Application Layer for Vehicle Based Road/Environment Condition Warning System using Vehicular Ad hoc Networks (VANETs)

by

MUHAMMAD HAFIZ BIN MUHAMMAD NASIR

DISSERTATION

Submitted to the Electrical & Electronics Engineering Programme
In Partial Fullfillment of the Requirements
For the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

©Copyright 2011
by
Muhammad Hafiz Bin Muhammad Nasir, 2011

CERTIFICATION OF APPROVAL

Application Layer for Vehicle Based Road/Environment Condition Warning System using Vehicular Ad hoc Networks (VANETs)

By

MUHAMMAD HAFIZ BIN MUHAMMAD NASIR

A project dissertation submitted to
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

Assoc. Prof. Dr Mohamad Naufal bin Mohamad Saad
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
SEPTEMBER 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

Muhammad Hafiz bin Muhammad Nasir

ABSTRACT

The increasing number of accident occurring in Malaysia is an important issue that needed to be taken into account. Lack of focus of the driver while driving vehicle contribute a lot in road accidents especially during rain, snow and fog environment. There is a need to develop a safety application for the drivers to prevent them from accidents in different scenarios such as fog, bad weather condition and road condition ahead. This project will focus on developing an application that able to alert the driver of any abnormal road condition and spread the warning message around it. It can be realized by implementing a new type of networks called as Vehicle Ad hoc Networks (VANETs). This network dynamically form ad hoc network between vehicles in the same time while travelling. This application use VANETs to spread the warning message to other vehicles. By the end of this project, the application will be able to work with sensors such as weather sensor, temperature sensor and road slippery sensor in obtaining the crucial data, process the data and broadcast the warning messages to other vehicle alert about the abnormal condition ahead.

ACKNOWLEDGEMENTS

I would like to thanks to my supervisors, Assoc. Prof. Dr Mohamad Naufal bin Mohamad Saad for giving me the opportunity to do this final year project and always understanding despite all the challenges faces throughout the timeline.

My utmost appreciation also goes to Prof. Anis Laoutti from TELECOM SudParis, Mr Sajjad Akbar and Miss Saira Gilani from M. A. Jinnah University who are guiding me in the field of VANETs. The information that you gave me is really helpful for me in understanding the project.

Finally, I would like to express my gratitude to my family, friends and everyone who has contributed directly and indirectly; to be supportive towards me throughout the project period.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Most of the accidents occurring contributed by the lack of focus of the driver while during vehicle especially during rain, snow and fog environment. There is a need to develop a safety application for the drivers to prevent them from accidents in different scenarios such as fog, bad weather condition and road condition ahead

This chapter will give the overview about our project. Starts with project background, problem statement, objective and end with scope of study.

1.1 Project Background

The road becoming busier by day as the number of vehicles is increasing. As we refer to *Figure 1*, the number of registered vehicle in Malaysia keeps increasing every years and as for 2010, it was increased by 13% compare to previous year and it reached 605,156 [1]. Thus, it will make the road more congested since the new road is not following the increasing rate of the vehicle and vehicle will be more prone for accident.

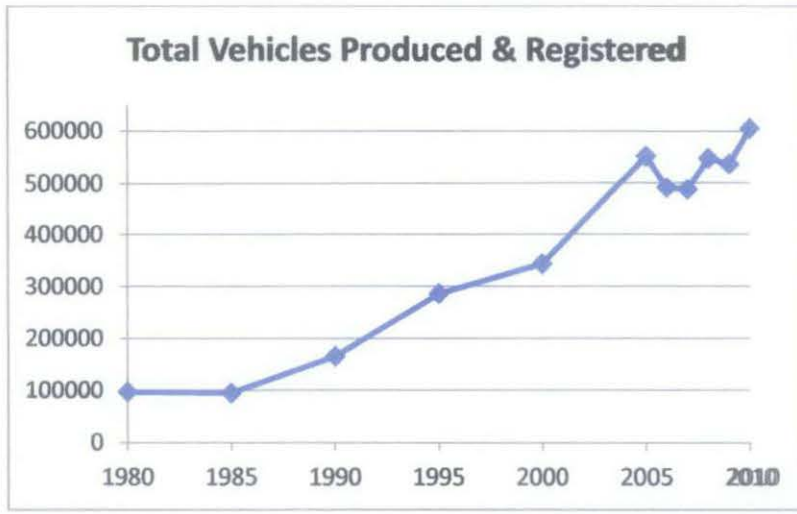

Figure 1: Total Number of Vehicles Produced & Registered [1]

There are several safety applications implemented in the vehicles that will assist the driver to avoid the accident. Those features implemented in the new vehicles nowadays as part of value-added safety feature. Most of safety features such as Automatic Brake System or Adaptive Light Control only involved within a vehicle [2]. If there is a mechanism that can tell the other drivers about safety alert whenever the vehicle detect it, it definitely could avoid some accident. The cooperation communication between vehicles to exchange safety information to each other can be realized. There is an emerging type of networks where communication occurs between two or more vehicles and it called as Vehicle Ad hoc Networks (VANETs). These networks form an ad hoc network while the vehicles travel to provide the opportunity to communicate between vehicles.

