School of Electrical & Telecom Engineering

TELE4121

Thesis Report

Supervisor: A/Prof. Vijay Sivaraman

November 2011

"Android interface for pollution monitoring system"

Dawei Lu

3297612

UNSW

THE UNIVERSITY OF NEW SOUTH WALES

Thesis title: Android interface for pollution monitoring system
Topic number: VR35
Student Name: Dawei Lu **Student ID:** z3297612

A. **Problem statement**

Most Australians live in large cities. Urban air pollution is naturally one of their important concerns. The quality of the air we breathe affects our health. Although Australia's air quality is better than many other comparable countries, it is important to take steps now to ensure that air quality is protected into the future.The problem with current system is that it can't effectively monitor some regions such as heavy industries company areas, underground railways, tunnels and so on.The Haze watch is introduced here exploits this fact by taking advantage of the built-in GPS hardware found in many such as Smart phones. So that a real-time air quality results will be provide to the public.

B. **Objective**

Our overall project objective is to create a Simple, Utility application but has great performance that satisfying all user. It can be classified into create and optimize android phone application to conveniently monitor and report a user's pollution exposure within a fast, stable, and low memory and power cost operation environment. Also helped with sensor calibration to get reliable data.

C. **My solution**

Create more functionality inside the application.
Create different operation mode to overcome the highly power consumption.
Create Bluetooth reconnection method to deal with problems after connection lost.
Optimize the code to achieve a fast and lower memory cost application.
Display more information and filter the non-sense data.
Built up gas container and do the research to help calibration.

D. **Contributions** (at most one per line, most important first)

Built up BatterySaver mode working with interval control to decrease the power drop.
Using self-learning method and database to control the reconnection problem.
Optimized code via optimize Java language and fit into Android and hardware system
Create real-time graphical display of data readings and battery levels.
Warming up notice to filter the incorrect data.
More functionality includes Address, sensor information, flexible data selection.
Background process and termination of GPS, Bluetooth opened by application.
Built up gas container and do the research for future calibration.
Field test

E. **Suggestions for future work**

Stability and Bluetooth connection of application need to be improved.
Sensor calibration.
Upload data privacy.

While I may have benefited from discussion with other people, I certify that this thesis is entirely my own work, except where appropriately documented acknowledgements are included.

Signature: Lu dawei Date: 20 / 10 / 2011

Thesis Pointers

List relevant page numbers in the column on the left. Be precise and selective: Don't list all pages of your thesis!

8	Problem Statement
9~10	Objective

Theory (up to 5 most relevant ideas)

11	Current monitoring method
12~15	Android system
46~47	Elementary of Optimization
55	Calibration

Method of solution (up to 5 most relevant points)

22	Android application
31~34	GPS interval control and operation modes
38~39	Bluetooth auto-connection
46	Program optimization
56~60	Calibration

Contributions (most important first)

23~30，37	Android application's functionality
31~36	GPS interval control and operation modes
38~44	Bluetooth auto-connection with self-learning method.
46~54	Program optimization and test.
56~60	Calibration

My work

9,15,34,38,56	System block diagrams/algorithms/equations solved
n/a	Description of assessment criteria used
36,43~44,48~49	Description of procedure (e.g. for experiments)

Results

9,23,36,43~44,48~50	Succinct presentation of results
22, 30,39~42,46,50~54	Analysis
9,23,36,43~44,48~50	Significance of results

Conclusion

63	Statement of whether the outcomes met the objectives
61	Suggestions for future research

Literature: (up to 5 most important references)

12~15	[4] Android developer, 2011
15~20	[8] Nikolaus Youdale, 2010
24~25	[9] Baidu encyclopedia, 2011
56~60	[16] Kunxuan Bi, 2011
56	[15] Dr. Martin Bucknall, 2011

Abstract

Most Australians live in large cities. Urban air pollution is naturally one of their important concerns. The quality of the air we breathe affects our health. Although Australia's air quality is better than many other comparable countries, it is important to take steps now to ensure that air quality is protected into the future. Even small improvements in air quality can achieve benefits for human health and wellbeing. [*1.Department of the environment and heritage, 2005*]

The report will discuss the “Haze Watch System” that is designed and implemented in the aim of monitoring air pollution quality. Through this application that individuals will be able to closely monitor their personal exposure to air pollution and avoid areas of high concentration which are damaging to their health.

This report focuses on the key component of our work together with my partner Kunxuan Bi in the area of android interface for pollution monitoring system. Detailed design explanations of the Android interface for the monitoring system will be explained and discussed.

Acknowledgements

I would like to acknowledge the other members of the Haze watch project, Kunxuan Bi, Hailian Zhang and Junjie Jiang for their support and hard working throughout this project. I would also like to specially thank my supervisor, Associated Professor Vijay Sivaraman who provided our group with excellent guidance, direction and support throughout this project.

Table of Contents

1.Introduction

Every day, the average person inhales about 20,000 litres of air. Every time we breathe, we risk inhaling dangerous chemicals that have found their way into the air. Air pollution includes all contaminants found in the atmosphere. These dangerous substances can be either in the form of gases or particles. [*2.Oracle, air pollution a global challenge]*

Air pollution can be found both outdoors and indoors. Pollutants can be trapped inside buildings, causing indoor pollution that lasts for a long time. The sources of air pollution are both natural and human-based. As one might expect, humans have been producing increasing amounts of pollution as time has progressed, and they now account for the majority of pollutants released into the air.

Nowadays, government plays the main role in environmental pollution monitoring, the way they monitoring is by building up monitoring sites, which spread all over the entire city. The problem with such system is that it can't effectively monitor some regions such as heavy industries company areas, underground railways, tunnels and so on.

To overcome such issues, further research could be undertaken into more accurate modeling systems where the environmental topology, environmental conditions and characteristics of the pollutant itself determine an approximation.

The Haze watch is introduced here exploits this fact by taking advantage of the built-in GPS hardware found in many such as Smart phones. So that a real-time air quality results will be provide to the public.

1.1 Project objectives

Our overall project objective is to create a Simple, Utility application but has great performance that satisfying all user. It can be classified into the listed groups written below in Figure1:

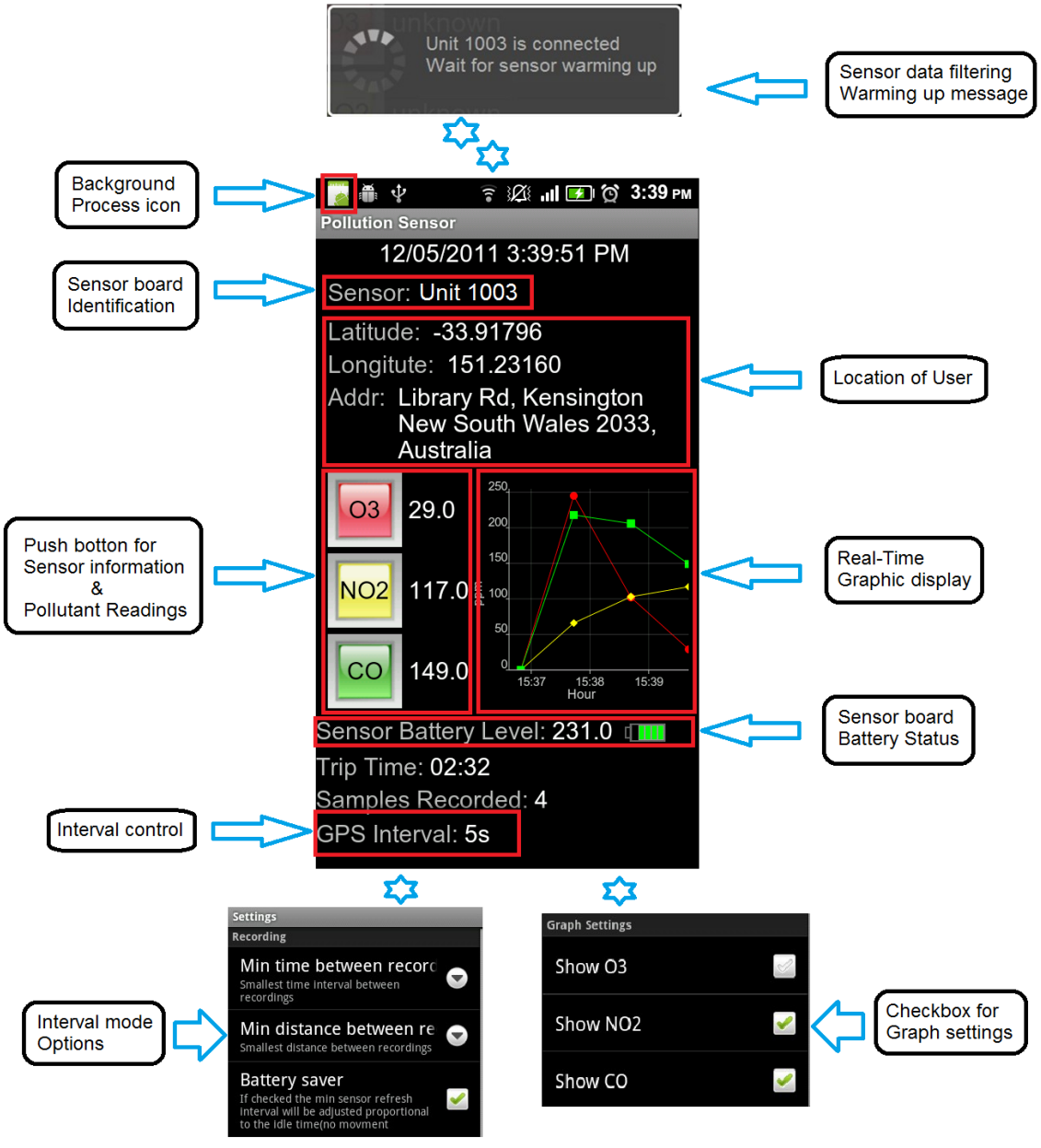

Figure 1: Overall project objective

- Provide more desired information inside Android application

    1) Pushbuttons
    2) Spinners
    3) Amount of text view information
    4) Checkboxes

- Create more functionality inside the application, so that user may have some control on the application during running.

    1) Graphical display of data Readings VS Time or Distance.
    2) GPS interval control options.
    3) Battery levels of sensing device

- Solve the existing 'Mobile sensor device' data filtering problem.
- Fix GPS location tracking termination problem after application exists.
- Achieve running the application as a background process.
- Solve the higher power consumption problem via different mode operation.
- Fix the Bluetooth search problem after connection lost between device and smart phone, and achieve the auto-connection method use self-learning program.
- Optimize the application.
- Field test.
- Help with the calibration of sensors.

1.2 Our role

In the year 2011, "Haze Watch System" is being break down into small pieces of work for modification and improvements. Works on mobile sensing unit are:

- Calibration of the sensor board sensors.
- Communication between hardware device and the Smartphone.

✧ Our roles:

- Smartphone application interface improvements for pollution monitoring system.

2. Background

2.1 Current monitoring method

In New South Wales the Department of Environment, Climate Change and Water (DECCW) has the responsibility of monitoring the air quality of NSW. There are only 14 static monitoring stations covering the entire city of Sydney, plus an additional 6 covering the rest of the state. The monitoring stations measure the pollutants:

1) Ozone
2) Carbon monoxide
3) NO2
4) SO2
5) PM (fine particulate matter)

The pollutant readings are displayed in their standard unit PPM and update on their webpage on an hourly basis.

The problem with the 14 monitoring site is when there is a need to determine pollution levels for any region in between these fixed locations, either a complex algorithms or inaccurate measures are used. This can lead to applied uses of unreliable data.

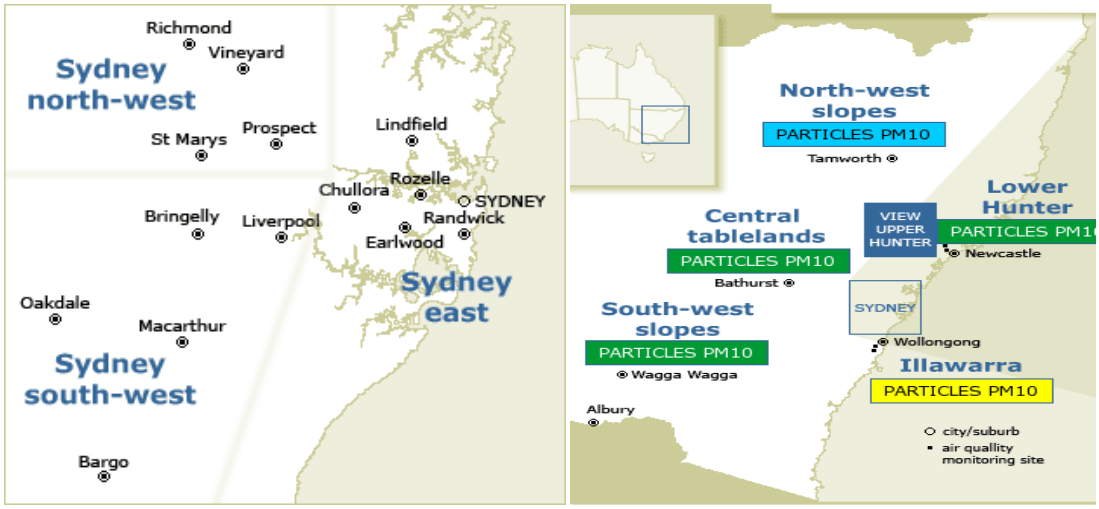

Figure 2: DECCW monitoring station map[3]

In contrast to monitoring methods using stationary stations, a detailed picture based on real-time data from mobile sensors for the entire populated area would offer major benefits to air quality control.

2.2 Android operation system

Android is a software stack for mobile devices that includes an operating system, middleware and key applications which is based on the Linux kernel for mobile operating system.

2.2.1 Features

- **Application framework** enabling reuse and replacement of components
- **Dalvik virtual machine** optimized for mobile devices
- **Integrated browser** based on the open source WebKit engine
- **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQLite** for structured data storage
- **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator, SDK tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE **[***4.What is Android|Android developers, 2011*]

The following diagram shows the major components of the Android operating system.

