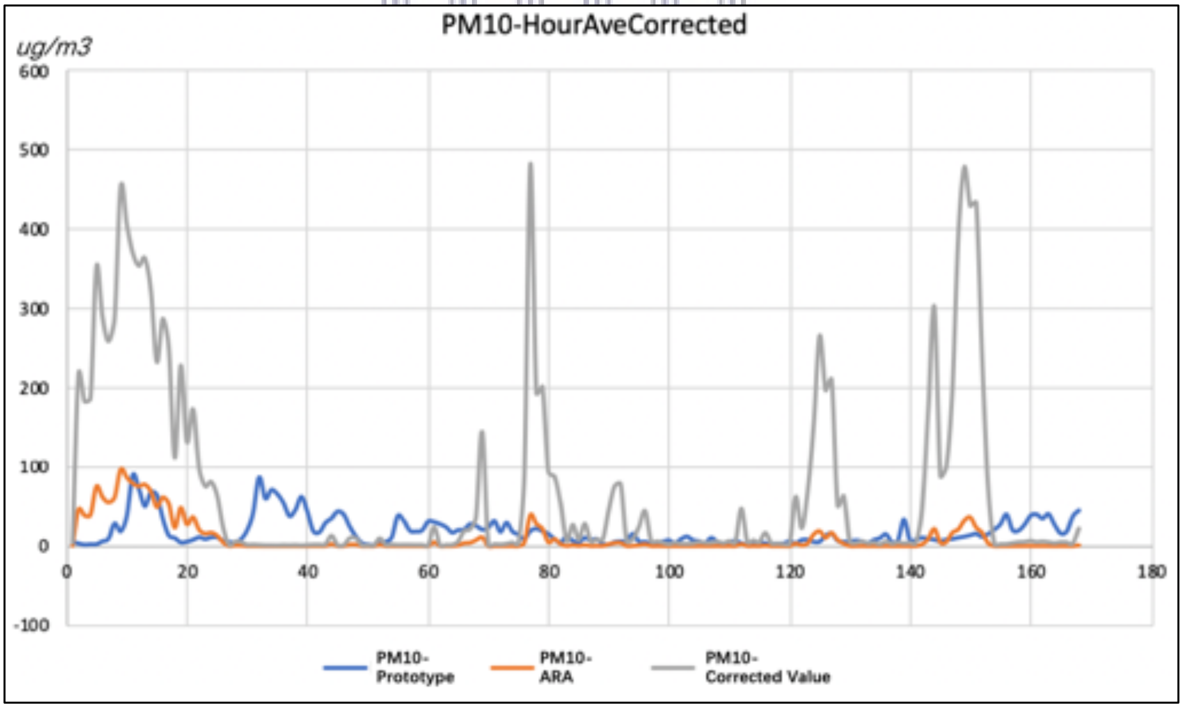


Figure 5-38: PM10 result comparison in hour average with ARA sampler (2)

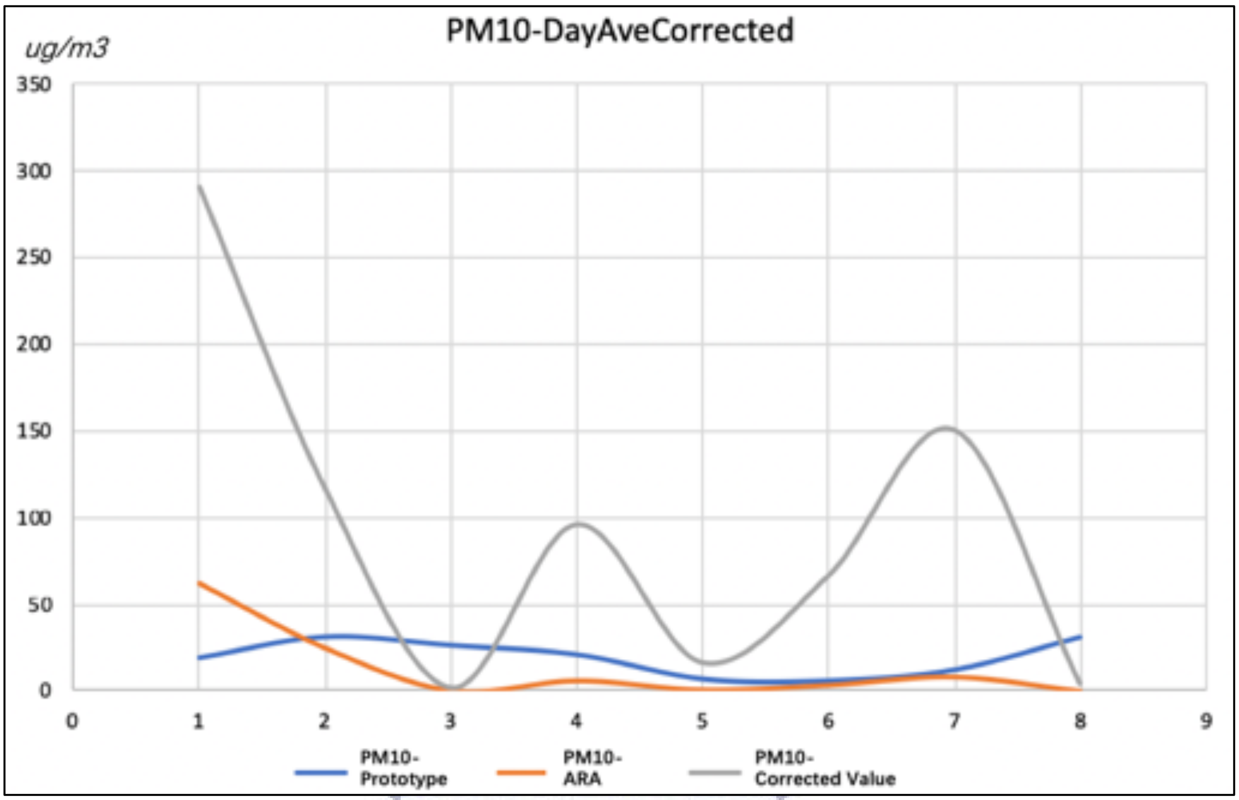


Figure 5-39: PM10 result comparison in day average with ARA sampler (2)

The above data was from the first part of the experiment after the two devices had been moved closer. The experiment was running for a week between 28/09/2018 and 05/10/2018. The blue line is the data from our device and the red line is the data of the ARA equipment. Since the ARA sampler actually has two systems, one uses sensor module to measure a real-time reading, and the other uses the gravimetric method with filters. The filter is weighed before the test and after the test to calculate the PM10 concentration in a certain period of time in the air. ARA equipment often uses the gravimetric method to calibrate the real-time PM sensor. For the formula, please refer to the **Equation 5-1**. The grey line in Figure 5-39 is the estimated corrected value actually obtained by the PM10 sensor real-time readings multiplying the correction factor. The formula for correction factor is shown below:

$$\text{Correction Factor} = \frac{\text{Filter Method Concentration } (\mu g/m^3)}{\text{Sensor PM10 Average } (\mu g/m^3)}$$

Equation 5-1

The correlation coefficient table is also shown below:

Table 5-6: The correlation coefficient between our device data and ARA data (2)

CC	TEMP	PM10	PM10Corrected
Hour Ave	0.97368679	0.21160255	0.09747487
Day Ave	0.99305647	0.13056659	-0.0760461

It is not difficult to see that the temperature data in this part is still very good, but the PM10 data is still very different. Although the correlation coefficient of the PM10 in hour average has increased a from 0.0103854 to 0.21160255 after a result of moving the two devices closer, it still can be seen from the figure that the difference between the two is still large. The difference is even greater when comparing our PM10 data to the corrected PM10 data. However, it can also be found that even with the same equipment, there is a big difference between the data from the ARA sensor real-time readings and the estimated accuracy data.