The promising idea of VANETs brings a new range of applications that include the active safety applications have potential influencing the safety of vehicle and its occupants.

There is a need to develop a safety application for the drivers to prevent them from accidents in different scenarios such as fog, bad weather condition and road condition. This project will focus on the design safety application that will make possible to notify drivers about the abnormal road condition. This application will also monitor the driving environment. Once the data about the abnormality of the road/environment has been collected, this application will notify the driver about how hazardous the situation is and spread this information to other vehicles, in order to take precaution of avoiding accidents.

Stated below is the timeline for Application Development of this research:

- First Year Domain
    - Application Scenarios
    - Sensors Selection
    - Map Overlay Design
    - Application Graphical User Interface (GUI) Development
    - Integration of Maps and Overlays with Application
- Second Year Domain
    - Improvement of GUI Development
    - Improvement in integration of Maps and Overlays with Application

- Integration Application with other communication protocol layers such as Dedicated Short Range Communication (DSRC)
- Real Time Lab Testing of Application
- Real Time Test bed Evaluation of Application

For this final year project, the focus will be more on the First Year Domain of the research project. The basic function will be defined during this year. Different kind of scenario will be taken into consideration when planning this project. The plan will include on what sensors are needed, what kind of map are needed, develop the user interface and add some extra features such as weather forecast, wind speed and temperature that will assist the driver on his driving experience. By the end of the First Year Domain, the application will able to view all the information needed on the GUI.

1.2 Problem Statement

Figure 2: Road accident situation

About more than 300,000 traffic accidents occur annually in Malaysia. In year 2010 alone, these accidents accounted for RM 9 billion in damaged property, 391,751 non-fatal injuries and 22,260 with injuries. In addition, the average death caused by road accident is 17 people each day [3]. Based on the World Health Organization (WHO), nearly 1.3 million people die as result of a road traffic collision and more than 3000 death each day [4].

The ratio of road accidents can be reduced significantly by using proper traffic management applications. This traffic management solution will make driver use the road much safer despite of the poor latency of human when reacting to emergency situation. There is a need to develop a safety application for the drivers to prevent them from accidents in different scenarios such as fog, bad weather condition and road condition ahead. The examples of abnormal condition are shown in Figure 3, Figure 4 and Figure 5. All those conditions contributed a lot in road accident around the world when the driver fail alert about the abnormal condition ahead.

Figure 3: Raining on the Road

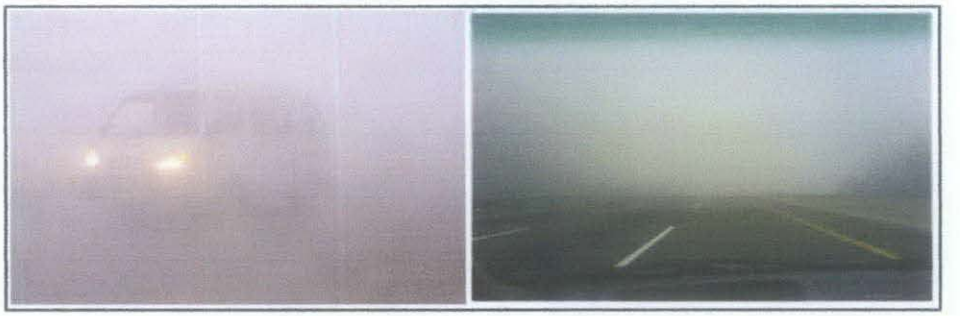

Figure 4: Fog on the road

Figure 5: Snow on the road

In order to develop a safety application for vehicle, the application need to be able to know the current location of the vehicle and able to sense the abnormal condition around it. The combinations of GPS, fog and road condition sensors are needed in order to interface it and spread the warning message to other vehicles through VANETs with the application. The integration of the sensors with the application helps the system to be interactive.