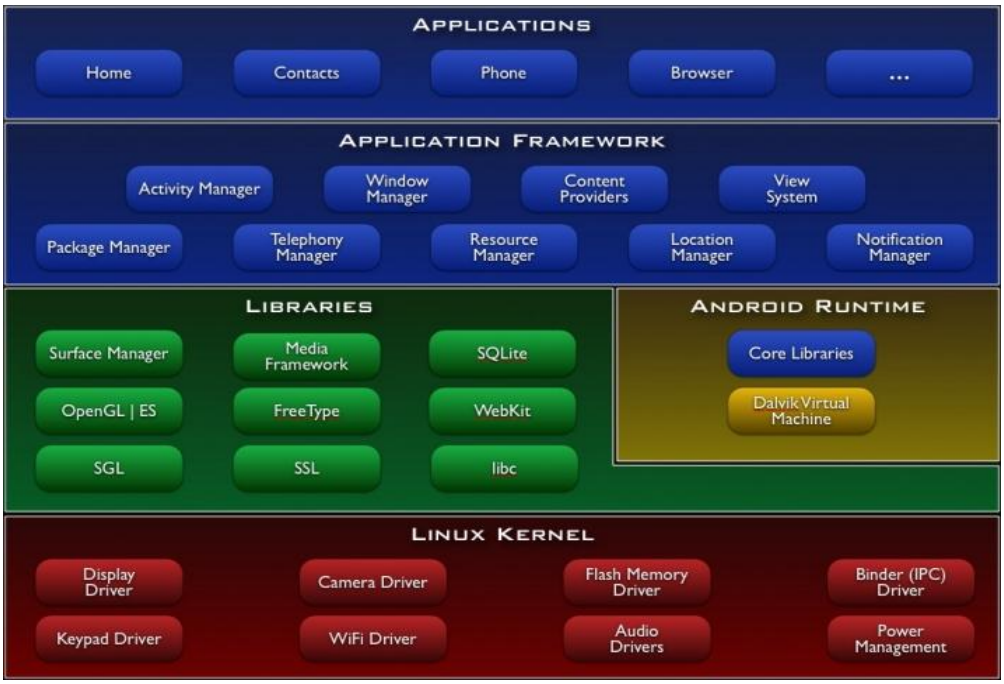

Figure 3: Architecture of Android operating system[*4*]

2.2.2 Application fundamentals

Android applications are written in the Java programming language. The Android SDK tools compile the code along with any data and resource files into an *Android package*, an archive file with an .apk suffix. All the code in a single .apk file is considered to be one application and is the file that Android-powered devices use to install the application. Once installed on a device, each Android application lives in its own security sandbox:

In this way, the Android system implements the *principle of least privilege*. That is, each application, by default, has access only to the components that it requires to do its work and no more. This creates a very secure environment in which an application cannot access parts of the system for which it is not given permission.

2.2.3 Application components

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter your application.

Here are four types of components:

- **Activities** represents each single screen with a user interface
- **Services** is a component that runs in the background to perform long-running operations or to perform work for remote processes
- **Content providers** manages a shared set of application data
- **Broadcast receivers** is a component that responds to system-wide broadcast announcements

Each component performs a different role in the overall application behaviour, and each one can be activated individually (even by other applications).[*5. Application fundamentals|Android developers, 2011*]

2.2.4 Advantages

Android mobile operating system has THREE advantages compare with other mobile system:

- **Openness:** Google and other members of the Open Handset Alliance which includes HTC, T-mobile, Motorola, Samsung and more than 30 other companies collaborated on Android's development and release. Google works with operators, manufacturers, developers and other related fields as deeper cooperator partner, to create a totally open and integrity mobile system for the mobile terminal. Which for our design, it means we can control the Bluetooth and communicate with our device.

- **Simple communication to Network:** The application can easily implant HTML, JavaScript and so on, which for our design, it means we can control the 3G/Wifi and communicate with our server.
- **Multi-Tasks operation:** Android is a completely multi-tasks environment, which for our design, it means we can make the application background process, it will detailed described at section 4.7

[*6. Fengsheng Yang, Oct.2010*]

As the phone we get is HTC G3 with system version android 2.2 Froyo, our design is based on this system version which already can achieve the Bluetooth and 3G connection as we need .

3. Haze watch system

3.1 System overview

The Haze Watch system consists of three principal components which is the sensor units, the server database and the user applications. Figure 4 shows the Haze watch system overview, which image says this concept.

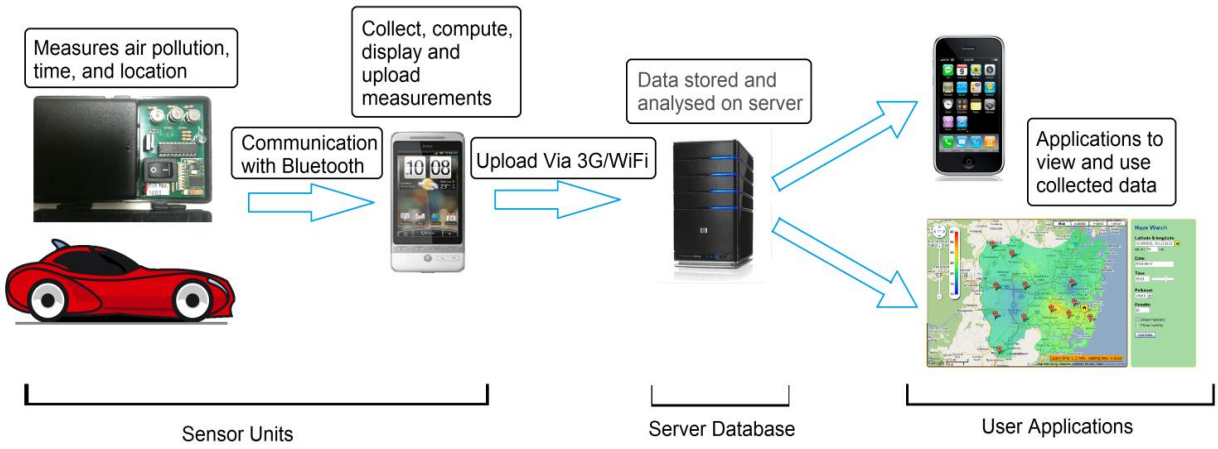

Figure 4: Haze watch system overview

Radio's traffic jam detection system is very practical and efficient warning system for the audiences, which it use participatory sensing methodologies [*7. J Burke et al, Participatory Sensing, ACM, 2006*], all the audiences can become the detector, the more detectors inform the traffic conditions the higher quality that traffic report will

have. It's the same idea of our plan, we want to have some "collectors" which carrying the hardware device that can collect air pollutant values, and many more "users" who can view the data with various users applications, a "collector" can also be a "user".

Essentially, our "collector" can be any vehicle, especially those take most of the day on the road and cover large area of city, for example the public transport are very valuable, we may make each bus take one sensor device and communicate with the Smartphone which drivers have, as the coverage and frequency of the public transport have, the data upload via smartphone can be received by our server to make a great pollution level view.

"User" don't need to carry a sensor device, they may assess their exposure to pollutants by using our user application either on the smartphone or webpage.[*8. N.Youdale, UNSW undergraduate Thesis, 2010*]

3.2 Sensor units

The sensor units consist of two parts which is a hardware sensing device shown in figure 5 and an Android phone application.

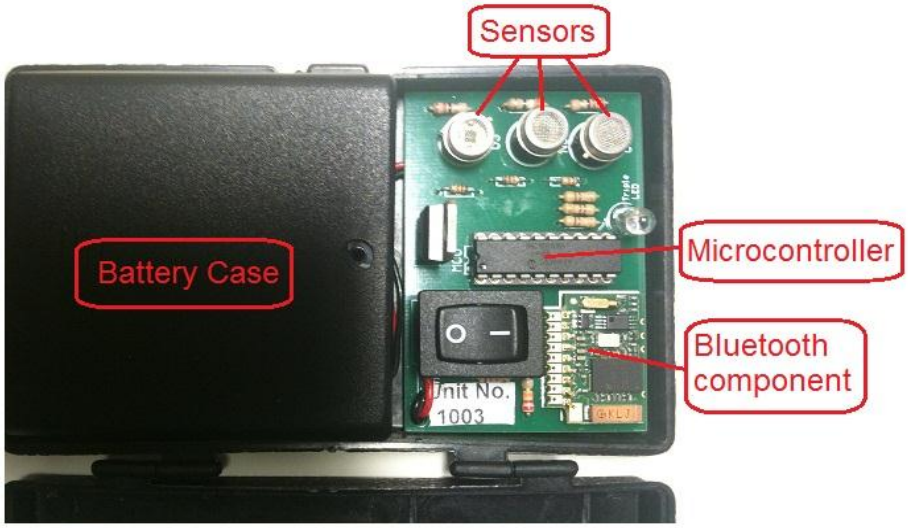

Figure 5: Sensing device overview

The PCB board was designed and built by last year thesis people. It currently has three gas sensors on the board which are:

- Carbon monoxide (CO)
- Nitrogen dioxide (NO_2)
- Ozone (O_3)

Three gas sensors receive pollutant values and forward to microcontroller, microcontroller convert and send those values within the format shown in figure 6, the communication between sensing device and android phone using Bluetooth components which exist on both.

Because of the sensing device is designed as simple and low-cost as possible, it's better to make all the precise algorithm inside the Smartphone as it has more built-in functions, the android phone application receive, calculate and display the pollutant samples, stamp them with date, time, and location then upload to the server database.

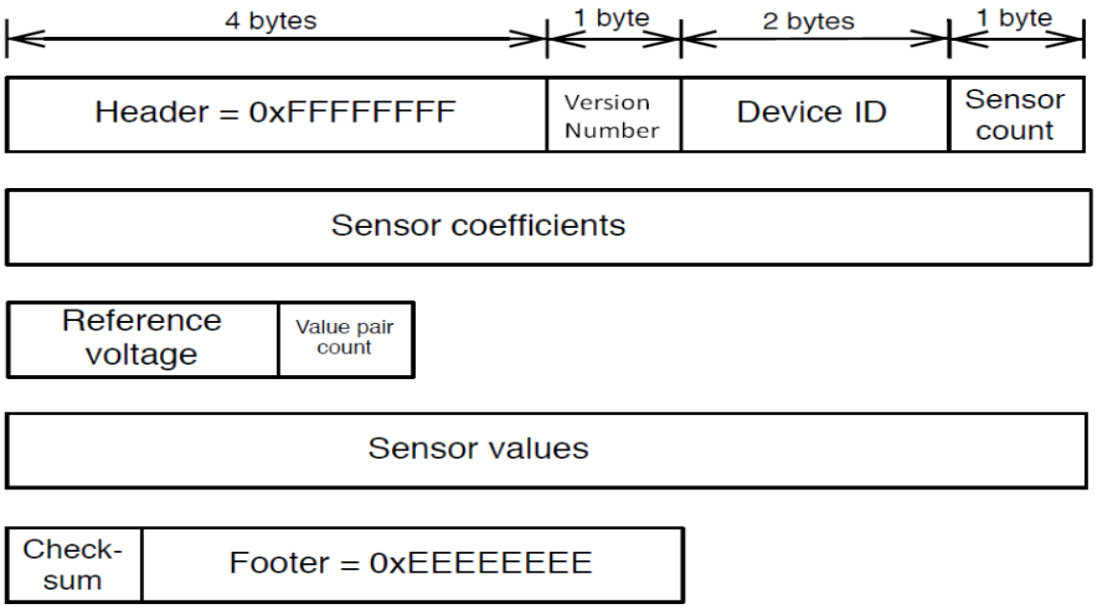

Figure 6: Message format for Bluetooth communication[*8*]

3.3 Server database

The server database is a very important part of our design, the main functions of server database for now are:

- Receive and record the pollution samples from sensor units inside a database
- Export the data to the user application when they require
- Host the project website and user applications

3.4 User applications

The user applications currently have two part involved, which are

- **Project website**(http://pollution.ee.unsw.edu.au/): which includes a pollution map based on the Google map, It allows users to view the pollution level in a colored graphical map shown in figure 7 and check at the specific point for the detailed information about each pollutant.

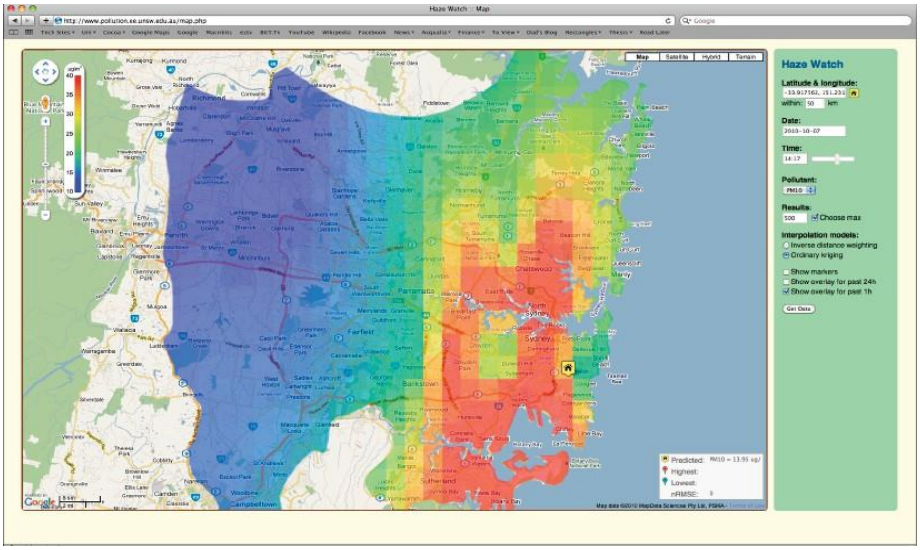

Figure 7: Screenshot of pollution map at project website

- **iPhone personal exposure monitor**: which makes the user can track their own location and display it with pollution level data received from server database, It

also can compute and draw a graphical chart to display the users exposure to pollutants over time, as shown in figure 8.

Figure 8: Screenshot of personal pollution exposure application running on an iPhone. Pictured is a graph of PM 2.5, with the guideline value (WHO) plotted in line[*8*]

3.5 Previous android interface and evaluations

3.5.1 Previous android interface

The android part application was written by last year thesis people is very simple, including 2 separate screens to complete the basic connection and monitoring function which are list blow.

- **Device selection**: Figure 9 shows the initial screen when the application is launched. The user can either choose to select the device from a built-in list of all devices or can perform a quick Bluetooth scan of all transmitting devices within range of Bluetooth detection. It also consists of a text field where the user can manually type in the unique MAC address of device to connect in case of Bluetooth detection failed.