Figures 5-38 to 5-41 below show the experiment carried out on the balcony, between the 22/10/2018 to 26/10/2018. As with pervious figures, the blue line is the data from our device, the red line is the data of the ARA equipment and the grey line is the estimated accuracy value by calculated the weight of the filter.

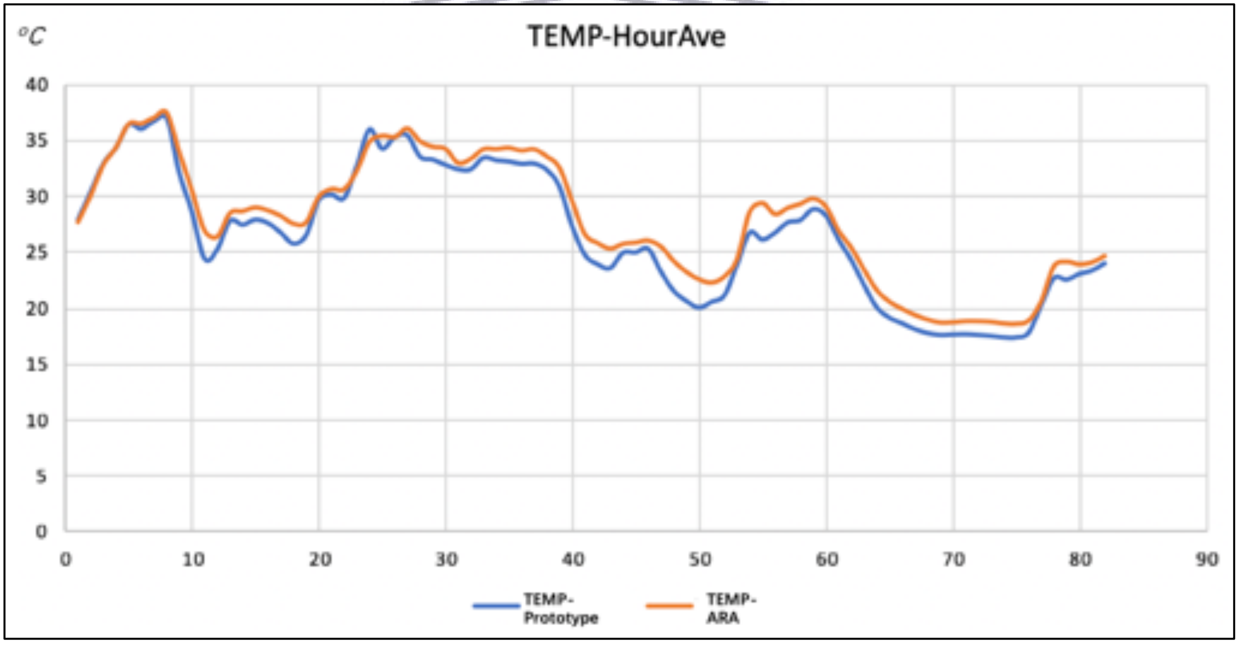


Figure 5-40: Temperature result comparison in hour average with ARA sampler (3)

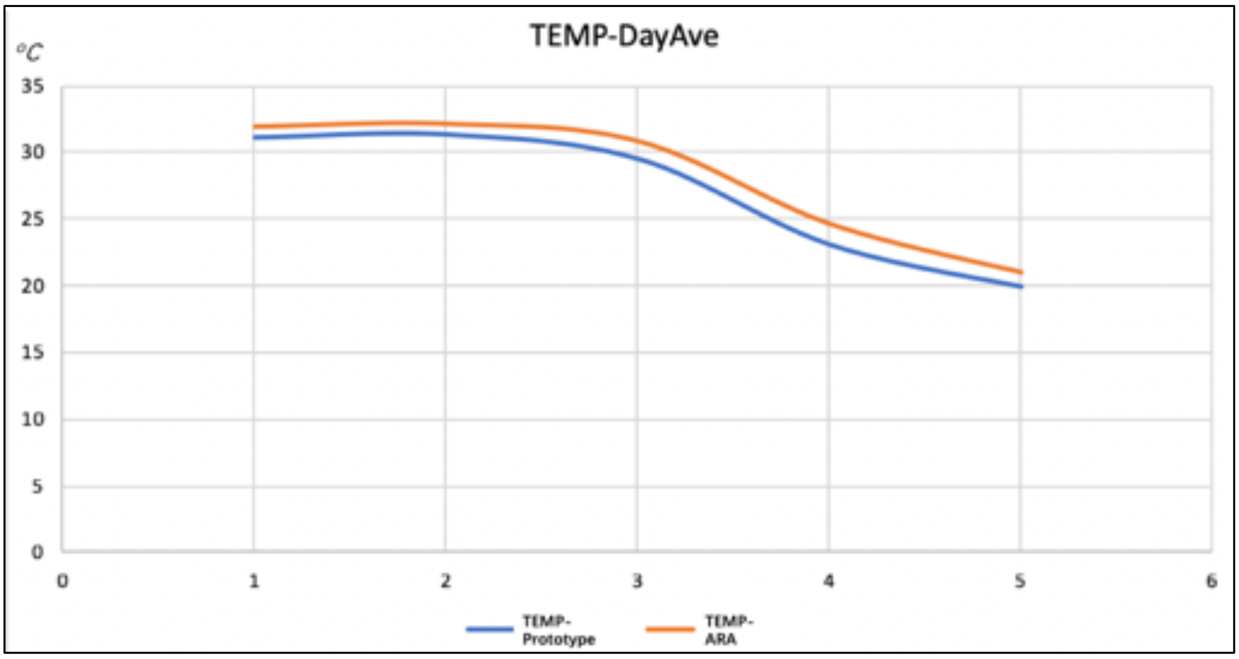


Figure 5-41: Temperature result comparison in day average with ARA sampler (3)