1.3 Objectives of the Project

The main objective of this project is to develop a safety application for vehicles which able to:

- Getting data by sensors (position abnormal condition and weather parameters)
- Spread data to other vehicles
- Received data and display it to the drivers

1.4 Scope of Study

For this project, the application will be developed for desktop version. The application will be interfaced with Global Positioning System (GPS) receiver, useful data from the internet and a network. The GPS receiver will be used to receive useful information such as the current coordinate, velocity of vehicle, satellite time and number of satellite connected. The information for weather forecast, temperature, wind speed and map of current position will be fetched from the internet. Then, when the application detects the abnormal condition, it will broadcast the warning message. That message should have type of warning condition, broadcast time and the coordinate of the sender and send it to the other vehicles.

To complete the project, the studies need to be done in how to interface the GPS data and the weather information from the internet into our application. After that, the network interface needs to be implemented into the application to make it able to send and to receive broadcast messages from other application within the network.

1.5 Relevancy of the project

The implementation of VANETs in broadcasting warning messages to others gives a promising future in providing more safety caution on the road. This project will contribute in giving early warning to the drivers about abnormal weather and road condition to make them more cautious driving on the road.

1.6 Feasibility of the project

This project needs a lot of testing when integrating all the safety features in it. This application need to be stable enough to fetching and sending all the data needed. If the procedure of this project followed closely, the objective can be achieved and complete on time.

CHAPTER 2

LITERATURE REVIEW

2.1 Intelligent Transportation System (ITS)

Intelligent Transportation System is a system that able to sense the driving condition. It also able to provide the information or vehicle control that can enhance the driving operation. The system will operate and make the decision for us such as route choice what can be supported by using the on-board navigation system [5].

2.2 Vehicle Ad hoc Networks (VANETs)

One part of the ITS is VANETs. The infrastructure provided in order to enhance the people safety and comfort. Vehicles equipped wireless communication device will form network between them [6]. The Figure 6 shows the idea on how the communication occurs between vehicles. It shows that the possible connections for VANETs are Vehicle-to-Roadside (V2R) and Vehicle-to-Vehicle (V2V).

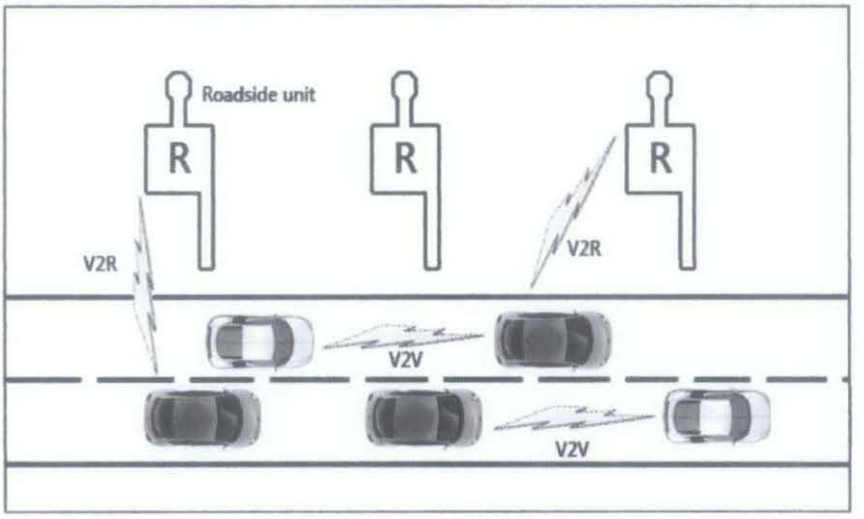

Figure 6: Vehicular Network view [6]

The existing wireless network architecture still could not efficiently support such demands (bandwidth, delay, security and reliability). Therefore, it becomes a major challenge to support and enable diverse applications and services. There are several current and future applications proposed for VANETs [7]:

- **Electronic Toll Collections (ETCs)**
  Each vehicle pays the toll electronically when it passes through a Toll Collection Point without stopping. It will scan Electrical License Plate at the

On Board Unit (OBU) of the vehicle and issues a receipt message to the vehicle including the amount of the toll, the time and the location of the Toll Collection Point.

- **Safety Warning Applications**
  Deliver the warning to nearby nodes in a very short time. Such application need to be delivered in a very short time with high priority. It will use Dedicated Short Range Communication (DSRC) in broadcasting the messages.
- **Internet Access**
  Future vehicles will be equipped with the capability so that the passengers in the vehicles can connect to the Internet.
- **Group Communications**
  Drivers may share some common interest when they are on the same road going the same direction.
- **Roadside Services Finder**
  Find restaurants, gas stations etc. in the nearby area along the road.