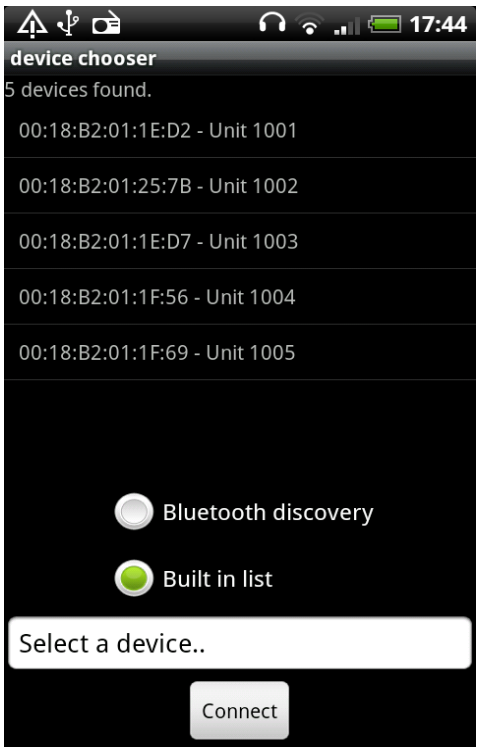

Figure 9: Screenshot of Device selection

- **Main page**: Figure 10 shows the main page of application when sensing device successfully paired with android application, it contains:
  - Date and time when pollution samples received
  - Current location's longitude and latitude
  - Most recently pollutant readings
  - Connection status with sensing device

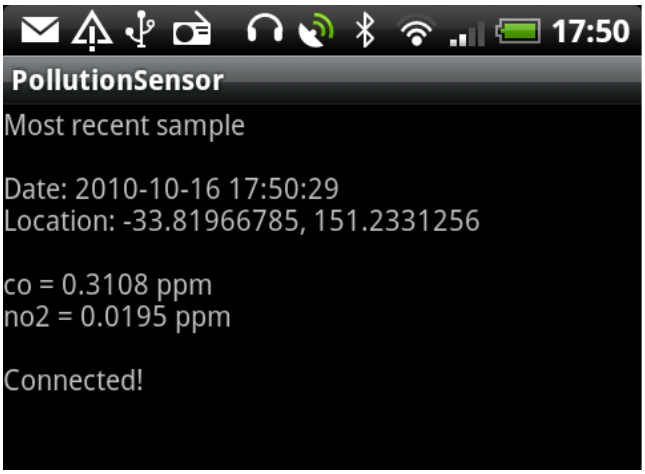

Figure 10: Screenshot of previous android application main page

During this state, as the sensing device will periodically send the packets which contain 10 sets of pollutant values every 30 seconds, the Bluetooth connection sleep for a short time after latest message received, and reconnect the device. Android phone application extracts all the information inside the packet, calculate the equivalent pollution readings in PPM(parts per million) according to the Calibration equation(for now the calibration haven't finished yet, the data shown on the screen maybe a wrong result, but it's very simple to fix this problem as the people who doing calibration now can get the right equation), application encode the data with date, time, location into a XML file and upload it to the server, It also write them into a CSV(comma separated value) file in order to inspection or analysis.

3.5.2 Evaluations

The previous android application was designed just for the testing purpose, it doesn't intuitive for the user, on the user point of view, a great application should be

- **Simple**
- **High quality of understandable information**
- **Great user friendly level**

Follow those principles, we decide to improve the application by modify its lacks listed blow and create more features:

- **Basic and unintuitive**

----Lack of desired and understandable information: Like exactly address where the longitude and latitude mean. The sensor unit number which user connect to

----Basic Text View method: No graphic display which is a great way to make user intuitional check the pollution level.

- **Controllability**

----Limit functionality: User can't control any performance of the application now

----Readings overwrite issue: No history or track of previous pollutants can be shown to the user.

----Program close issue: For now the application can't turn off by normal way, if force to close will cause data lost problem.

- **GPS/Bluetooth termination problem**

After the application exits, the GPS and Bluetooth module which activated by application doesn't turn off which will trigger some inconvenient for user to manually close them and battery consumption.

- **GPS LOST** (Focused by other group)

Those are the lacks of pervious application, the improvement and creation will be specific explain blow.

4.Android phone application

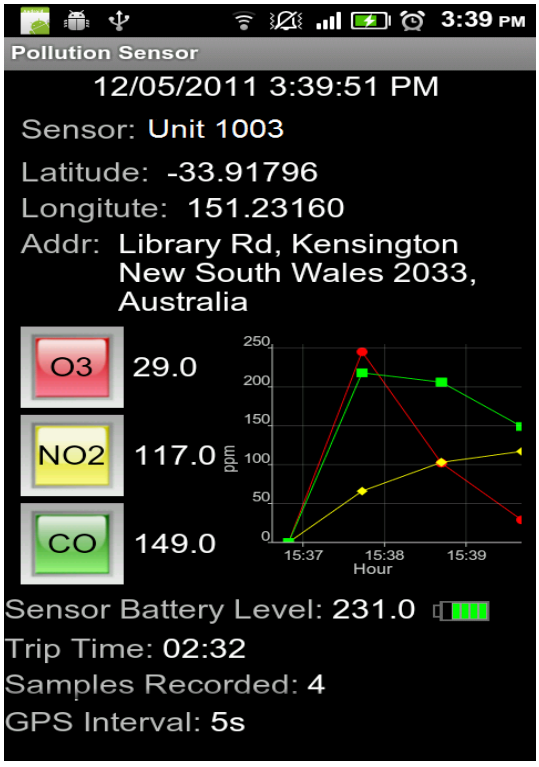

Figure 11: Overall New Interface

Figure 11 shows the overall New interface for our application, all the text view information in the previous interface is still kept, what we add from top is:

- Sensor ID: User may want to know which sensing device is connected to the application. The sensor board contains a Bluetooth module that each one have a unique MAC address, we can name and recognize the device due to this characteristic.
- Current location: Displayed in longitude latitude and detailed explanation on where does this longitude and latitude means in human sense of address, this function will specifically described in section 4.2
- Push buttons and Pollutant readings in PPM unit which will specifically described in section 4.3
- Real-time Graphical display which will specifically described in section 4.4
- Sensor battery level in volts, this function will specifically described in section 4.5
- Trip time: Record the overall application running time
- Sample numbers: Record how many samples received from sensing device to Smartphone
- GPS interval control which will specifically described in section 4.7

4.1 Sensor data filtering

By testing on the sensor board performance, we realized the sensors being used in the sensor board requires certain amount of time to warm up before it starts taking contributable measurements, Which means the first few minutes reading will make no contribution to the pollution monitoring system. After the field test we found that the sensor usually takes 3 to 5 minutes to get to the reliable and stable values.

Therefore filtering those data is necessary. And this explains why a notification of warming up message shown in figure 12 is required.

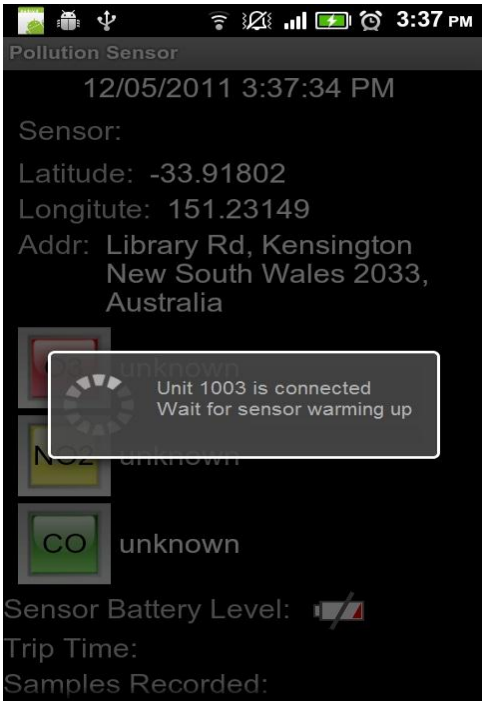

Figure 12: Screenshot of sensor warming up notification message

A simple detection algorithm is done along the application running while extract incoming message:

- Inside the mobile sensor board, we modified the incoming message produced by the microcontroller simply assigning same negative values into the field called sensor value for first 1 to 2 minutes since sensor board been launched.(the overall format of the massage is in section 3.2)
- Inside Android phone application, once the Smartphone detects the entire sensor values they obtained from the incoming message are coincidently same negative value, the warm up page will show up.

4.2 Location estimation

Figure 13: Comparison between pervious and current location display

In previous application interface shown on the left of figure 13, only longitude and latitude of a location being displayed, for most of the users those two numbers may mean nothing, Even though the two values itself are accurate and valuable, but it's better to let the user know exactly address information contains street name, suburb, state, and postcode and so on.

Figure 14: Android location estimation overall method

The overall logic of obtaining fast and accurate location response is introduced below, figure 14 shows the comparison between three different kind of GPS techniques:

Civil GPS is the most basic positioning topology for every positioning system, but due to its relatively service positioning accuracy is not very high which will lead the error estimation can be up to 200m maximum, this error will affect the location seriously in a way that user may see a wrong location being displayed and confuse themselves.

A normal way deal with this error is by using DGPS(differential GPS), which will calculate the position not only using the satellites but also using differential base stations, the positioning accuracy can now be enhanced to 5m maximum.

In situations like indoor or low signal regions, DGPS and GPS cost around 2~3mins to initialize themselves, with the help of AGPS (assisted GPS) it solved the weaker signal positioning problem.

The AGPS's logic is quite similar to DGPS, instead of using satellites and differential base stations only, it adding another base station which the phone operators built around us, calculation and the location information will be received via GSM network, therefore, Combine all the GPS techniques can provide user a more reliable and precise location estimation. [*9. GPS-Baidu encyclopedia, 2011*]

After confirm the longitude and latitude, the application program will forward them to Google map server, put longitude and latitude into a “Geopoint” which is a function Google map provide to implement the inverse solution of address, we also add a function named "locationlistener" to estimate the GPS coordinate's change, as soon as the coordinate changes, Locationlistener will automatically take out the new longitude and latitude into “Geopoint” to get the most recently address. As the address does not related to the sensor data rate, we make an individual display system for the Location function, So that users can see the real time location change and also the server will get the most reliable location information.

4.3 Sensor information push buttons

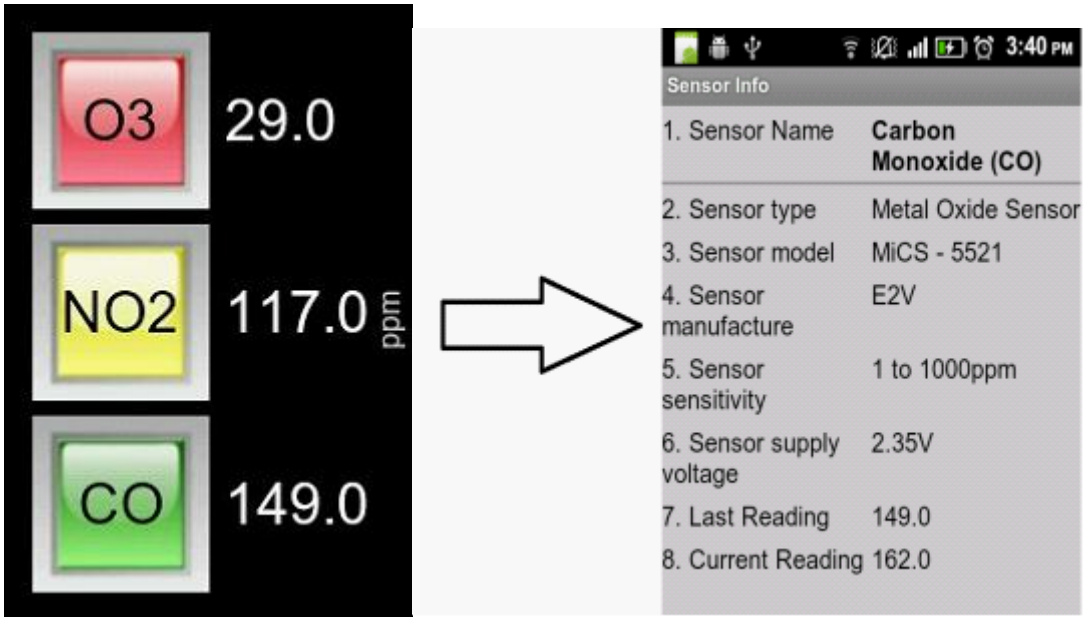

Figure 15: Screenshot of Push buttons and Sensor Information page

Considering users may wonder what type of sensor is being used in the sensing device and the performance of the sensor, we include a push buttons field in our design for user to click on and check the sensor detailed information. See figure 15 for what is being displayed.

Apart from what is being displayed currently, we will add the last calibration date and duration consist in sensor information page. The reason calibration is a very important data needs to be display is the low-cost sensors we are using each have a slightly different performance characteristic, meaning that the conversion equation will vary slightly in each sensor unit after the long time functioning

4.4 Real-time Graphical display

To be a high user friendly application of monitoring air pollution, it's better to make the Readings more visualize and lively, so create an open and shut view of readings can be a great idea. Here comes an idea of display a real-time running graph.

The graph works as a visualized process for the user to track pollutant readings along their trip. AS the graphical display is synchronized with the received data, users can easily see the changes of the readings VS time by looking at the graph. Consider of the Limitation of the space of main page and screen size, the graphical display may not be very clear to check at the specific value of sensor, so we create a link with the graph of main page, click on it, application will display the graph in a full screen size, shown on the right of figure 16, the graph can be shown both in portrait and landscape orientation using gravity induction sensor towards user's move. Also the graph itself can be enlarged and minimized as well.