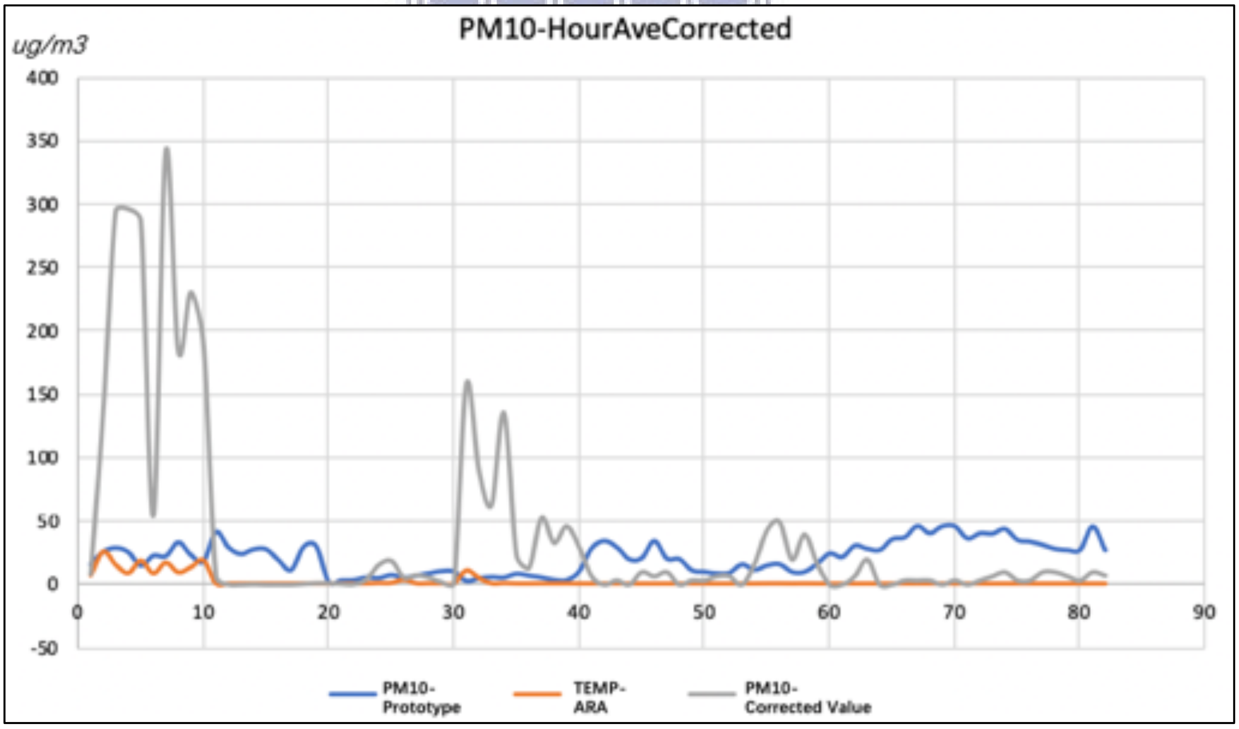


Figure 5-42: PM10 result comparison in hour average with ARA sampler (3)

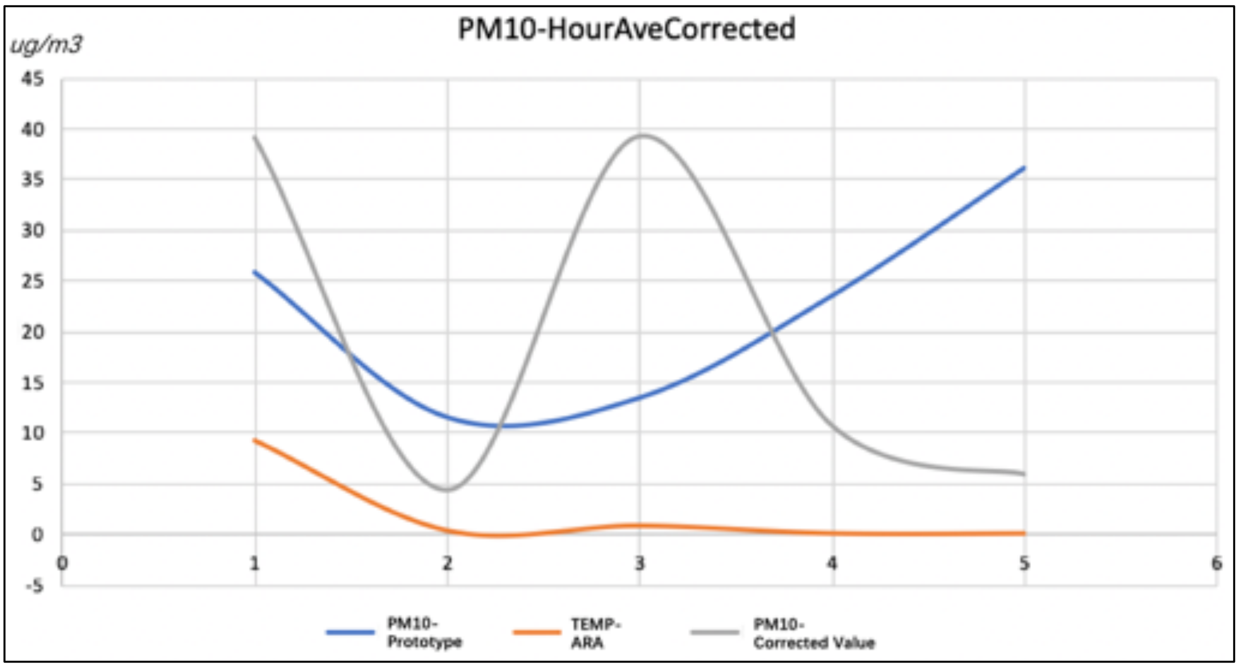


Figure 5-43: PM10 result comparison in day average with ARA sampler (3)

Table 5-7: The correlation coefficient between our device data and ARA data (3)

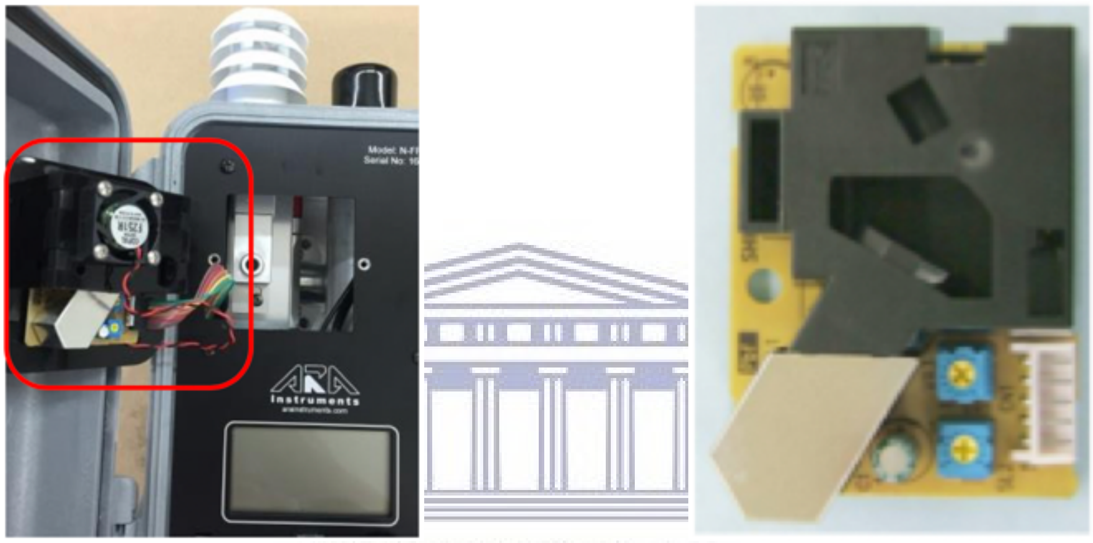
CC	TEMP	PM10	PM10Corrected
Hour Ave	0.99271503	-0.022391	-0.0800811
Day Ave	0.99844503	0.1544635	-0.1978002

From the data, the correlation coefficient values were obtained and shown on table 5-7 above. Similarly, the temperature data gave good results but consistently, the PM data obtained were different. Both when comparing the ARA sensor real-time data and the estimated accuracy data. They only showed a similar downward trend in the first two days from the PM10 day average figure.