2.3 Safety Application for Vehicle

The application layer provides communication services to the actual applications that are running on the node. These applications are expected to be produced by variety of manufacturers when the VANETs has widely implemented. The applications will monitor the sensors and spread the warning messages when any abnormality is detected. This safety application will be also able to notify the driver on his own location, about his neighbors' locations, weather condition ahead and about the road condition ahead [8]. Thus, the driver can adjust his vehicle to adequate situation such as speed of the vehicle or turn on the fog light to avoid any accident that might occur.

2.4 Global Positioning System (GPS)

2.4.1 GPS Receiver

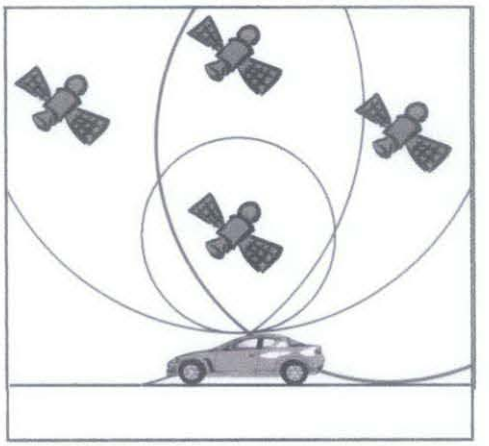

Figure 7: How GPS determine its position [9]

The Global Positioning System (GPS) is a satellite based navigation system that enables a user with a GPS receiver to pin-point his location to an accuracy of a few meters anywhere on earth's surface that has a visibility of at least three GPS satellites [9]. GPS receivers take the information from the GPS satellites and use triangulation to calculate the current exact location. Figure 7 shows how GPS receiver gets the information from different satellites. Each satellite will give the distance between the receiver and the satellites. To obtain 2D position (latitude and longitude) and track movement, a GPS receiver must be locked at least three satellites. When it receives four or more satellites in view, the receiver will be able to determine the current 3D position (latitude, longitude and altitude). With all those information, the receiver will be able to calculate other information such as speed, bearing, track, trip distance, distance to destination and much more [10].

Most of GPS receivers' output data are in National Marine Electronics Association (NMEA) format. The data send out by the GPS sent via RS-232 port. Table 1 shows the useful GPS NMEA messages that used for navigation applications.

Table 1: Information provided by the GPS receiver [11]

$GPRMC	Contains recommended minimum specific GPS/transit data
$GPALM	Provides GPS almanac information
$GPGLL	Provides latitude, longitude, and UTC (Universal Time Coodinated) data
$GPZDA	Contains UTC along with day, month, year, and local time
$GPGGA	Contains UTC, fix, and position data
$GPGSA	Provides GPS DOP and active satellite information
$GPVTG	Provides "track made good" and ground speed
$GPZDA	Provides the current time and data

All NMEA data is emitted as ASCII data. With all those useful information from **Error! Reference source not found.**, a lot of features can be implemented in this project.

2.4.2 Determining GPS Precision

The precision of GPS receiver can be obtained from $GPGSA sentence. That sentence gives some useful accuracy information of the GPS receiver. The information are Horizontal Dilution of Precision (HDOP), Vertical Dilution of Precision (VDOP) and Position Dilution of Precision (PDOP). In Figure 8, the gray area is the possible point for the GPS position. The GPS tells us that the point could be anywhere in the grey area. HDOP is the mean DOP of latitude and longitude. VDOP is the DOP for altitude while PDOP is the mean DOP of latitude, longitude and altitude. The DOP value ranged from one till fifty. One represents ideal accuracy while fifty represent very poor accuracy [12].

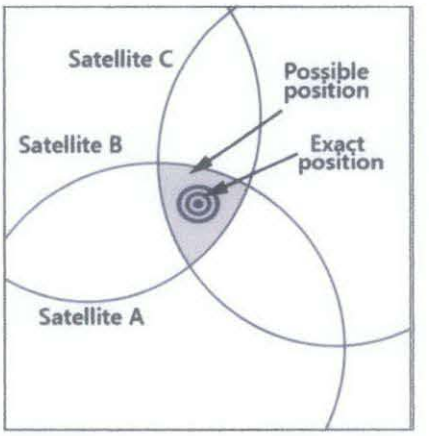

Figure 8: The cause of error in accuracy [12]

Typical GPS receiver is capable to have 5-7 meters accuracy or average 6 meters without any enhancement. For the car navigation, HDOP is commonly used to determine the precision of vehicle position. The formula that helps to determine the precision of GPS receiver shows in (1).

$$(Accuracy\ of\ GPS) * (DOP) = (Max\ allowable\ error) \quad (1)$$

By the combination between the average GPS receiver accuracy and the VDOP that recommended for vehicle, the accuracy of the GPS coordinate from the GPS receiver can be determined.