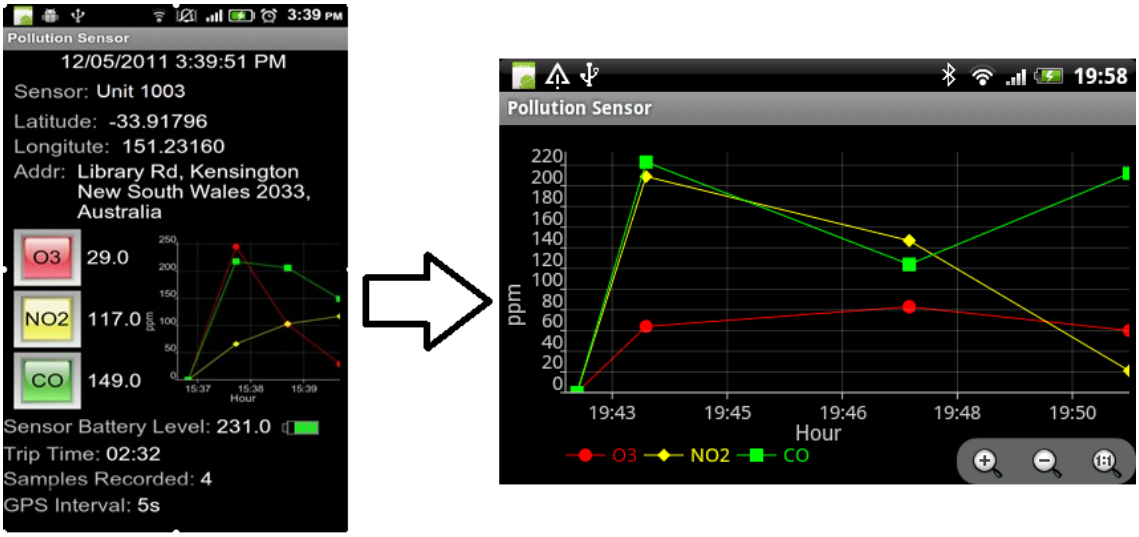

Figure 16: Screenshot of Graphical Display

More features inside the graphical display is you can click on each checkbox at the graph settings by only display the desired pollutant plot which shown in figure 17. This is the preparation for the future develops which we can easily check at the specific pollutant out of many others.

Figure 17: Screenshot of Graph settings and Specific pollutant plot

4.5 Battery level

On a user point of view, they would like to make sure their sensor board doesn't run out of battery along their trip to gathering readings. But it is hard to actually see from the sensor board whether the battery should be changed or not, so it is necessary to display the sensor board battery consuming conditions on the Smartphone.

The discharge curve blow shows the calibration of a rechargeable battery done by previous year thesis people.

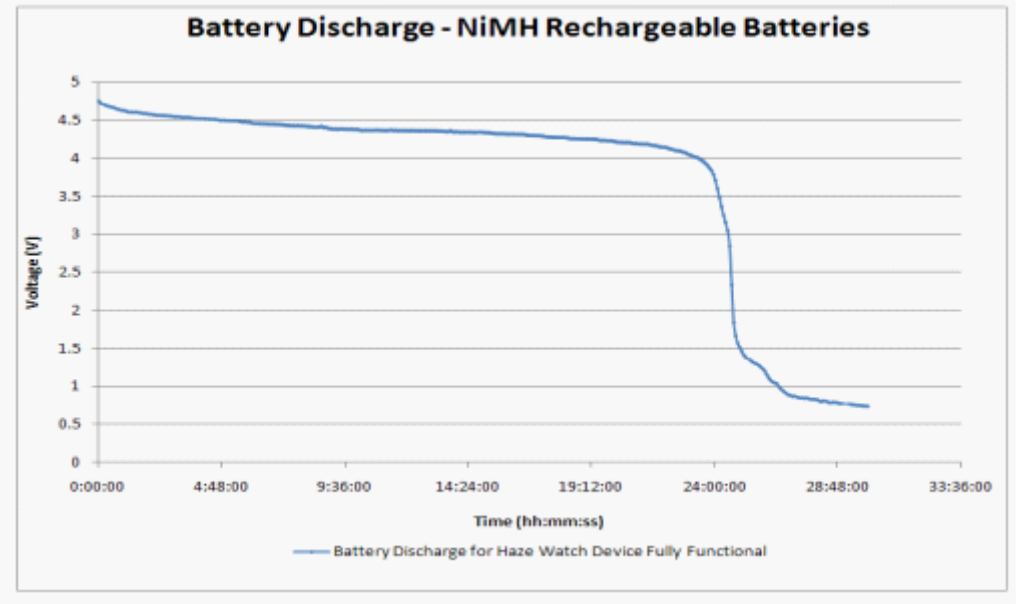

Figure 18: Battery Discharge curve

We may easily recognize that the voltage of the battery slowly drops off during the first 22 hours of operation. During this period the wireless sensor board was completely functional. Similar to all battery discharge curves there is a sudden exponential drop in supply voltage, the supply voltage dropped off suddenly at approximately 23 hours of operation and during this time, the Bluetooth stopped functioning and the wireless sensor board was inoperable.

As the microcontroller has the ability to send hex numbers from 0 to 255 to represent the battery level, using a simply algorithm show below, we can get the relationship between voltage reading in volts and Hex number representation.

(Reference voltage (3.3) /255) * HEX obtained = Voltage of battery

We create a new field inside the protocol which is called “battery status”, it will send to device all the time to tell the battery level value, we also include a field in our application program to take out the “battery status” and apply the algorithm above to calculate the remained battery power of sensor device.

Figure 19: Screenshots of battery level display icon

The battery level display icon was chosen like the way it shows in the figure. There are 5 grids inside this icon. The first 4 grids in Green stand for 80% of battery, device function normally. Once the display only shows one grid in red. It is a reminder for user that the battery should be changed. The value currently displayed for the battery level is its hex representation. Later on it will be changed to display the percentage of battery remains.

4.6 Power consumption

Due to communication between phone, device and server, Bluetooth, Mobile network and GPS connection always turns on once while application is launched, this is the main concern of power dissipation of the Smartphone in a result of android phone's battery drain faster than usual.

On the user side they may feel unsatisfied about the application, as power efficiency factor is an important issue we should considered.

So we looked back to whole procedure of this application, the power consuming problem inside the application is divided into four main parts on the android phone application side:

- Bluetooth connection between device and phone.
- GPS communication power consumption of phone.
- Frequently refreshing displayed data in the graph of the application.

- Uploading package via 3G/Wi-Fi network to server.

As sensor board device sent one set of data per 5seconds to the Smartphone, we need the Bluetooth communication between phone and device all the time along the whole trip of gathering measurements. It's the same reason for GPS communication as well, because of the location's accuracy is a very important factor of our application, we want the exactly location all the time, which means we need to get the GPS communication keep on tracking for every slight user movement.

We create a function that under some certain conditions user can actually control the rate of display data on the graph as well the package forwarding rate to server. A slower rate on both sides can save the android phone's battery, decrease the system memory and 3G/WI-Fi network usage.

4.7 GPS interval control and Operation modes

4.7.1 Normal mode

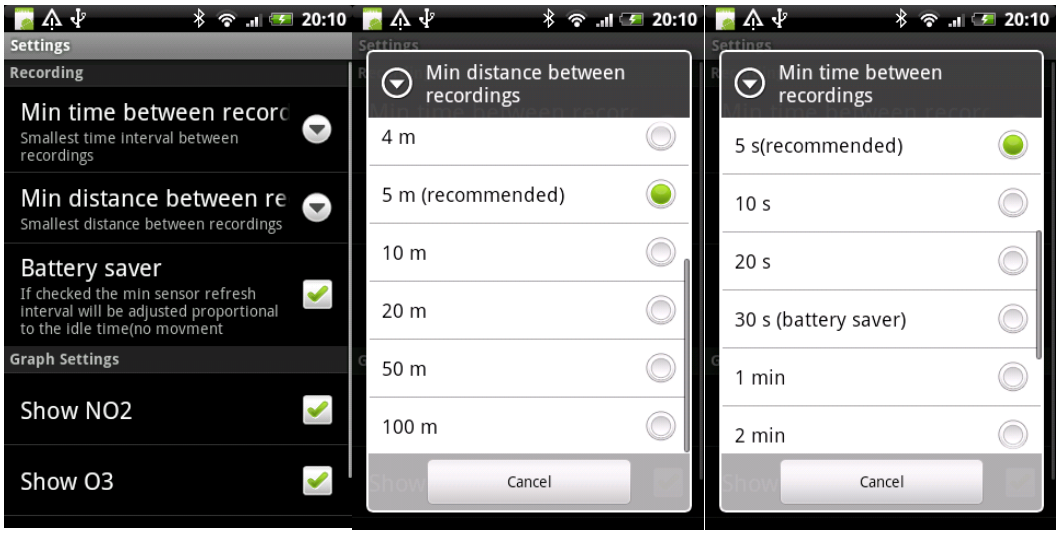

Figure 20: Screenshots of Interval control setting

There are three general INTERVAL control options, click at each spinner, a selection menu will appear and user can choose to control time interval or distance interval or simultaneously control both by select different options so that varies the graphical plot of the sensor readings and package forwarding rate to server.

Here we given three examples of the figure that plots out the different options user may make on the interval selection.

Let's go through them one at a time.

The figure 21 shows data displayed only depends on time traveled.

This setting is under the choice of distance interval equal zero and time interval varies.

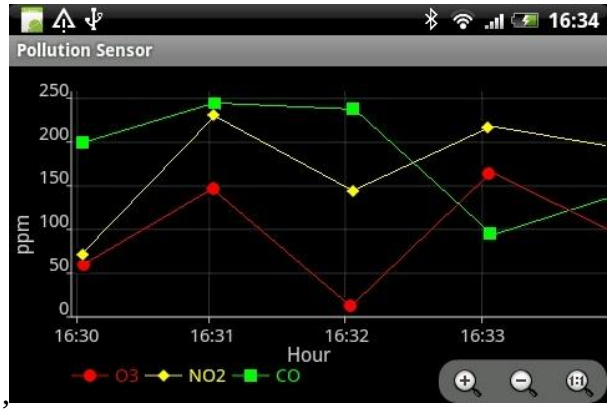

Figure 21: Example of Time interval control

Once you decide your time interval. The figure will refresh readings and application will upload to server based on user time settings, which in the figure 21 here is 1mins at a time.

The figure 22 is when user only controls the distance interval.

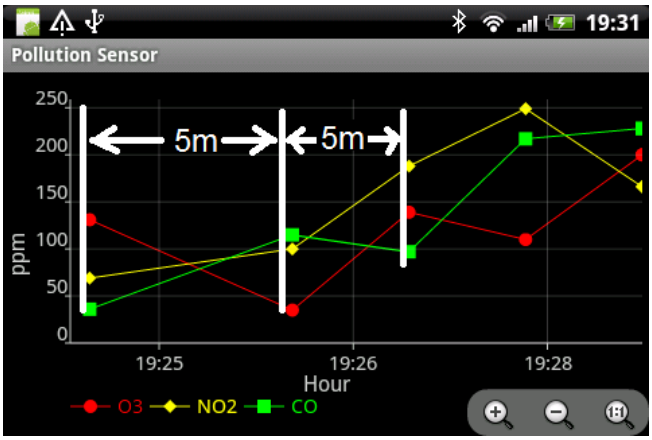

Figure 22: Example of Distance interval control

The logic of distance interval control is same as time interval control, the application will only process the data when the distance satisfy the choice which user made

The figure 23 shows a simultaneously control both time and distance interval plot.

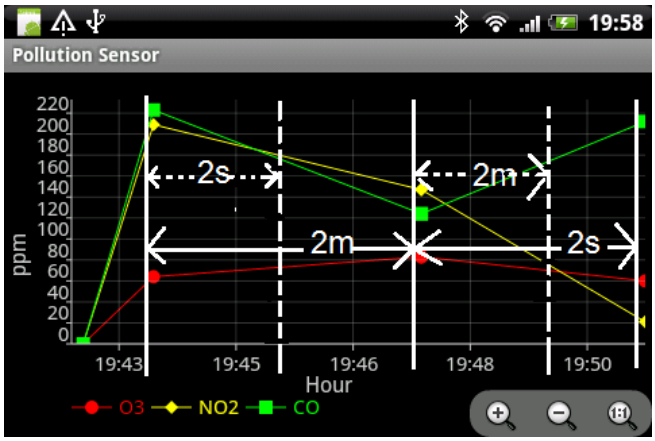

Figure 23: Example of Time and Distance interval control

Assume user set the time and distance interval to 2seconds and 2meters, as we define that two types of interval control are equal priority, so data will displayed in the graph and uploading data while both of the conditions satisfied.

We can see this more clearly in the figure 23. The readings will not show if only time interval or distance interval satisfy the requirement.

4.7.2 Battery saver mode

We already can control the time and distance interval for the purpose of reduce power consumption of android phone, it's unpractical that a user want to change the time and distance interval during the trip, so invent an automation mode which can adjust the time and distance interval depends on the device's current condition to save the power is very important for our application.

The reason we making this mode doing automatic activation is due to its unnecessary to refresh the figure readings and forwarding data to server too frequently when the user is travelling quite slow, the pollutant readings user may receive should almost the same, in this case we still acquire data from the sensing device frequently as usual, all the data will record inside the android phone's database SQLite, the two changes in the application, one is graphical plot will slows down itself, second is the package forwarding rate to server with the same location is not really necessary upload too frequently for the server side.

4.7.3 Alarm mode

Alarm mode is also an automation mode which been designed to face the special situation.

If any of the sensor reading shows an extremely unusual value which means the reading is above the standard value (refer to table1: WHO guidelines of pollutant values in chapter7.1) inside normal air, the application will first compare the next incoming data, to make sure the previous value was not an error and then make decision on whether alarm mode should be activate or not.

Once the alarm mode is activated, readings will be displayed and uploaded at a maximum frequency rate, as for this situation we are more likely willing to monitor pollutant data readings more frequently for further purpose.

4.7.4 Overall algorithm

The overall algorithm of the mode operation is shown in figure 24.