Analysis

The data obtained by the GPS sensor of our device is, 33.915520 S and 18.572069 E, which is basically the same as the coordinates 33.915567 S and 18.572156 E that Google map gets for the Eurosteel warehouse. It can be concluded that the GPS sensor works well, and the temperature sensor also shows good stability and accuracy in the test. Due to the lack of comparative data, the CO sensor's working status cannot be determined from this test, but for the PM sensors, the data compared with ARA was not particularly ideal. It should be noted that in the last part of the experiment carried out on the balcony, some errors were pointed out in the real-time data collected by the ARA sampler, which were likely as a result of the constant human intervention / proximity. It can also be seen from Figure 5-37 that the real-time PM10 readings of ARA equipment during this period were close to 0 in most of the test. Therefore, we believe that in air detection, especially outdoor monitoring, the accuracy of the data is often related to the detection environment of the equipment and the stability of the monitoring device. Thus the lower the human intervention and proximity, the less likely the probability of affecting the data collected. This is one advantage of our prototype as it does not require a lot of manual intervention to operate

By reading the instructions of the ARA sampler, it is pointed out that the air sensor needs to be cleaned after running for a certain period of time. Therefore, it is also necessary to develop standard cleaning specifications for our device like the ARA's, and from the manual of ARA sampler, it is found that the real-time reading sensor actually used in this device is very similar to the PPD42NSPM PM sensor produced by a Japanese company Shinyei, as shown in Figure 5-42 below.



(a) (b)

Figure 5-44: ARA sampler PM sensor(a)(ARA, 2016), PPD42NSPM PM sensor(b)(Shinyei, 2017)

The cost of this PPD42NSPM PM sensor was between ZAR 50 to 100 (as at the time of writing), which was similar in cost to the sensors used in our device. They are both low-cost laser sensors used for the PM detection. To a large extent, the sensors are similar, the major differences between both devices is the design of the outer casings. Through this experiment it is easy to see that although the PM data was not particularly ideal, the overall working state of our device system is stable. Our prototype can thus be classified as competent and suitable for use in scenarios where the professional equipment ARA sampler is used.

5.1.5 Test 4: Calibration against the gas stations of SAAQIS in Cape Town

This test serves to request the support of the gas stations of SAAQIS in benchmarking our prototype air pollution system against the SAAQIS calibrated air pollution stations. The prototype has proved that it is now able to measure and save air pollution into the system, but it needs to be validated before further improvements. Our request consists of getting the permission to install our prototype inside one of the SAAQIS stations to collect air pollution data, for a period of one month in a Stellenbosch gas monitoring station. Data collected included: temperature, humidity, Carbon Monoxide and Particulate Matter in minutes.

Plans for a detailed test: In the first week we were to visit and study the workings of the station and make necessary adjustments to our prototype for it to function in the station. The prototype would be deployed in the station for a 4 weeks period to collect the air pollution data. Data collected by the Stellenbosch station over the same time period would also be collected in order to compare with the data collected by our prototype for the validation and possible improvement of our system. If the test showed good results, we hoped that our system could receive strong support from the government and other institutions to help South Africa improve on air quality evaluation.

Unfortunately, though we received approved and authorization from the Western Cape Government to carry out the test, there was no personal to take us to the Stellenbosch gas monitoring station. The progress of the experiment has thus been halted. Subsequently and as a result of research time constraints, test 4 was not done and no results are available for it.

5.2 User's Guide

5.2.1 System Summary

The hardware device is one part of the project "Smart cities pollution monitoring System", it is used to collect the data and save them into the database on a server. The project has four layers; which are: the sensing, networking, middleware and application layers. This research work focused primarily on the sensing and networking layers, including the hardware and software as well as the implementation. Aspects of the middleware and application layers were also done. The hardware uses the software to choose a best connection to transmit the data to the middleware layer before forwarding to the users. The main programs were running on a Raspberry Pi board.

The system of the whole research includes three parts: the transmitter (the hardware monitoring device), the controller (the admin control app) and the receiver (server and users). The transmitter contains all hardware parts and can be connected to multiple power supplies. The transmitter needs to access the Internet through communication module(s) to transmit data to the receiver. The receiver needs to have the Python installed, it could be PC, laptop or another hardware monitoring device. The admin control app needs to be installed in the controller and could be an app running on an Android device. It is an App that is compatible with Android 1.5 API Level3 or higher.

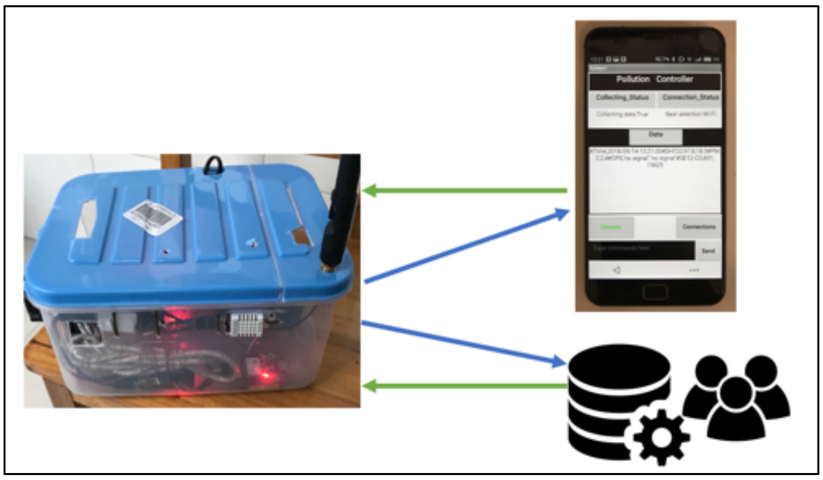


Figure 5-45: Three parts of the system

5.2.2 Users access levels and contingencies

The system is aimed at providing pollution data to public, therefore anyone should be able to have the access to the data collected by our system. Only the Admin has the right to control the entire system through a special software. In addition, there are certain restrictions on the use of the hardware device. For example, devices used for environmental monitoring detection will require professional installation and periodic maintenance. The data of such devices will be uploaded to the unified server. For the private and domestic devices there will be certain restrictions set about what data should be upload to the server. The reason is that the accuracy of data collection will be greatly affected by the non-professional installation and the difference in test environments.

With respect to contingencies, in cases where there is no network access available in the region and no communications are available, the data is stored on the memory space. However, after if there is no access for a prolonged period, the storage space within the device will get full and data newly collected would be last. Also, in scenarios where the Internet access within a region of interest is unstable, the device would not be able transmit data to the data using the Internet.

5.2.3 Getting started

Installation

The information of latest version of all the programs and control app can be found from www.cs.uwc.ac.za/~schen. Where the *HardeareProgram* contains the python programs for the hardware device; the *MiddleServer* and *WechatServer* contain the python programs for the server; the *SensorMoudule* contains the Arduino code for sensor modules. The *PollutionControler.apk* app needs to be downloaded and installed on an Android device, used for admin to control the hardware device. The details of installation instructions can refer to the chapter Implementation.