CHAPTER 3

METHODOLOGY

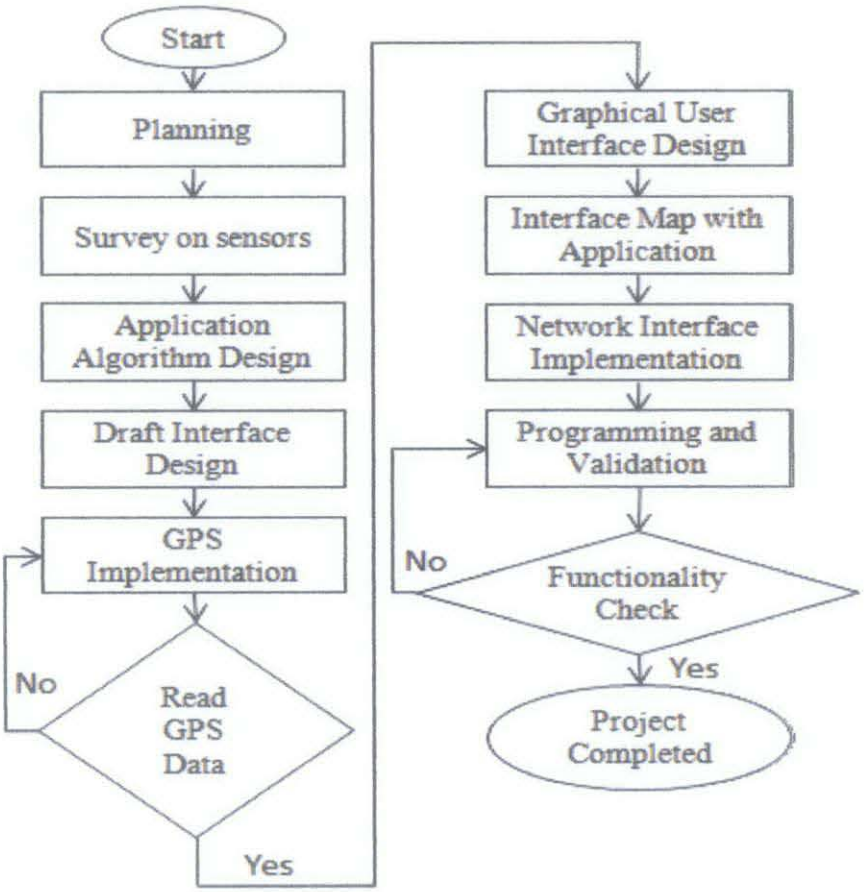

Figure 9: Flowchart of the project

3.1 Planning

The first step of the project is planning. Some studies need to be done especially in the area of VANETs. It involves a lot of background studies about the problem. Background studies help to understand about our angle to tackle the problem. Then, further studies on literature review helps to gain useful information that will assist in the project execution especially in decision making on what features need to be included in the project. Table *2* and Table *3* show the Gantt chart for this project.

Table 2: Final Year Project 1 Gantt Chart

	Final Year Project 1 Week Number													
Activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Title selection / proposal														
Find resources														
Study about research														
Proposal defense														
Find suitable sensors														
Familiarize with Visual Basic														
Familiarize with Google Map API														
Develop Application Interface														

Table 3: Final Year Project 2 Gantt Chart

	Final Year Project 2 Week Number													
Activities	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Interface GPS with the application														
Test the obtained GPS data														
Develop User Interface														
Interface Map with the application														
Interface with Network														
Minor Improvement of the application														
Programming Validation														

3.2 Sensors – Byonics GPS2

The sensor that will be implemented in this application is the GPS receiver. This GPS receiver will be the main part of the application. It provides vital information for the vehicle such as coordinate, velocity and satellite time. The GPS receiver that will be used is Byonics GPS2. Sensors to detect the abnormal condition will not be implemented in this project and will be replaced with a selection button that will simulate the detection on the application.

Byonics GPS receiver is a 5-volt GPS receiver with output 4800 baud NMEA 0183 serial. It is using SiRFStartIII chipset that able to track up to 20 satellites. The position acquisition times for this GPS receiver are 42 sec for Cold Start, 38 sec for Warm Start and 1 sec for Hot Start. It cased with weather resistant and contains a magnet that helps to be easily mounted outside the vehicle. The NMEA sentences sent out each second are $GPGSA, $GPRMC, $GPGGA and $GPGSV [13].