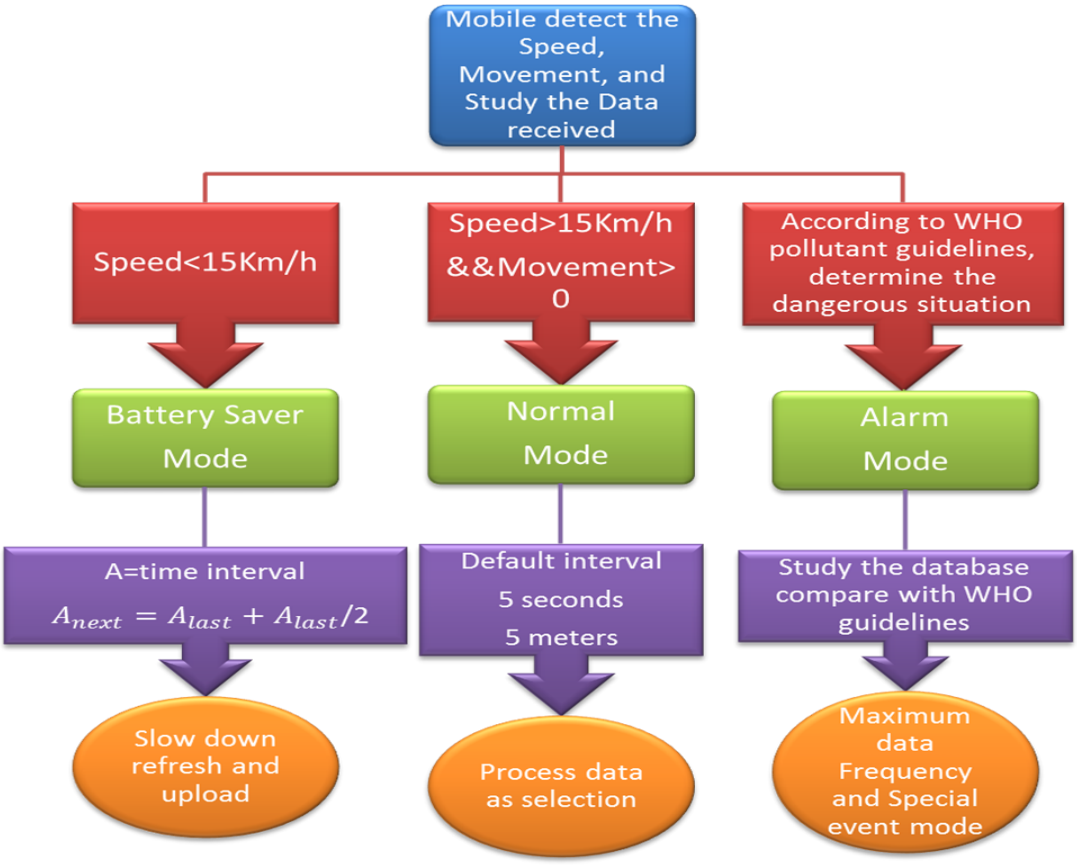

Figure 24: Overall algorithm of Automation mode

First of all, application will detect the Speed and movement of user carrying the device, according to the current Latitude and Longitude we obtained from GPS, to make sure us getting the correct speed for process this algorithm.

We applied the "haversine" formula to calculate distances between the two points on a sphere from their longitudes and latitudes. [*10. Chris Veness, 2002~2007*]

R = earth's radius (mean radius = 6,371km)

$\Delta lat = lat_2 - lat_1$

$\Delta long = long_2 - long_1$

Haversine formula:

$a = \sin^2(\Delta lat/2) + \cos(lat_1).\cos(lat_2).\sin^2(\Delta long/2)$

$c = 2.\text{atan2}(\sqrt{a}, \sqrt{(1-a)})$

$d = R.c$

(Note that angles need to be in radians to pass to trig functional: degree/57.29577951).

As the time is also a known value we can use, we may also calculate the speed apply to

Speed (V) = Total Distance Traveled / Total Time Taken = d/t

Considering GPS's accuracy problem, we make a comparison of both values to make sure what the speed of device is.

Then we make a choice here, if the Speed is slower than 15km/h which is a normal person fast walking speed, the application operates at battery saver mode, and if Speed exceeds 15km/h which means you are moving relatively fast, the program itself will jump to normal mode operate at the default GPS interval option which is 5seconds and 5meters.

In the battery saver mode the application will proportional adjust the data display and upload rate by increase the previous time interval follow next interval= current

interval + current interval/2. This time interval will change as long as it detects a fast speed movement or a condition satisfying alarm mode.

4.7.5 Field test results of operation modes

After finish the programming of operation modes, field test is the best way to inspect the performance. As the operating modes are created to decrease the power consumption of smartphone, comparison between of usage will show us the difference between Normal mode and Battery Saver mode.

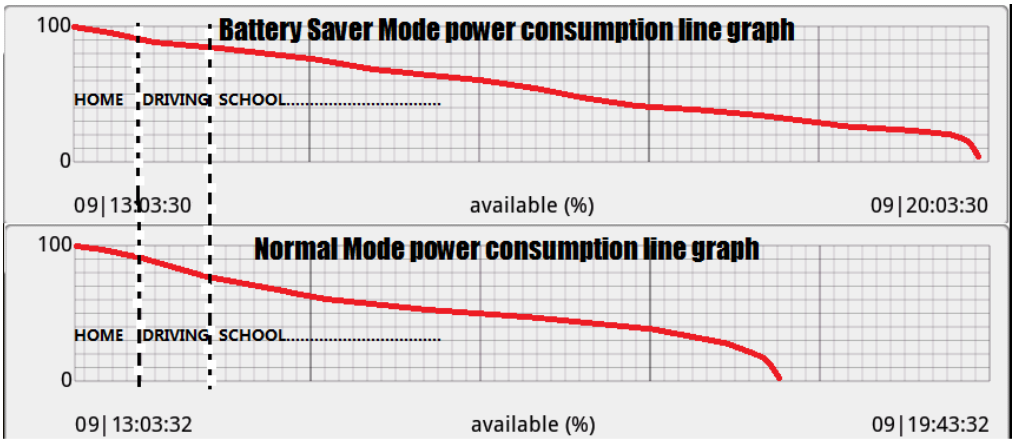

Figure 25: Power consumption between Battery Saver mode and Normal mode

As shown on figure 25, the test is under same situation which is from my home to school, between dotted lines is under driving, and other times are always at state of rest or slow movement, device and smart phone turn on at almost same time running different mode of operation. From the figure, we can see battery saver mode (approximately 7 hours) significantly improve the operating time compared with normal mode (approximately 5.2 hours), and specific compare two modes, we can see a great energy conservation during the driving time, as within the same time battery saver mode only drop 10% instead of 20% power drop caused by normal mode.

4.8 Background process

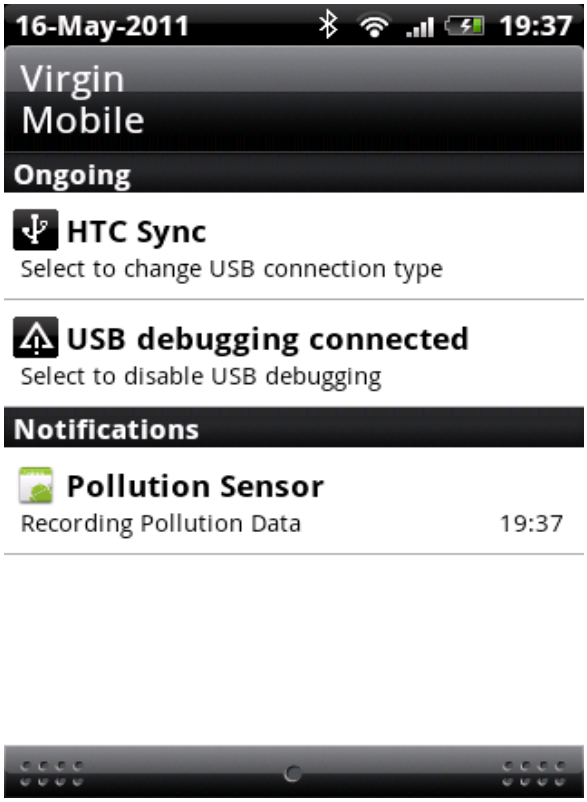

Figure 26: Background process

As we mentioned before, the aim that we developed this application is think everything that a user want, to make simple utility software but do great performance that satisfying all users, therefore a simple click that can leads the application to a background process is very important, which is shown in figure 26.

Android system provide a "service" mechanism to implement the NUI(Non-User Interface) service, which make the application can process in the background, when background process starts, It will only occupy limited system resources with achieved logic until UI(user interface) requests communication. Its life cycle can last as long as the system memory remains. Background process is a stable and efficient function as user expected.

5. Bluetooth Auto-Connection

As the sensing unit consists of a phone and a device, under some conditions the device may turn off or user may take the phone away from the device, but phone application still running, and as the connection lost, it will search and try to connect the device all the time, No matter in background process or battery saver mode, battery will drop or even run out. By the time user realizes it will affect the users plan on taking measurements. As a designer of this smart application, we don't want this to happen. Some algorithms and detection method need to be considered to perfect the application.

5.1 Bluetooth Auto-Connection Algorithm

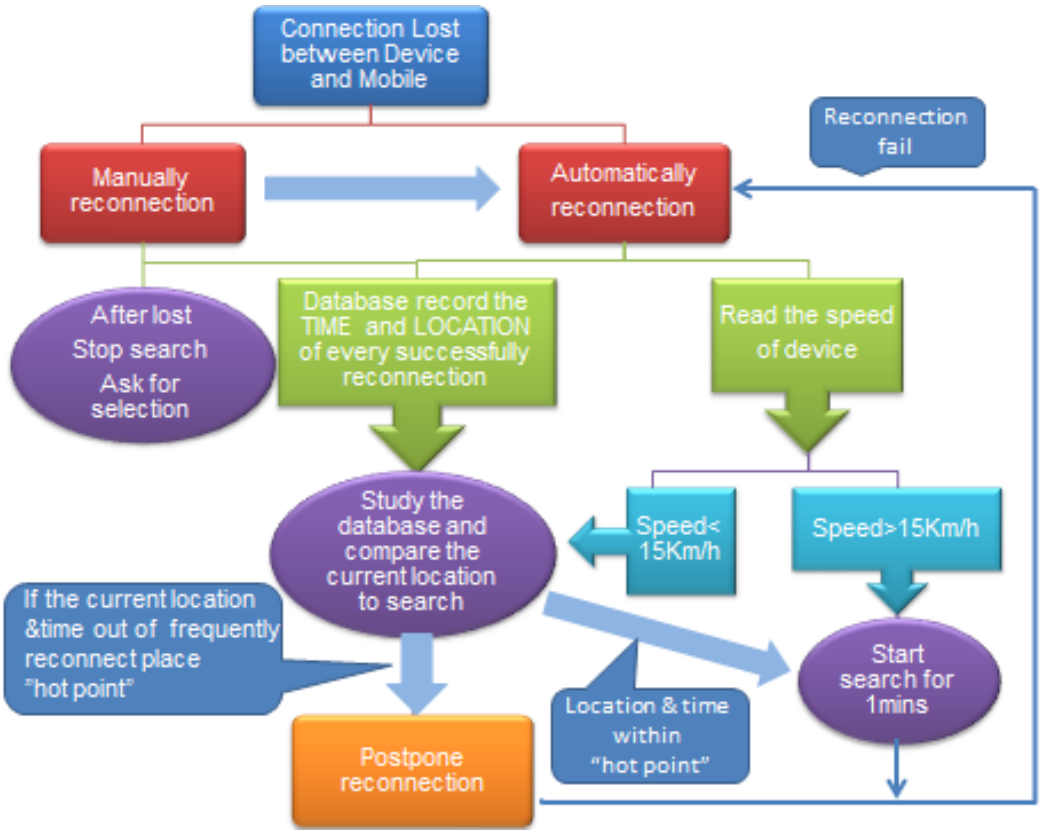

Figure 27: Bluetooth auto-connection algorithm structure

Figure 27 shows the algorithm of Bluetooth auto-connection, when the connection between device and mobile application is lost, application should detect the disconnection, then pop up a noticeboard can let user to select either "connect now"

or just “dismiss” the reconnection, the selection “connect now” will make the application try to connect the device use the same unique identifier number as last time successfully connection, this is designed for user to immediately connect the device when the device still within the mobile’s Bluetooth connectable area, application will record the location and time of the successfully reconnection which used for the further development. If the user select “dismiss” or just ignore the noticeboard for more than 1 minute, application will operate in the automatically reconnection method.

When the automation method launch, application will detect the speed of the mobile which we get before (refer to *4.7.4 Overall algorithm*), and study the database to compare the location. if the speed from GPS returned and application calculated value are all exceed 15KM/h, as the device always paired with the car, we assume the situation now is accidently connection lost and device should still inside the mobile’s Bluetooth connectable area, so the application try to connect the device for 1 minutes, after 1 minutes, if the mobile still cannot connect to the device, it will wait for 10 minutes, and back to the first step of automation method, otherwise application will also record the location and time of the successfully reconnection.

Database record the Time and Location of every successfully reconnection point, take a study period of 1weeks, which means every 7 days, the application will study the database, and select the most frequently successfully reconnection point as “hot point”, the method of selection is reading all the location and time, take the adjacent location points (within the 50m area of each other’s) as one point, same as the time, if within 7 days, the location point or time point appears more than 5 times, application will treat it as a “hot point”, and each time application lost connection at “hot point”, Bluetooth will start to connect the device for 1 minute, otherwise the application will postpone and wait for the situation change.

5.2 Data storage

Data storage is the basis of the application program, every system, application should

solve this problem first, data must be stored via suitable way, and can be used and upgraded effective and sample.

5.2.1 Data storage of Android system

5.2.1.1 Introduction of Data storage

There are 4 different ways data stored within Android system, as all of them are privately-owned by their applications, if we want to make use of those data, we must use "Content Providers" which provided by Android system. First is a brief introduction of 4 different data storage ways.

- *Shared Preference*: used to store the data format of "key-value pairs", it is a lightweight key value storage mechanism, only works for basic data type.
- *Files*: the files operate via "FileInputStream" and "FileOutputStream", Inside the Android system, files is owned by one application program, this application program cannot read&write other application program's files.
- *SQLite*: standard database support SQL statement which android system provide.
- *Network*: use network to store and acquire data.

5.2.1.2 Comparison between different data storage

- *Shared Preference* is mainly focus on the storage of system configuration information, like try to keep the graph's checkbox setting of user interface for user next application launch. As Android system interface use "Activity" stack, when the lack of system resources appears, it withdraws some interface, so some operations need to be kept active, in order to show up when the next time activate the operation. One emphasis need to be highlight is we cannot directly share the Preferences' data between few programs.
- *Files* record and store the data via the format of files, when users need those data, they just read those files to get the data. As android system is based on the Linux

Kernel, all the files are within the format of Linux. In the default situation, files are also shared between different programs.

- *Network* store the data in the network, but have the limitation of requiring the device keep the network connection either 3G or Wi-Fi, like we pair the e-mail address with our handsets.
- *SQLite* is an open source ACID (*atomicity, consistency, isolation, durability*)-compliant embedded relational database management system contained in a relatively small (approximate 275 kB) C programming library [*11. Wikipedia, SQLite*].