System Functions

Hardware device:

The hardware device is connected with the sensor modules; the data from the sensor modules can be stored in the device and by using the network selection algorithm to send to server through different communications.

Server:

The python programs on the server can receive, prepare and store data into database then push to the users.

Controller:

PollutionControler is a tabbed mobile application. Admin can control the device to start and stop data transmission by using the *PollutionControler* app and can also control the data transmission method.

Using the system

1. Run the Python programs mentioned above on server for receiving and pushing data.
2. Install the hardware device in a sheltered and high place to reduce the impact from rain and round dust. Plug the device to a power supplier.
3. If there is Internet access, plug the Ethernet cable or configure the Wi-Fi username and password correctly to the device; if there is no Internet, configure other communications module or just save data in the device without transmitting data to the server.
4. Plug the configured sensors modules. Some of the sensors have a certain life span and it requires regular maintenance and replacement.
5. Pair the Bluetooth of device with the admin mobile app and connect the device with the admin app to check if the system starts working properly.
6. Then admin is able to switch communications or send commands to the device manually. Admin exits the app or disconnecting Bluetooth will not affect the data transmission.
7. All of the data will be finally stored in the database of the server, then pushed to the users according to the requests.
8. Users are able read data thorough WeChat public channel and Pubhub.

Special instruction for error correction

In case, the device does not work properly, first connect the admin control app with the device and try to enter the command "*savingon*" then "*savingoff*" to restart the saving function of the device. If problem is not solved, then try to enter the command "*reboot*" restart the device or disconnect and reconnect the power supplier to force restart the device. Some other useful commands for the device also include: "*bton*" and "*btoff*", these two commands are used to control whether the device sends data to the mobile control app via Bluetooth: "*screenon*" and "*screenoff*", these two commands are used to control whether the device displays data on the device screen.

5.3 Summary

This chapter mainly proves the usability and stability of the device through several tests. The chapter also describes the details of the tests, how potential users can get started with the system with errors free configuration. The advantages of the system include: low cost, safety and reliable communication between multiple platforms; it can also meet the requirements of scalability and reliability. The system provides an economy and stable solution for air monitoring. The series of tests conducted also proved that the system based on the developed hardware device can meet both commercial and domestic needs. The system is capable of working in a stable matter irrespective of the use case requirement that is either indoor or outdoor.

6. Conclusion

The rapid industrialization and increase in carbon emission have made air pollution a major concern. The health hazards resulting from polluted air are numerous and dire. Information on air pollution levels is limited especially in developing countries. This is due to the exorbitant cost of deploying professional air pollution monitoring systems, and in cases where the systems are in place, the information are either not readily available or are difficult to interpret. This research work proposed and built an air monitoring system based on Raspberry Pi, Arduino and air pollution sensors. The proposed solution is relatively cheap, easy to control and administer and provides opportunistic data dissemination to enable different networks to be selected based on their current quality of service.

The proposed prototype was compared to professional devices used by SAAQIS to monitor CO levels and also the professional ARA PM sampler to measure temperature and PM levels. The results show that although our prototype is cheaper and smaller in size, but it actually showed relatively better performance in air pollution monitoring and especially on mobility which other devices don't have. The accuracy of the data of the professional device were relatively better, however the accuracy of the data depends on multiple factors such the accuracy of the sensor(s) used in the device and the design of the device. Therefore, it is recommended that sensors with better accuracy should be used in the future development, and this could further increase the accuracy of the device. In terms of functionality, our prototype was able to collect GPS, temperature, CO and PM data, while the professional devices were limited to a few functionalities mostly temperature, CO or PM.

6.1 Comparison with professional air pollution analysers

Tables 6-1 and 6-2 show a concise comparison of our prototype with professional air pollution analysers.

Table 6-1: Comparison between our device and Professional analysers

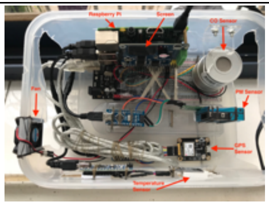
Device	Name	Price	Power consumes	Portability	Data precision	Function ality
 Raspberry Pi Screen CO Sensor Fan PM Sensor GPS Sensor Temperature Sensor	Smart Cities Air Pollution Monitoring device (with GSP, PM, CO, Temp sensors)	5,400 Rand	4.7 watts (with data transmission modules)	Small size, Less weight, Portable, 2Kg	±0.15 ppm	GPS, Temp, CO, PM
	Thermo Fisher Scientific 48i gas filter correlation Co analyser	185,712 Rand	275 watts	Big size, not portable, 22Kg	±0.1 ppm	Temp, CO

Table 6-2: Comparison between our device and ARA N-FRM sampler

Device	Name	Price	Power consumes	Portability	Data precision	Functionality
	Smart Cities Air Pollution Monitoring device with louvered shield (with PM, Temp sensors)	1,300 Rand	±2.6 watts (without data transmission modules)	Small size, Less weight, Portable, 1.5Kg	±10% @100~500µ g/m3 ±10µg/m3@0~100µ g/m3	Temp, PM
	ARA N-FRM sampler	>140,000 Rand	±3.0 watts	Big size, not portable, 6.8-11Kg	±1% for filter sampling ±10% for real-time data	Temp, PM

From this, it is clear that our prototype has a broad application prospect. Being a modular device, the same device can be used in multiple application areas by simply selecting the appropriate sensor for the pollution data needed. As shown in Figure 6-1, the device can be used for air pollution detection, human body data detection as well as water pollution detection. Being modular, the corresponding sensor modules can be independently disassembled and calibrated.