Figure 10: Byonics GPS receiver [13]

3.3 Application Algorithm Design

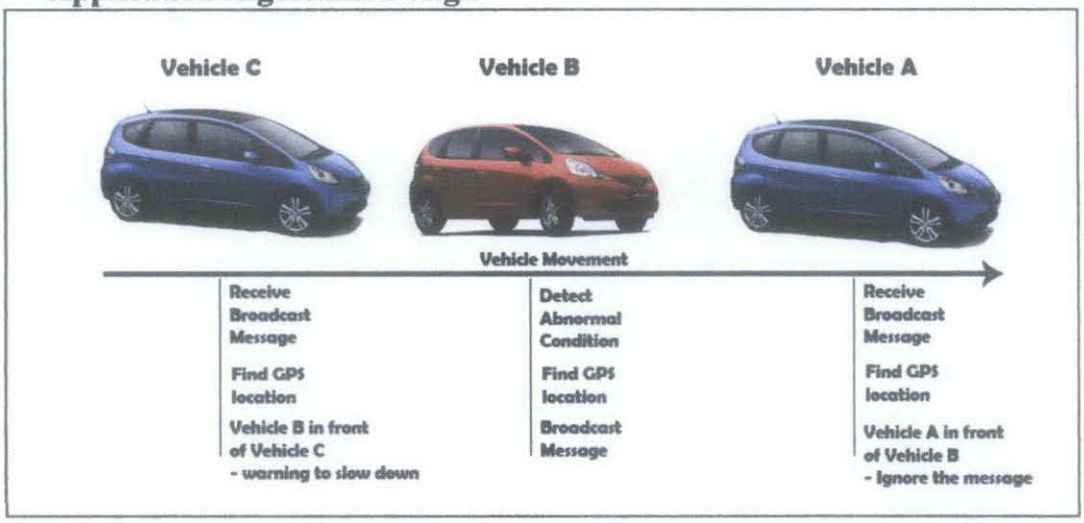

Figure 11: Example of situation when vehicle B detect abnormality on the road

Figure 11 shows the example when a vehicle detecting abnormal condition on the road and will try to transmit the warning message. That message will be received by vehicles around it. Below is the algorithm that will be using when Vehicle A and C receive the warning message:

1. When Vehicle B detects abnormal condition on the road, it will capture the current GPS coordinate and broadcast the message.
2. Vehicle A and Vehicle C will receive the message from Vehicle B.
3. Vehicle A and Vehicle C will compare its own current GPS coordinate with Vehicle B GPS coordinate.
4. Vehicle A and C will determine the movement path for its own vehicle. It can be done by comparing the previous coordinate with the current coordinate.
5. After movement path determined, Vehicle A and Vehicle C will determine either the vehicle is in front or at the back of Vehicle B.
6. If vehicle is in front of Vehicle B, vehicle will ignore the message.
7. If vehicle is at the back of Vehicle C, vehicle will process the message and show warning to slow down.