5.2.2 Android SQLite

After the comparison between 4 different data storage methods of android system, we can easily get the Shared Preferences, Files and Network all work for storing some simple and small data sizes data, if we need to store, manage, upgrade and maintain a large amount of data which required add, read, renew at any time, the methods mentioned before almost cannot fulfill the demands. Google consider those problems and provide the SQL database, which is specialist in the area of dealing with large amount data sizes' data: It is more rational and powerful on the store, manage, maintain and other fields, a lot of programs come with system make use of SQLite database, like "Contacts".

5.2.2.1 Distinctive Features of SQLite

- *Lightweight*: In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application, but an integral part of it [*11. Wikipedia, SQLite*].
- *Independence*: Most SQL database engines are implemented as a separate server process. Programs that want to access the database communicate with the server using some kind of interprocess communication (typically TCP/IP) to send requests to the server and to receive back results. SQLite does not work this way.

With SQLite, the process that wants to access the database reads and writes directly from the database files on disk. There is no intermediary server process. There is no separate server process to install, setup, configure, initialize, manage, and troubleshoot of SQLite database. Programs that use SQLite require no administrative support for setting up the database engine before they are run. Any program that is able to access the disk is able to use a SQLite database. SQLite is the only one that allows multiple applications to access the same database at the same time [*12. SQLite, Distinctive Features Of SQLite*].

- *Isolation*: A SQLite database is a single ordinary disk file that can be located anywhere in the directory hierarchy. If SQLite can read the disk file then it can read anything in the database. If the disk file and its directory are writable, then SQLite can change anything in the database. Database files can easily be copied onto a Micro SD card or other memory card used in mobile phones [*12. SQLite, Distinctive Features Of SQLite*]. As the feature of single database file, it's easier to manage and maintain.
- *Compact*: When optimized for size, the whole SQLite library with everything enabled is less than 350KB in size [*12. SQLite, Distinctive Features Of SQLite*], during the program running on a mobile phone, the less resources it need the faster phone and program is operating.
- *Security*: SQLite database use monopolistic feature and shared lock on the database level to achieve independence transaction processing, this means several progresses can read the data from same database at same time, but only one can write. It's accomplished by the requirement of shared lock before progress write into the database.

5.2.2.2 Implementation of SQLite

General SQLite operations include: create/open/close/delete database, create table, insert/delete/modify data from table, delete specified table and inquire for the specified data from table. In order to manage and maintain the data better, we package

database operations category inherit from “SQLiteOpenHelper” category, insider the structure of “SQLiteOpenHelper” need Context, name of database, “CursorFactory”(normally is null, or just use default database), version number of database(cannot be negative), we also use Content Providers which is the bridge between data storage and search of all the program function, as all the content providers can achieve some common port, include search, add, edit and delete data, applications can use the only “ContentResolver” port to apply the specified Content Provider.

5.3 Test results

5.3.1 Results

Bluetooth auto-connection method is a background process, we cannot directly see what happened to the application, but different Bluetooth reconnection type will cause different power consumption of smartphone, like what shows in the figure 28:

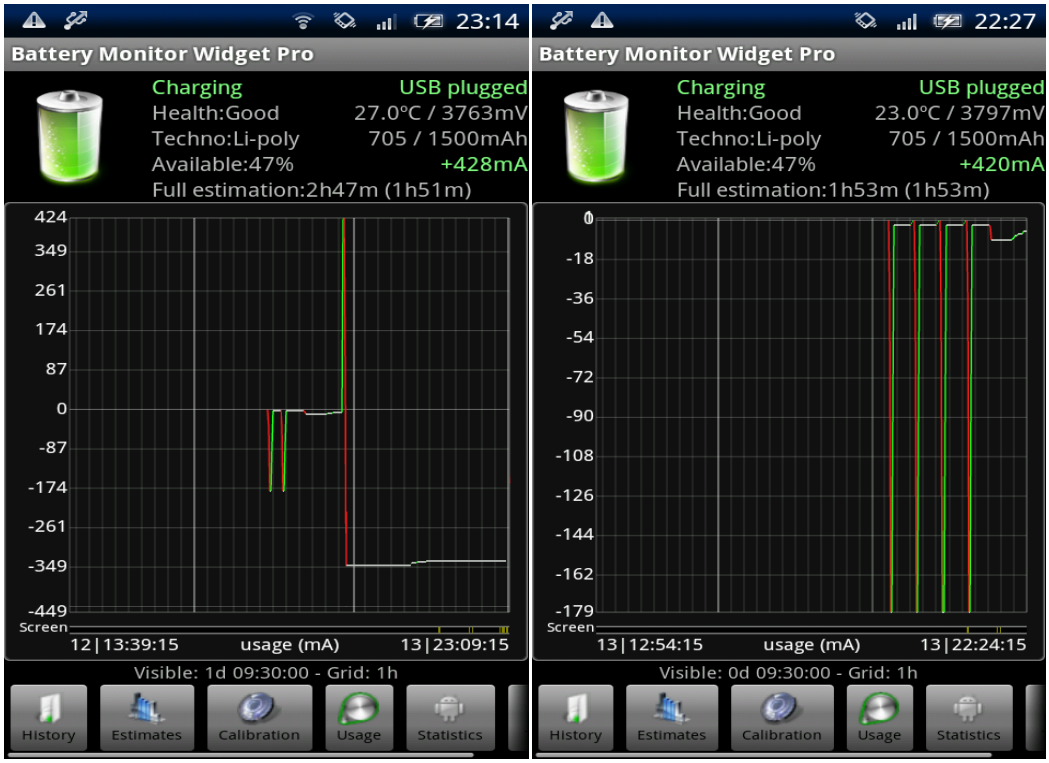

Figure 28: Power consumption comparison between old and new automation method

The left graph shows the old version's power consumption after Bluetooth connection lost compare with right graph shows the automation method's power consumption, X-axis represents time and Y-axis represents battery usage in mA, from left graph, we can see that within old version, application make the Bluetooth search the device all the time, the battery usage keep at -349mA for a long time, it results bigger battery power loss and a certain extent memory footprint. After automation method launch, application start to search one time regarding to the algorithm, as the reconnection failed, application postpone reconnection for 10 minutes and then start over, after 4 times connection tries with -179mA battery usage each time, Bluetooth successfully reconnected.

Compared those two graphs, we can easily see the differences between two methods, automation method significant improve the power consumption of Bluetooth reconnection, and also create a more reliable reconnection method.

5.3.2 Stability

After the field test, the Bluetooth auto-connection method didn't work as well as we expect. There have two problems which first is the sensitivity of "hot point" detection, as the "hot point" of location is determined by the value of longitude and latitude returned from handsets' GPS system, from the research shown in figure we can see that the most precise civil GPS system usually have an error range of 30 meters, most of the mobile will got a much bigger than this, situation will be worse if the handsets within indoor, tunnel, or urban area. This limits the performance of "hot point" reconnection method.

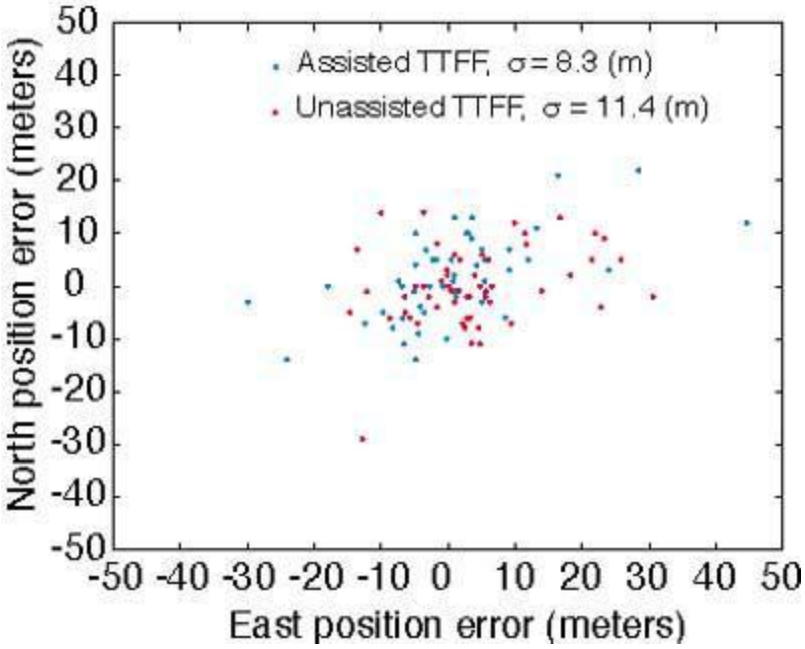

Figure 29: Output position performance of GPS [*13*]

Second is Bluetooth lost detection method, for now, as we put the data received from the device into the database first then application will access the database to get the data, the detection method to decide if the device is still connected is check if there have Header right behind the last Footer inside the database, after the field test, the sensor device may occurs the situation of sending nonsense or chaotic data, application have a poor ability to detect the connection lost due to the data error comes from the device, this limits the performance of whole Bluetooth auto-connection method.

6. Program Optimization

Currently development of Android system most use Java as the program language, as Java belongs to high level language, provide great convenience to the programmer, in that way, Java certainly have higher efficiency than some low level language, as we all don't want to use assembly language, to optimize the program is the best way to enhance the performance of application.

6.1 Elementary of Optimization

To minimize the limitation of application's performance caused by hardware environmental of Java platform, we must use improved design and optimization algorithm during the development stage. Mobile phone manufacturer always fix the mobile software with the phone before the development of J2ME from SUN Corporation, which significantly enhance the mobile's extended function ability by support the third-party software based on J2ME, Android system also use J2ME with Java to develop the application, it requires simpler and more designable program development.

6.1.1 Requirement to be excellent program

1. *Brevity*: brevity is the program with great structure, executable code and excellent design, which will let the program easier to understand and minimize the error.
2. *Readable*: program with more commands and understandable variables make the program readable.
3. *Modularity*: this means we need develop the program like the constituent of universe which is consist of atom, electron, nucleon, quark and connections. Similar to that, a great program should build up with small modules which consist of smaller parts, like we can just use function of move, insert and delete to write a text editor, reusability of software modules is very important.
4. *Understanding Hierarchy*: program must have distinct gradations just like layered

cake. Application operates on the framework, framework operates on the system, and system operates on the hardware. Even inside the program, high-level can access the low-level; low-level should not know what happened to high-level. Basic feature of Event/feedback is sending no directionality notice to the high-level. As module and gradation is defined by API (application program interface), it prescribe an operation limit to everything.

5. *Great design*: the price of thinking is less than the testing, so before develop the application, we should take some time to design the application, write down what we want to achieve and a flow chart.
6. *Efficient*: program should not only operate faster, but also need save resources, it can't implicate the files, data connections and so on. We can optimize the program after test at operation level, but design need to be done before the development at high level.
7. *Clear*: clear is the basic of an excellent program, and also is the quality assurance of last 6 points.

6.1.2 Program performance test

In order to optimize the program, the first thing we need to do is program performance test; through it we can find the lack parts and suit the remedy to the case. We will use the performance testing method provided by Java language, find and fix the bug; in addition, performance must be tested with the least modification applied to the objects.

OOP (Object Oriented Programming) get through inherited abstract and modularity process to solve the problem domain. But modularity cannot solve all the problem, sometimes, the problems may occurs in multiple modules, we have to add "Log" to every function, in order to record the information of every function's entry and leaving, make the function point which can solve the problems all of the modules will result as redundancies increased, and when multiple function points appear inside one module, the code will become very hard to maintain. So we use AOP (Aspect

Oriented Programming) instead of OOP, if OOP solve the problems via vertical structure, AOP deal the problem via horizontal direction.

Also we use tendencies proxy class to achieve the several ports during the program running, every tendencies proxy class corresponding to an Invocation handles object, by means of tendencies proxy port call the invoke of invocation handler, we can get the results.

Using the thinking of AOP with help of tendencies proxy, we can get the performance test working. Performance test mainly include 5 parts:

- Computing performance: time to implement the code.
- Memory cost: memory cost when program running.
- Initialize time: time between start the program and program operate regularly.
- Scalability: how the program deal with the increased flow. We normally focus on the scaling up in order to deal with the bigger load. There are two main type of scalability, horizontal scalability transfer the application to the more powerful server, vertical scalability launch multiple services on every server.
- Perceived performance: how the users feel the speed of application.

6.1.2.1 Computing performance test

Java provide the "System.currentTimeMillis()" to let us get millisecond of current time. Instead of put this indicator inside every function, we use "java.lang.reflect.Proxy" and "java.lang.reflect.InvocationHandler" to solve the problem.

After running the program, we can get the log information from DDMS, it clearly shows the time of every function operated, we can select the most efficient function.

As we can see from the figure 30, after multiple tests of objects, it shows the differences between the thesis A version and thesis B version (without Bluetooth auto-connection function), we can see a huge improvement of operating time due to the program optimization as thesis version.