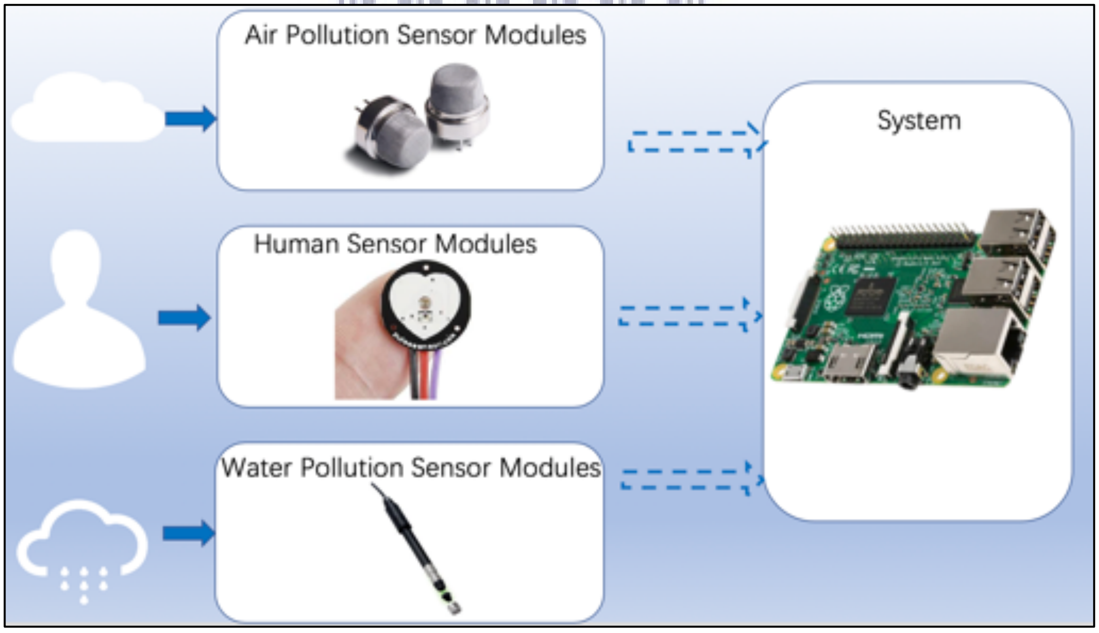


Figure 6-1: Possible fields of application of the system

Besides the advantages of being low cost, safe and reliable, our device also has less dependence on the Internet network and electricity. It can thus be readily deployed in areas with limited Internet access and electricity. The low power consumption footprint allows the prototype to be powered effectively by solar panels. By connecting to a mesh network built on LoRa and/or Zigbee, our prototype can transmit data over a long distance and even to the internet as long as one of the devices in the mesh network has internet access. If the server is on a LoRa network, our prototype can communicate directly with it as long as it is within range of LoRa.

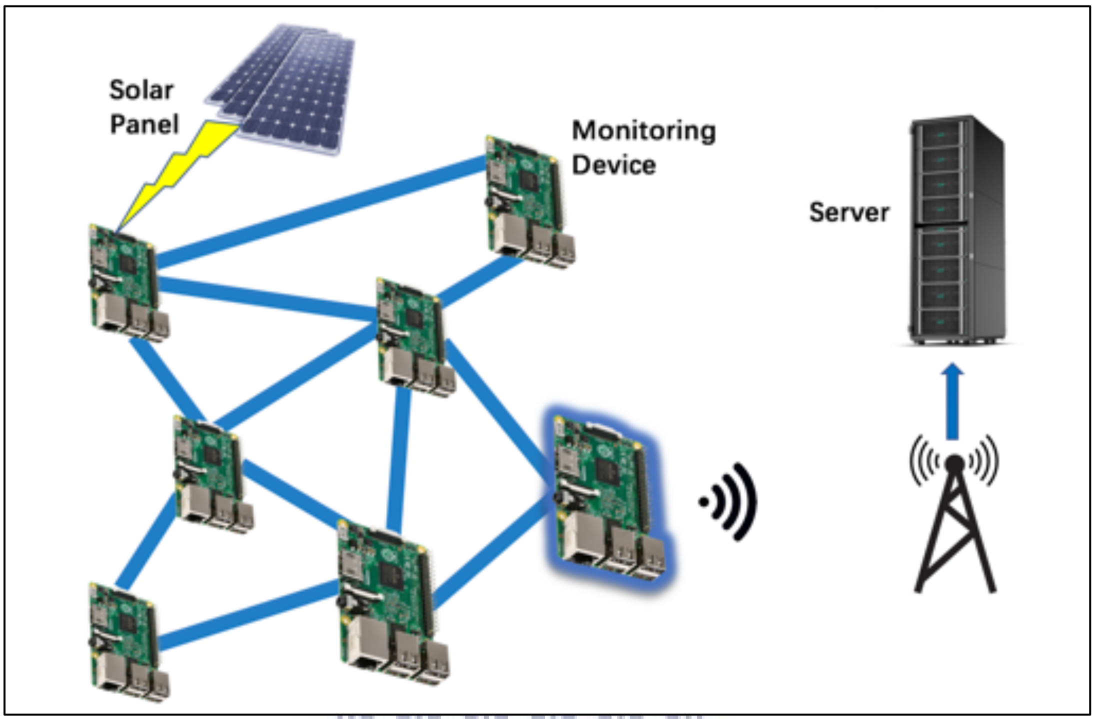


Figure 6-2: LoRa, Zigbee MESH network

In conclusion, after deployment and a series of tests, the device met the requirements of scalability and reliability, while being economical. The system based on this platform can meet both commercial and domestic requirements. The system is capable of providing stable level of functionality irrespective of the installation mode – that is either in fixed or mobile installation.

As a side note, other low-cost sensors were also tested on the prototype such as UV sensor, noise sensor, human temperature sensor and human heart rate sensor. These were done to lay a foundation for research in the field of healthcare.

6.2 Future work

The next phase of this research will be to install more sensors of different types and precision to the system. Interfacing and docking with pub/sub protocols such as CoAP, MQTT, AMQP, ZeroMQ would also be a welcome addition as this would enable users check the recent air quality trend via a website. Finally, a network of sensing devices could be deployed around Cape Town to take measurements that can be integrated with the SAAQIS stations in South Africa to complement the current air pollution system and thus fill the visibility gap in its pollution map.

Ultimately, we hope that this research will help the public understand and have access to information on surrounding air condition in real-time, popularize knowledge of air pollution and used in campaigns for protecting the environment.

References

ARA (2016) *ARA N-FRM Sampler*, *ARA Instruments*. Available at: http://arainstruments.com/products/n-frm-sensor/ (Accessed: 31 October 2018).

Ashining (2017a) *AS32-TTL-1W LoRa module*, *Ashing.com*. Available at: http://www.ashining.com/show?product_id=59 (Accessed: 10 October 2018).

Ashining (2017b) *AS32-TTL-1W LoRa module datasheet*, *Ashing.com*. Available at: http://www.ashining.com.cn/data/as32_ttl_1w.zip (Accessed: 10 October 2018).

Ashton, K. (2009) 'That "Internet of Things" Thing', *RFiD Journal*, 22, pp. 97–114. Available at: https://www.rfidjournal.com/articles/pdf?4986.

Augustin, A. *et al.* (2016) 'A Study of LoRa: Long Range &amp; Low Power Networks for the Internet of Things', *Sensors*, 16(9), p. 1466. doi: 10.3390/s16091466.

Bagula, A. *et al.* (2012) *Ubiquitous sensor networking for development (USN4D): An application to pollution monitoring*, *Sensors*. doi: 10.3390/s120100391.

Baidu (2013) *Raspberry Pi*, *Baidu*. Available at: https://baike.baidu.com/item/Raspberry Pi (Accessed: 31 October 2018).

Baidu (2018a) *Arduino*, *Baidu*. Available at: https://baike.baidu.com/item/Arduino (Accessed: 31 October 2018).

Baidu (2018b) *BeagleBoard*, *baidu.com*. Available at: https://baike.baidu.com/item/BeagleBoard/9342116 (Accessed: 31 October 2018).

Brauer, M. *et al.* (2012) 'Exposure Assessment for Estimation of the Global Burden of Disease Attributable to Outdoor Air Pollution', *Environmental Science & Technology*, 46(2), pp. 652–654. doi: 10.1021/es2025752.

Chen, S. *et al.* (2018) 'Internet-of-Things for Air Pollution Monitoring in Smart Cities : An Efficient Network Selection Model', *ISTAfrica*, pp. 1–13. Available at: https://www.researchgate.net/publication/325171920_International_Information_Management_Corporation.

ChinaScienceCommunication (2017) *Air Pollution*, *Baidu*. Available at: https://baike.baidu.com/item/空气污染/341495?fr=aladdin (Accessed: 30 October 2018).