3.4 Draft Interface Design

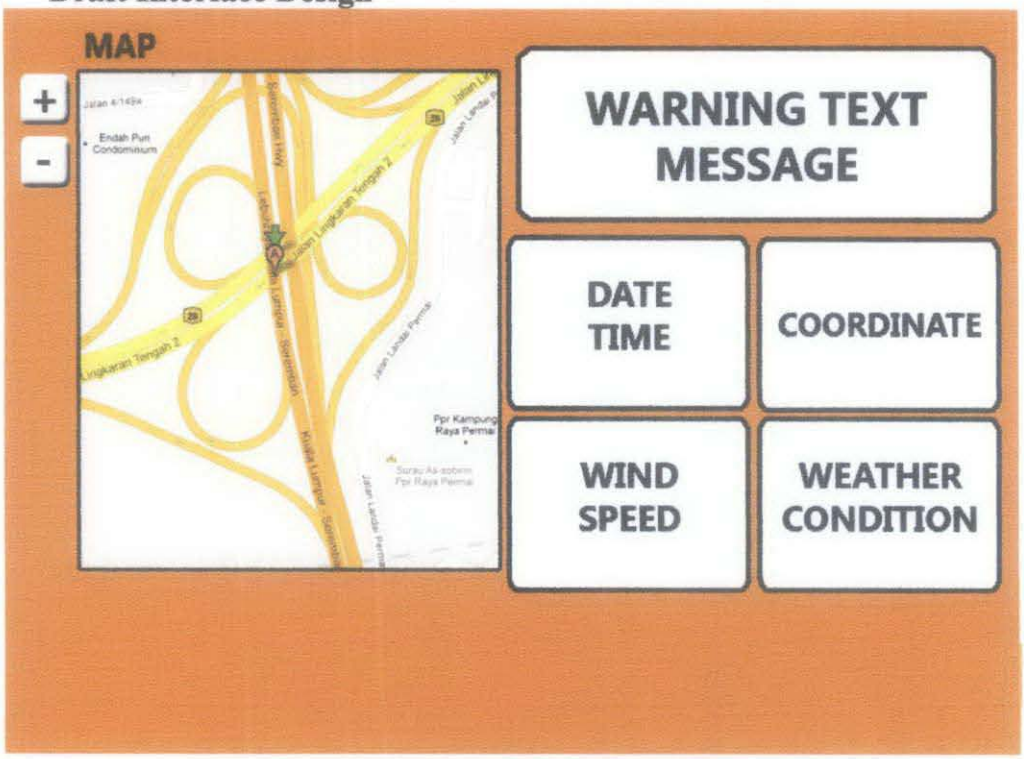

Figure 12: Graphical User Interface example

The planning decision on what to include in the application decided in this phase. Figure 12 is the draft design on how Graphical User Interface (GUI) will be. The features that will be include in the interface need to be able to assist the driver to have better driving environment.

3.4.1 Warning message display

Whenever sensors detect some abnormality on the road, it will notify the driver and view the warning message on the *display*. *Figure* 13 shows the example of the warning messages.

Figure 13: Example of warning message

3.4.2 Map shows the current situation when abnormal situation detected

Whenever a vehicle receives warning information from other vehicle about the abnormal condition, it will alert the driver. The map is shown as in Figure 12. The message that received from other vehicle will consist on what type of warning and what is the coordinate of the origin sender.

3.4.3 Zoom in and zoom out map

The zoom in and out feature is needed to make the driver easily see the map much clearer by his own preference.

3.4.4 Weather report on that location

Weather report for current vehicle location will help the driver to know what will and happening to the weather during driving. It helps driver to prepare any bad weather situation and will make more alert during driving.

3.4.5 Wind speed

Driver need to know the wind speed for his current location. It helps alert the driver to drive much more careful because the strong wind may result in the loss of control of vehicle and have potential for the vehicle to completely blow over.

3.4.6 Give a warning sound when warning message arrived

Warning sound is important to make the driver alert whenever vehicle receives warning messages from other vehicles or detecting any abnormality on the road.

3.4.7 Show the distance between vehicle and warning location

The distance between two point need helps the driver to estimate the distance of the warning location. This helps driver to prepare in facing the abnormal condition ahead. The formula to calculate the distance between two coordinates is:

$$dlon = lon2 - lon1$$

$$dlat = lat2 - lat1$$

$$a = sin\text{^}2(dlat/2) + cos(lat1) * cos(lat2) * sin\text{^}2(dlon/2)$$

$$c = 2 * arcsin(min(1, sqrt(a)))$$

$$d = R * c \quad (2)$$

3.5 GPS Implementation

GPS receiver implemented during this phase. It is connected to RS232 port. For computer that does not have RS232 port, the USB-Serial converter can be used.

Figure 14: USB-Serial converter

To connect Byonics GPS2 with the RS232 port, some modification needs to be done since RS232 does not supply 5-volt to the GPS receiver. The Byonics GPS2 designed to be powered from a 5volt source with + on PIN 4 and Ground on PIN 5 [14]. This modification will make this GPS receiver to use 2 USB port. One port for USB-Serial converter to receive data from Byonics GPS2 and another port is to power up the GPS receiver. Figure 15 shows how connect Byonics GPS2 with the USB-Serial converter. The 5-volt power supply received from the modified USB wire shows in Figure 16.