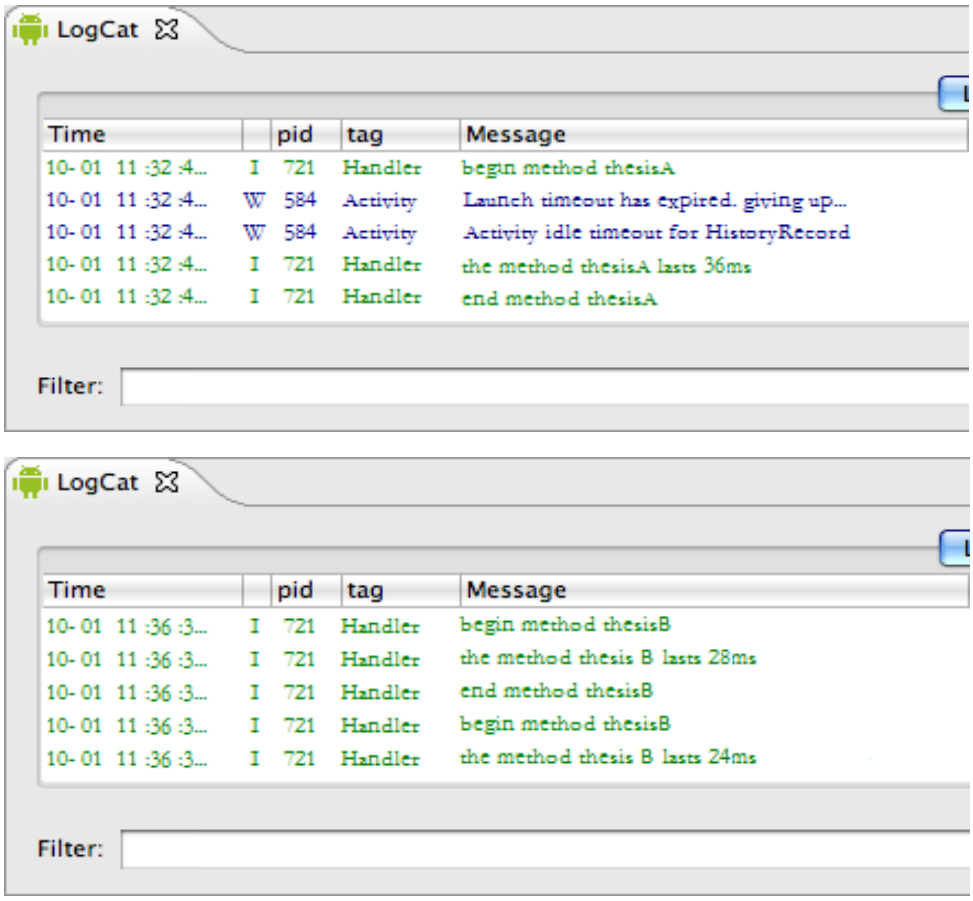

Figure 30: DDMS screen shot of computing performance test

6.1.2.2 Memory cost test

When a Java application operating, there are lots of factors can affect the memory cost, like object, thread and so on. Here we only consider the virtual machine pile of space cost, so we can use the “freeMemory()” and “totalMemory()” under Runtime class.

Same as computational performance test, we can get the log information from DDMS, it clearly shows the memory cost of each function.

As we can see from the figure, it show the differences between the thesis A version and thesis B version, as thesis B version add Auto-connection method which cause the increase of memory cost, but with the help of program optimization, memory cost still within the affordable range.

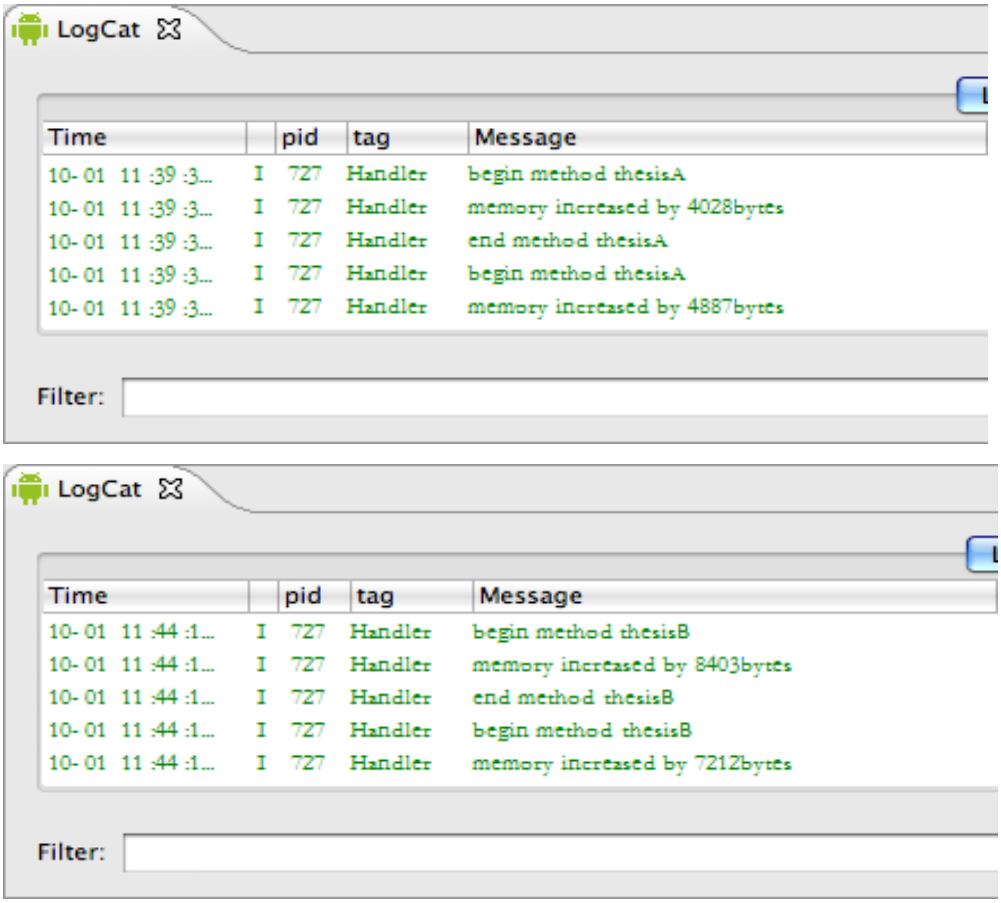

Figure 31: DDMS screen shot of memory cost test

6.2 Java language optimization

Obviously, the resources (memory, CPU, network bandwidth...) can be used by the applications is very limit, the aim of Java language optimization is making the program finish the scheduled tasks with as less resources as possible. We optimize the code via minimize the size and enhance the operating efficient of code. As inside the Java program, the problem of performance is mostly not caused by Java language, it's on account of program.

First we should develop good programming habits, like using "Java.lang.String" and "Java.util.Vector", they will significantly increase the performance of our function. The normal optimization methods been used are:

- Optimize Loop, re-organized the repeat sub-expression to enhance the operating performance of loop.

- Decrease the number of objects to enhance the performance.
- Minimize the network transfer data to reduce the waiting time.

We use the other 3 performance optimization method:

- **Use object pool pattern, increase the utilization rate of objects**.

  Java is object oriented programming language, create and release the objects cost relatively bigger resources, it's impossible to achieve the logical function without create the objects, so how to use the existing objects more efficiently is the key point to solve the problem.

  Object pool pattern create a specified number of objects due to requirement, apply the idle object when we need to create a new object, and release the object back to the pool, this way can effectively avoid the performance lost due to create and release objects.

  Analysis our requirement of application, we found that the maximum of objects we will use is 7, so we can first define an object pool as shown in figure 32:

```
Object[5]object=new Object[5];
for(int i=0; i<7: i++){
object[i] = new Object();
}
```

Figure 32: Code for object pool

In the object class we add Mark attribute "used" and "reset" to make objects reset to initial state, when the application need to create an object, just reset one idle object, when the object been released, just edit the mark for the next reset purpose.

- **Locality use basic data type replace the object to save the resources cost.**

  Although objects block details implementation, it's based on the sacrifice of storage space. Use basic data type only needs few storage spaces. Objects are greater at logical achievement, but in most situation, basic data type is more efficient, Locality use basic data type replace the object within the logical can save the resources cost.

  In our program, the typical work is during the real-time graph, as the point on the graph decrease the horizontal ordinate at a certain number, use objects are

more inefficient than the basic data type, so we replace the objects with basic data type as shown in figure 33:

```
while(1){
  for(int i=0; i<10;i++){
  if(point[1][i]==999){
  point[0][i]==time.getXposition();
  point[1][i]==ppm.getYposition();
  }else{
  point[1][j]~=10;
  }
  }
}
```

Figure 33: Code for replacing objects with basic type data

- **Use simple value computing instead of complex function computing to save the processor time.**

  Any application cannot avoid the function computing, complex and huge function computing will occupy large amount of spaces and processor time and then affect the performance. Minimize the complexity of function and decrease the numbers of call function computing will result in the improvement of computing performance.

6.3 Android program optimization

Beyond doubt, all the devices based on android platform must be an embedded device. Modern handsets are not only a mobile phone, but also a portable laptop, but, even the fastest handsets are still far behind the middle performance desktop. Even don't mention the weak battery life ability, that's why we need to consider the efficiency at any time when we program the android application.

The two fundamental principles to decide whether the system is reasonable are:

- Don't do what no need to do
- Save the memory cost as much as you can

The success of Android system is experience that applications give to the users, to optimize the application will give the user the most conveniences. Here we list some

principles we applied to optimize the application:

1. **Avoid create object as far as possible:**

   Create objects have pay the prices, one generational memory management mechanism with Thread pool allocations can make the price of create objects significantly decrease.

   If we allocate an object into a User Interface Loop, we must force to recycle memory, this will make the application appears brief pause, so what we actually did are:

   - When fetch the string from the original input data, return a sub-string from original string instead of create a copy.
   - All the string return function, whatever the situation return value is always "StringBuffer", so change the definition and implement of function, make the function direct return instead if create a temporary object.

   In a general way, we avoid create the temporary object as far as possible, the less object means the less garbage recycle, which improve the performance.

2. **Use self-function:**

   When deal with strings, we use functions like "String.indexOf()" and "String.lastIndexOf()" which comes with the object. As those function implemented via C/C++, it's 10 times faster than a Java Loop.

3. **Use static instead of virtual:**

   If the functions don't need to access the outside of object, we will make the function static. It can be called more quickly as it doesn't need a virtual function oriented sheet.

4. **Avoid use internal Get, Set function:**

   Like C++ language, we usually use "Get" function to substitute for direct access this attribute. It's good within C++, because the compiler will set the access method to be "Inline", and we can edit the access any time we want.

   But within the android programming, access to virtual method will cost a lot, so we use "Get" and "Set" function at external, direct access at internal access.

5. **Carefully use "For-each" Loop:**

"For-each" Loop are often used at "Iterable" port's inheritance collect port. In those objects, an iterator is allocated for object to access its "hasNext()" and "next()" function. We use "For-each" loop inside the array, but with the Iterable object, we ignore using "For-each" loop as there may occurs an extra object created.

6. **Avoid Enumeration:**

Enumeration is very useful, but considers the size and speed of application, it costs much more than benefits. For example:

```
public class Foo{
  public enum Shrubbery {Ground, Crawling, Hanging}
}
```

Figure 34: Code for enumeration

Code shown in figure 34 will become a 900 bytes class file just for 3 integers, so we avoid use enumeration.

7. **Avoid floating point type:**

As we all know, cross-reference integers and floats can make the application operate faster, follow this way, when we test the application, it takes seconds to draw the graph. After check with internet, unfortunately, embedded processor usually don't support the floating point process, so all the "float" and "double" are achieved via software, a basic floating point process will take milliseconds. Also the processor HTC G3 used only has multiplier. We avoid the hash table and large mathematic calculation by using only integers and many simple calculations, it's significantly improves the speed of the application.

7. Calibrations in Haze watch system

Calibration is a comparison between measurements – one of known magnitude or correctness made or set with one device and another measurement made in as similar a way as possible with a second device. The device with the known or assigned correctness is called the standard. The second device is the unit under test, test instrument, or any of several other names for the device being calibrated [*14. Wikipedia, Calibration*].

The calibration of gas sensors is the basis of whole project, and gets a great challenge for the long term outcome of sensor board. The carried out experiment which last year done of calibration is comparing the measurements taken by both wireless sensor board and the Gas Alert Micro 5 Commercial meter. After record numbers of data sets, study and estimate the output equations, the experiment is using vehicle exhaust as the gas generator, so the concentration of data cannot be controlled as well as the recorded readings. The equation and method is just briefness and rough experimental results, lack of accuracy and system integrity will cause incorrect readings from the device and then affect the stability of whole system.

A successful calibration has to be able to "hold a calibration" through its calibration interval. In other words, the design has to be capable of measurements that are "within engineering tolerance" when used within the stated environmental conditions over some reasonable period of time. Having a design with these characteristics increases the likelihood of the actual measuring instruments performing as expected [*14. Wikipedia, Calibration*].

7.1 Method 1: Experimentally doing calibration using pure gas cylinders.

This method was suggested by Dr. Martin Bucknall from Bio-analytical center [*15*]. The overall idea of calibration is shown in figure 35:

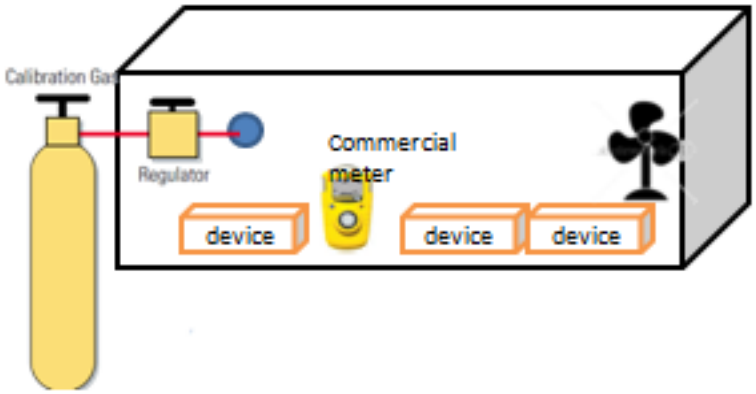

Figure 35: Structure of Calibration idea [*16*]

As see from the graph, the gas container which is a tank made by plastic material; the plastic lid is sealed by o-ring on top of tank and can be screw up and down, the lid will tightly on to insure there is no air leakage of the tank; a septum is drilled on the middle of tank's side which will connect the gas cylinder with regular on it; a commentator free fan will be attached insider of tank to mix up the injected gas; a commercial meter and 3~6 wireless sensor board are putting insider the tank one near another for the purpose of taking readings.

The general idea is using Gas Alert Micro 5 Commercial meter as a standard reading when we slowly inject certain amount of gas into the tank, after the gas finish mixes with air insider the tank, we compare the readings between commercial meter and the data we get from the matlab (code refer to Appendix A) which we already assign the port within the computer as shown in figure. All the desired features of Gas tank are being built at school workshop; the next step would be purchase gas cylinders with mixture of the requisite gas.