CNEMC (2017) *How to ensure the data accuratcy of grid monitoring*, *China National Environmental Monitoring Centre*. Available at: http://www.sohu.com/a/133123698_692727 (Accessed: 30 October 2018).

Cordeur, M. le (2015) *Africa's hidden killer causes 600 000 deaths every year*, *fin24*. Available at: https://www.fin24.com/Economy/Africas-hidden-killer-20150605 (Accessed: 30 October 2018).

Dalgleish, T. *et al.* (2007) 'Ambient air pollution: A gloabla assessment of exposure and burden of disease', *Journal of Experimental Psychology: General*, 136(1), pp. 23–42.

Available at: http://apps.who.int/iris/bitstream/handle/10665/250141/9789241511353-eng.pdf?sequence=1.

DTK (2017) *DRF1609H*, *DTK Electrionics*. Available at: http://www.dtkcn.com/product/product13.html (Accessed: 25 September 2018).

Ebyte (2018a) *How does Zigbee become the leading connection technology in smart homes?*, *Chengdu Ebyte Electroic Techology Co., Ltd.* Available at: http://www.ebyte.com/new-view-info.aspx?id=426 (Accessed: 31 October 2018).

Ebyte (2018b) *Which one is the best partner in the IoT trend, Loro, Sigfox or NB-IoT ?*, *Chengdu Ebyte Electroic Techology Co., Ltd.* Available at: http://www.ebyte.com/new-view-info.aspx?id=431 (Accessed: 31 October 2018).

Ebyte (2018c) *Why Zigbee has tried it best to do MESH network in smart home*, *Chengdu Ebyte Electroic Techology Co., Ltd.* Available at: http://www.ebyte.com/new-view-info.aspx?id=418 (Accessed: 31 October 2018).

Elecfans (2018) *Seven differences between GPRS module and GSM module in application*, *elecfans.com*. Available at: http://www.elecfans.com/baike/tongxingjishu/wuxiantongxin/20180119618901.html (Accessed: 31 October 2018).

Electronics, G. (2018) *SIM800L GSM MODULE 5V version V2.0*, *Geek Electronics*. Available at: https://geekelectronics.io/shop/sim800l-gsm-module-5v-version-v2-0/ (Accessed: 15 October 2018).

Elizabeth, G.-D. (2016) *How the Internet of Things will change our lives*, *International Organization for Standardization*. Available at: https://www.iso.org/news/2016/09/Ref2112.html (Accessed: 31 October 2018).

Froiz-Míguez, I. *et al.* (2018) 'Design, Implementation and Practical Evaluation of an IoT Home Automation System for Fog Computing Applications Based on MQTT and ZigBee-WiFi Sensor Nodes', *Sensors*, 18(8), p. 2660. doi: 10.3390/s18082660.

Google (2018) *No Title*, *Google*. Available at: https://www.google.co.za/maps/place/Eurosteel/@-33.9160975,18.5688683,728m/data=!3m1!1e3!4m5!3m4!1s0x1dcc505510185cc5:0xba5c1c8cd36fe336!8m2!3d-33.916102!4d18.571057 (Accessed: 31 October 2018).

Hartely, T. (2014) *AirPi Kit v1.4 - Raspberry Pi weather shield by tmhrtly on Tindie*, *Tindie*. Available at: https://www.tindie.com/products/tmhrtly/airpi-kit-v14-raspberry-pi-weather-shield/ (Accessed: 31 October 2018).

Hartley, T. (2013) *AirPi*, *Airpi.es*. Available at: http://airpi.es/about.php (Accessed: 31 October 2018).

Jameco (2016) *How to Navigate Your Raspeberry Pi 3 Model B*, *Jameco.com*. Available at: https://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html (Accessed: 31 October 2018).

Ji, X. (2014) *Summary of PM2.5 determination methods*, *Baidu*. Available at: https://wenku.baidu.com/view/ee48b4fa4028915f804dc2af.html (Accessed: 31 October 2018).

Kithion, B. (2016) *Arduino vs Raspberry Pi Vs BeagleBone – Which one to choose for IOT?*, *learniot*. Available at: https://learniot.wordpress.com/2016/04/14/arduino-vs-raspberry-pi-vs-beaglebone-which-one-to-choose-for-iot/ (Accessed: 31 October 2018).

Letswireless (2018) *SMS wireless alarm control system*, *Letswireless.com*. Available at: http://www.letswireless.com.cn/cn/yyfa/show.asp?id=11 (Accessed: 31 October 2018).

Liu (2016) *Arduino Nano*, *360doc*. Available at: http://www.360doc.com/content/15/0930/10/28009762_502400834.shtml (Accessed: 9 September 2018).

Liu, T. (2016) *Digital-output relative humidity & temperature sensor/module DHT22*, *Sparkfun.com*. Available at: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf (Accessed: 31 October 2018).

Locke, D. (2010) *MQ Telemetry Transport (MQTT) V3.1 Protocol Specification*, *ibm.com*. Available at: https://www.ibm.com/developerworks/cn/webservices/ws-mqtt/index.html (Accessed: 31 October 2018).

Ma, Y. *et al.* (2008) 'Air pollution monitoring and mining based on sensor Grid in London', *Sensors*, 8(6), pp. 3601–3623. doi: 10.3390/s8063601.

Man, J. (2016) *Raspberry Pi 3 Model B Technical Specifications*, *Element14*. Available at: https://www.element14.com/community/docs/DOC-80899/l/raspberry-pi-3-model-b-technical-specifications (Accessed: 31 October 2018).

Mandava, T. (2019) *A Decision support system for accessing the impact of air pollution on human health*.

MARTIN, T. (2016) *How to setup Bluetooth on a Raspberry Pi 3*, *cnet.com*. Available at: https://www.cnet.com/how-to/how-to-setup-bluetooth-on-a-raspberry-pi-3/ (Accessed: 31 October 2018).

Ngandu, K., Ouahada, K. and Rimer, S. (2018) 'Smart Meter Data Collection Using Public Taxis', *Sensors*, 18(7), p. 2304. doi: 10.3390/s18072304.

OSXDaily (2012) *How to Install Paramiko and PyCrypto in Mac OS X the Easy Way*, *OSXDaily.com*. Available at: http://osxdaily.com/2012/07/10/how-to-install-paramiko-and-pycrypto-in-mac-os-x-the-easy-way/ (Accessed: 31 October 2018).

Quot (2019) *National Air Quality*. Available at: https://pan.baidu.com/s/1gd8GUxt?errno=0&errmsg=Auth Login Sucess&&bduss=&ssnerror=0&traceid=#list/path=%2F&parentPath=%2F (Accessed: 21 February 2019).

Raspberrypi (2016) *RASPBERRY PI 3 MODEL B*, *Raspberrypi.org*. Available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (Accessed: 31 October 2018).

Ritchie, H. and Roser, M. (2017) *Air Pollution*, *Our World in Data*. Available at: https://ourworldindata.org/air-pollution (Accessed: 30 October 2018).

Roy, R. (2016) *The cost of air pollution in Africa*. Pairs: OECD Publishing. doi: http://dx.doi.org/10.1787/5jlqzq77x6f8-en.