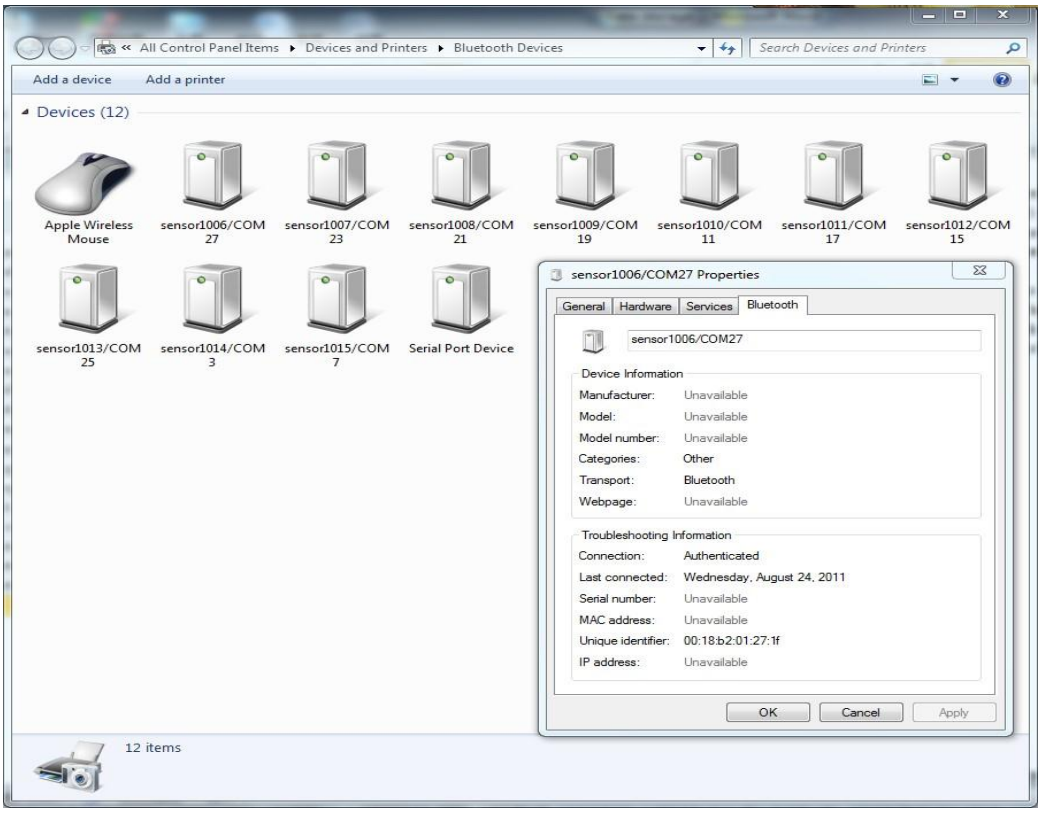

Figure 36: Computer assigned ports with sensor devices

Successful calibration has to be consistent, systematic and traceable. For area air quality and safety gas monitors, the requirements need to be simple, repeatable, and economical.

According to WHO air quality and health specification and the capability of sensor specifications we obtained the guideline values of each gas:

Gas name	WHO Guideline values (ug/m^3)	PPM	SENSOR CAPABILITY Range
Ozone (O_3)	100ug/m^3 (8 hrs mean)	0.05ppm	0.01 -1PPM
Carbon monoxide (CO)	100mg/m^3 (15min) 35mg/m^3 (1 hr) 10mg/m^3 (8 hrs) 7mg/m^3 (24 hrs)	87.33ppm 30.568 ppm 8.7336 ppm 6.1135 ppm	1 -1000PPM
Nitrogen dioxide (NO_2)	40 ug/m^3(annual mean) 200 ug/m^3 (1hr mean)	0.02128ppm 0.10638ppm	0.05 TO 5 PPM

Table 1:WHO Guidelines of pollutant levels [*16*]

From table 1, we will observe that all the metal oxide sensors we use have the measurable capability within range of WHO guidelines. Based on table 1 and under the calculation help of Dr. Martin we decide the measurement ranges and purchase list would be:

- ✧ CO: 0 to 200 ppm pure gas, increments of 10ppm. If we inject 2.8ML into the tank, that will be 200 ppm.
- ✧ NO2: 0 to 1 ppm, increments of 0.05ppm. we should get 10000PPM diluted in air or nitrogen. If we inject 1.4ML into the tank, that will be 0.5 ppm.
- ✧ O3: 0 to 5 ppm, increments of 0.05ppm, we should get 5000 ppm diluted in air or nitrogen. If we inject 1.4ML of this in to the tank, that will be 0.5 ppm.

All injection of sensible volumes is using the syringes as suggested by Dr. Martin to ensure accuracy.

On the other hand, we also asked E2V company which manufactured the sensors, they suggested us using the absolute measurement for processing the sensor signal, which is during the calibration, we need calibrate two points usually it is zero and the point of our standard gas. For example, to calibrate the CO, firstly we put the sensor in the ambient air to record the output as the zero and then to give like 50ppm or 100ppm CO through the sensor and then to record the output. With these two points, we get the characterization line of the sensor, and then we can get the concentration from the sensor output at different concentration. As sensor may drift, the intervals between calibrations can be different from sensor to sensor. Generally, the manufacturer of the sensor will recommend a time interval between calibrations. However, our sensor is not good at the repeatability and reliability it is possible to observe how well the sensor is adapting to its new environment for only few weeks before next calibration.

Also as suggested by Dr. Martin, we contact BOC and Liquid which are two companies can purchase the gas bottles. Then we came across difficulties and concerns about not only the budget are lower than the gas bottles but also we don’t have the commercial qualification and we are not expertise to do the calibration. As the calibration need to be certificated, we postpone this method and try to find if there exists any company have the qualification and willing to help us with calibration and

ensure its reliability [*16*].

7.2 Method 2: E2V Gas Sensor Evaluation Kits

From the response of the companies that manufacture and calibrated sensors, we find out that the company will also create products that can calibrate and adjust the sensors just like the design of our sensor board but without the wireless function. Companies in Sydney which can help us all have a concern, since they are not the manufacture company and we get our own board, they can't guarantee the calibration procedure will be precisely correct. Luckily E2V have the evaluation kits particularly for those metal oxide sensors we used.

Figure 37: E2V metal oxide semiconductor gas sensor evaluation kits

Figure 37 shows the components contains in the evaluation kits, we can easily get the high performance data output via USB connect to our computer by using the kits, the software can log and control the performance of different sensors to be assessed.

In the meantime, during the field test of sensor board, we also found some discrepancy and weakness along the sensor board design.

- ADC (analogue to digital converter) output gets error digital value compare to the input voltage.
- Data format sometimes messes up, like header and footer mix together.
- Sensor voltage output does not match the output pin of microcontroller. The

situation now is all the readings we been received have shift one position, like shown below:

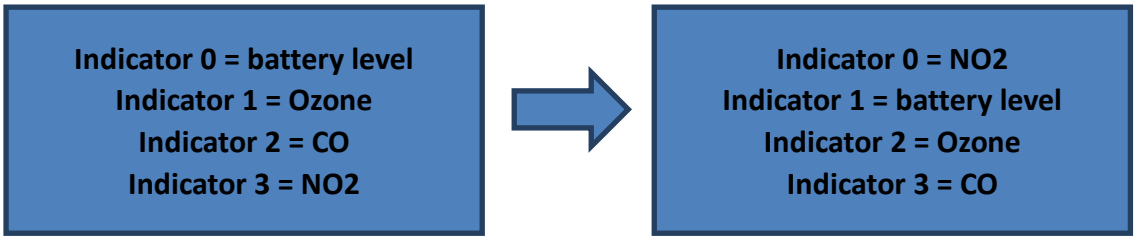

Figure 38: Microcontroller output shift problem

Those weakness and discrepancy makes calibration work done by ourselves even harder. So use the kits seems more reliable and would be a perfect choice to have this kit around the project if it can solves the weakness of current designed board and also able to help with the calibration.

8. Future Work

8.1 Wireless sensor board and Calibration

As mentioned in chapter 7.2, wireless sensor board which designed by last year thesis people have few problems after our field test, troubleshooting should be taken to solve the problem of ADC converting error and transmitted data format error. The procedure for troubleshooting test could start with ignore the PCB design, just connect microcontroller with power and Bluetooth or Oscilloscope, then use power generator input into ADC and varies the voltage level, get digital output either from Bluetooth transmission shown at computer, or read the pulse from oscilloscope, compare the analog input and digital out, see if the problems within the microcontroller or the other parts of circuit.

Also consider of calibration issue, we recommend purchasing the evaluation kits from E2V company, and the idea of using this evaluation kit for the future work would be:

- The evaluation kit has a JTAG header which allows advanced users to upload their own software to the microcontroller (MSP430F2616) and make full use of the available interfaces. Once make sure the evaluation kit could help and easy understandable, microcontroller should carried out to our message protocol format.
- As now the evaluation kit use USB connected to computer for the output, we can attach Bluetooth module into it or design a board based on the principle of evaluation kit, in order to communicate with the Smartphone, but either way, we should fully understand the structure and properties of evaluation kit first. [*16*]

8.2 Android Application

Although the application is working now, it still has discrepancies need to improve:

- Stability of application: There are lots factors can affect the stability of our application, like data message format error, Location server lost or temporary

shutdown in purpose of maintain, It needs more field test and debug to improve the stability, and as there must still have unknown factors may cause the application crash, it need a function to record the operation statement and error, for the purpose of future development.

- Privacy concerns: While collecting pollutant data, the location of user which periodically uploaded to the server is just like a real-time track, those can be used in illegal way, and also that information is users' privacy. By the consideration of prevent people's privacy rights, the location information should be encrypted before upload to the server, and only server can decrypt the information and authenticated people can access to the server. It's a huge work to be done and could be a great improvement in the area of network security.

9. Conclusion

This report has briefly introduced the “Haze Watch” project and shown how it works to improve pollution monitoring. As the combination of our sensor units, server database and user application, It is believed that, this system can help the public and society to build a new type of pollutant report system which will improve our health and environment.

The most focusing aspect for us is on android interface design and implementation within mobile sensor unit. The interface results we provide to the currently system fulfilling the aim of using android system as a collector element. After optimization and with the help of modes operation and automation method, application significantly improve the power consumption and performance. More modification and creation are expected in the future.

10. Reference sheet

[1]Australian Government, department of the environment and heritage, air quality fact sheet. Published 2005[Online, accessed 18th May 2011]

http://www.environment.gov.au/atmosphere/airquality/publications/standards.html

[2]Oracle Education, The Environment a global challenge.[online, accessed18th May 2011]

http://library.thinkquest.org/26026/Environmental_Problems/air_pollution.html

[3]State Government of New South Wales. (2010, Apr.) Environment, Climate Change and Water.[Online, accessed 18th May 2011].

http://www.environment.nsw.gov.au/air/

[4] What is Android| Android developers.[Online, accessed 2011].

http://developer.android.com/guide/basics/what-is-android.html

[5]Application fundamentals|Android developers.[Online, accessed 2011].

http://developer.android.com/guide/topics/fundamentals.html

[6]Fengsheng Yang, Android Unleashed Android application development secrets, China Machine Press [Book, Oct 2010]

[7]J Burke et al, Participatory Sensing, ACM, [Paper, 2006]

[8]N. Youdale, Haze Watch: Database Server and Mobile Applications for Measuring and EvaluatingAir Pollution Exposure, [University of New South Wales Undergraduate Thesis, 2010]

[9]GPS-Baidu encyclopedia.[Online, accessed 2011].

http://baike.baidu.com/view/628443.htm

[10]Chris Veness, Calculate distance, bearing, etc. between 2 Latitude/Longitude points, [Online, 2002~2007]

http://www.yourhomenow.com/house/haversine.html

[11]Wikipedia, SQLite,[Online]

http://en.wikipedia.org/wiki/SQLite

[12]SQLite, Distinctive Features Of SQLite, [Online]

http://www.sqlite.org/different.html

[13] Jani Jarvinen, Javier Desalas, Jimmy Lamance. Assisted GPS: A

Low-Infrastructure Approach. [Online, March 1, 2002].

"http://www.gpsworld.com/gps/assisted-gps-a-low-infrastructure-approach-734".

[14]Wikipedia, Calibration, [Online]

http://en.wikipedia.org/wiki/Calibration

[15]Dr. Martin Bucknall, Bio-analytical center, [Online]

http://www.bmsf.unsw.edu.au/AboutUs/ContactPages/Bucknall/ProfileMB.htm

[16]Kunxuan Bi, Android interface for pollution monitoring system, [Paper, 2011]

11. Appendix A

```
function []=calibration(com_number,filename)
%
sensor = serial(com_number);
set ( sensor, 'BaudRate', 9600);
set (sensor, 'Terminator', 45);
fID=fopen(filename,'w');
fopen(sensor);
    Sensor0 = 0;
    Sensor1 = 0;
    Sensor2 = 0;
    Sensor3 = 0;
    Plotsamples = 100;
    x1 = zeros(1,Plotsamples);
    x2 = zeros(1,Plotsamples);
    x3 = zeros(1,Plotsamples);
    i = 1;
    countt = 0;

while(countt~=10)

  count = 1;
  voltage1total = 0;
  voltage2total = 0;
  voltage3total = 0;
  samples = 1;
  while (count <= samples)

 buffer = fread(sensor,1);
 while( buffer == 255 || buffer == 238)
    buffer = fread(sensor,1);
 end
 if (buffer == 00) % Sensor 1
    Sensor0 = fread(sensor,1);
    buffer = fread(sensor,1);
 end
 if (buffer == 01) % Sensor 1
    Sensor1 = fread(sensor,1);
    buffer = fread(sensor,1);
 end
  if (buffer == 02) % Sensor2
    Sensor2 = fread(sensor,1);
```

```
    buffer = fread(sensor,1);
  end
  if (buffer == 03) % Sensor 3
    Sensor3 = fread(sensor,1);
    buffer = fread(sensor,1);
  end
    voltage0 = Sensor0/255 * 3.3;
    voltage1 = Sensor1/255 * 3.3;
    voltage2 = Sensor2/255 * 3.3;
    voltage3 = Sensor3/255 * 3.3;
      fprintf('here\n');

    fprintf(fID,'%f   %f   %f\n',voltage1,voltage2,voltage3);


    voltage1total = voltage1total + voltage1;
    voltage2total = voltage2total + voltage2;
    voltage3total = voltage3total + voltage3;
    count = count + 1;
  end


    voltage1 = voltage1total/samples;
    voltage2 = voltage2total/samples;
    voltage3 = voltage3total/samples;

    resistance1 = (10000*5)/(5-voltage1);
    resistance2 = (10000*5)/(5-voltage2);
    resistance3 = (10000*5)/(5-voltage3);

    %pollution1 = 0;
    %pollution2 = 0.0405*exp(5.2621*voltage2);
    %pollution3 = 0.0405*exp(5.2621*voltage3);

    pollution1 = voltage1;
    pollution2 = voltage2;
    pollution3 = voltage3;

      count = 0;
      countt=countt+1;
end

fclose(sensor);
end
```