SAAQIS (2004) *About SAAQIS*, *saaqis.org.za*. Available at: http://www.saaqis.org.za/About.aspx (Accessed: 30 October 2018).

SAAQIS (2015a) *District Municipality AQMP development status*, *Saaqis.org.za*. Available at: http://www.saaqis.org.za/AQPlanning01.aspx (Accessed: 30 October 2018).

SAAQIS (2015b) *View Air Quality Monitoring Stations on Map*, *Saaqis.org.za*. Available at: http://www.saaqis.org.za/Mashup.aspx (Accessed: 30 October 2018).

Shinyei (2017) *Particle Sensor Model PPD42NS*, *switchdoc.com*. Available at: http://www.switchdoc.com/wp-content/uploads/2018/08/Grove_-_Dust_sensor.pdf (Accessed: 9 November 2018).

SIMCom (2016) *SIM800L GSM/GPRS Mdoule*, *SIMCOM*. Available at: http://www.simcomm2m.com/UploadFile/TechnicalFile/SIM800L SPEC CN170914.pdf (Accessed: 27 September 2018).

Smythe, J. (2017) *ARDUINO VS BEAGLEBONE VS RASPBERRY PI*, *Mighty Gadget*. Available at: https://mightygadget.co.uk/arduino-vs-beaglebone-vs-raspberry-pi/ (Accessed: 31 October 2018).

Tech4Good (2016) *People's Award, Winner, 2016*, *Tech4GoodAwards*. Available at: https://www.tech4goodawards.com/finalist/arnav-sharma/ (Accessed: 31 October 2018).

ThermoFisher (2018) *Model 48i CO Analyzer*, *thermofisher.com*. Available at: https://www.thermofisher.com/za/en/home/industrial/environmental/air-quality-analysis/ambient-gas-monitoring/carbon-monoxide-monitoring.html# (Accessed: 30 October 2018).

Tu, Z. (2017) *Raspberry Pi fixed serial port number according to USB port information*, *Zhihu*. Available at: https://zhuanlan.zhihu.com/p/30276810 (Accessed: 31 October 2018).

U-blox (2015) *NEO-6 u-blox 6 GPS Modules Data Sheet*, *u-blox.com*. Available at: https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf (Accessed: 31 October 2018).

UNenvironment (2016) *Air Pollution: Africa's Invisible, Silent Killer*, *unenvironment.org*. Available at: https://www.unenvironment.org/news-and-stories/story/air-pollution-africas-invisible-silent-killer-0 (Accessed: 30 October 2018).

Vidal, J. (2016) *Air pollution more deadly in Africa than malnutrition or dirty water, study warns*, *The Guardian*. Available at: https://www.theguardian.com/global-development/2016/oct/20/air-pollution-deadlier-africa-than-dirty-water-or-malnutrition-oecd (Accessed: 30 October 2018).

Waveshare (2015) *PIONEER600 , Raspberry Pi Expansion Board User Manual*, *waveshare.com*. Available at: https://www.waveshare.com/w/upload/1/17/Pioneer600-User-Manual.pdf (Accessed: 31 October 2018).

WeChat (2017) *Wechat public platform access guide*, *Wechat*. Available at: https://mp.weixin.qq.com/wiki?t=resource/res_main&id=mp1421135319 (Accessed: 31 October 2018).

Western Cape Provincial Government (2015) 'State of Air Quality Management 2015 State of Air Quality Management 2015', p. 80. Available at: https://www.westerncape.gov.za/eadp/files/atoms/files/State Of Air Quality Monitoring Report 2015.pdf.

WHO (2009) 'Global Health Risks', *WHO Library Cataloguing-in-Publication Data Global*, 1, p. 62.

WHO (2012) 'Burden of disease from Household Air Pollution for 2012', *World Health Organization, Global Health Risks*, 1(February), pp. 1–17. doi: 10.1016/S0140-6736(12)61766-8.Smith.

WHO (2016) *WHO releases country estimates on air pollution exposure and health impact*, *who.int*. Available at: http://www.who.int/en/news-room/detail/27-09-2016-who-releases-country-estimates-on-air-pollution-exposure-and-health-impact (Accessed: 30 October 2018).

WHO (2017) *WHO releases country estimates on air pollution exposure and health impac*, *WHO*. Available at: http://www.who.int/en/news-room/detail/27-09-2016-who-releases-country-estimates-on-air-pollution-exposure-and-health-impact (Accessed: 30 October 2018).

Wiki (2018) *Arduino Nano*, *Wiki*. Available at: https://wiki.eprolabs.com/index.php?title=Arduino_Nano (Accessed: 31 October 2018).

Winsen (2016) *Electrochemical Gas Detection Module User ' s Manual V1 . 3 （Model：ZE11 ）*, *Zhengzhou Winsen Electronics Technology Co.,Ltd*. Available at: https://www.winsen-sensor.com/d/files/PDF/Gas Sensor Module/Industrial Application Gas Sensor Module/ze12-electrochemical-module-manualv1_5.pdf (Accessed: 31 October 2018).

Winsensor (2012) *The future development of environmental sensors such as air quality*, *winsensor*. Available at: http://www.winsensor.com/hangye_news/1037.html (Accessed: 31 October 2018).

Wu, J. (2014) *GSM module*, *CSDN.net*. Available at: https://blog.csdn.net/wujiangguizhen/article/details/25421179 (Accessed: 31 October 2018).

Xingyu (2016) *Causes of smog formation and PM2.5 detection method*, *Baidu*. Available at: https://wenku.baidu.com/view/40a20cfa4b35eefdc9d33357.html (Accessed: 31 October 2018).

Yong, Z. and Haoxin, Z. (2016) 'Digital universal particle concentration sensor PMS5003 series data manual'. doi: 10.1016/j.wocn.2009.10.005.

Zymbit (2016) *Goleta Air Monitoring Project*, *Zymbit*. Available at: https://forms.zohopublic.com/phil12/form/ZymbitContactMe/formperma/xui9Pd3ROmE0Kut_0Yc3zJ0y7oWH7LSVnVIxSoXYtOQ (Accessed: 31 October 2018).

Appendix A

(Applied sensors and communication modules table)

Table A-1: Price list of the applied sensors

Sensor Name	Parameters	Price
ZE12-CO	CO	4000 Rand
PMSA003	PM1/PM2.5/PM10	210 Rand
NEO-6M	Longitude/Latitude	65 Rand
DHT22	Temperature/Humidity	40 Rand
MAX30100	Heart rate/blood oxygen	20 Rand
MLX90615	Human temperature	65 Rand
GY-301	UV level	45 Rand
MAX9814	Sound	20 Rand

Table A-B: Price list of the applied communication modules

Communication Module	Parameters	Price
Sim800l	GSM/GPRS	50 Rand
AS32-TTL-1W	LoRa	60 Rand
DRF1609H	Zigbee	100 Rand

Appendix B

(Request letter of authorization to test our air quality system in Stellenbosch)

Appendix C

(Approve letter of authorization to test air quality system in Stellenbosch)

Appendix D

(PMSA003 PM sensor manual(Yong and Haoxin, 2016))

Appendix E

(ZE12-CO sensor manual DHT22(Winsen, 2016))

Appendix F

(DHT22 Temperature sensor manual(T. Liu, 2016))