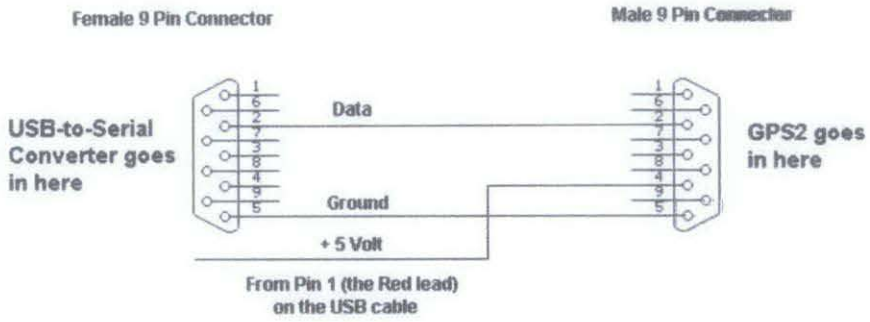

Figure 15: How to modify Byonics GPS2

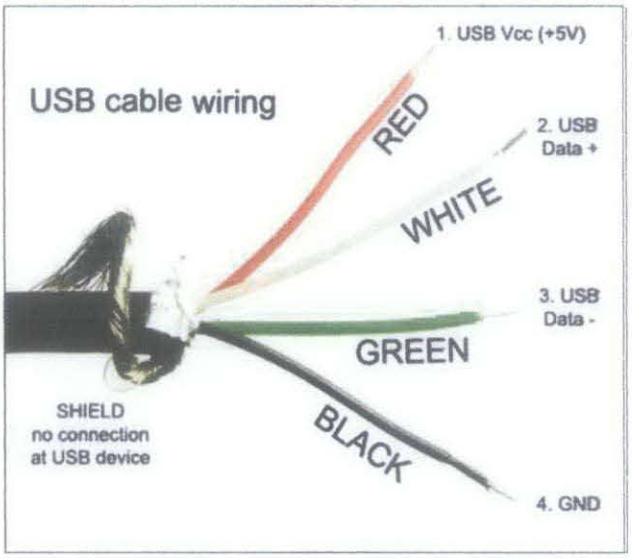

Figure 16: Obtain 5-volt from USB cable

The GPS receiver converter shows in Figure 17. This converter can help GPS receiver to send data to computer and receive 5-volt power supply from the USB port. Figure 18 shows how to interface GPS receiver and USB-Serial converter. Both USB connect to USB ports and GPS receiver's LED (red) should be light up.

Figure 17: The completed GPS receiver converter

Figure 18: The complete GPS receiver's connection

3.6 Read GPS Data

The Byonics GPS2 provides NMEA sentences which are $GPGSA, $GPRMC, $GPGGA and $GPGSV. The example of the GPS receiver's output shown in Table 4. Each type of sentences has different kind of information. The information separated by a comma sign. The $GPGSA, $GPRMC and $GPGGA will be used in the application since those sentences provided sufficient information to run the project. The application will read the data, split and sort it into the application's memory.

Table 4: Example of GPS receiver's output

$GPGGA,154220.000,0423.1711,N,10057.8293,E,1,06,1.3,90.6,M,-14.1,M,,0000*4E
$GPGSA,A,3,14,18,25,30,12,29,,,,,,,2.6,1.3,2.3*35
$GPGSV,3,1,12,09,59,086,,29,46,212,44,21,43,340,,14,27,245,28*71
$GPGSV,3,2,12,15,23,021,,18,23,334,26,25,17,182,37,12,16,149,18*7A
$GPGSV,3,3,12,30,07,212,36,02,05,136,,22,04,301,,05,02,081,*71
$GPRMC,154220.000,A,0423.1711,N,10057.8293,E,0.00,160.48,280911,,,A*66
$GPGGA,154221.000,0423.1711,N,10057.8293,E,1,06,1.3,90.6,M,-14.1,M,,0000*4F
$GPGSA,A,3,14,18,25,30,12,29,,,,,,,2.6,1.3,2.3*35
$GPRMC,154221.000,A,0423.1711,N,10057.8293,E,0.00,160.48,280911,,,A*67
$GPGGA,154222.000,0423.1711,N,10057.8293,E,1,06,1.3,90.6,M,-14.1,M,,0000*4C
$GPGSA,A,3,14,18,25,30,12,29,,,,,,,2.6,1.3,2.3*35
$GPRMC,154222.000,A,0423.1711,N,10057.8293,E,0.00,160.48,280911,,,A*64
$GPGGA,154223.000,0423.1711,N,10057.8293,E,1,06,1.3,90.6,M,-14.1,M,,0000*4D
$GPGSA,A,3,14,18,25,30,12,29,,,,,,,2.6,1.3,2.3*35

3.6.1 $GPGSA

It shows GPS Dillution of Precision (DOP) and active satellites. This sentence provides details about the nature of the fix. The DOP is the indication of the effect of satellite geometry on the accuracy of the fix. The explaination of this setence shown in Table 5. The information that used in this application is HDOP since this is for the vehicle application, the altitude accuracy can be ignored.

Table 5: $GPGSA sentence data [15]

$GPGSA,A,3,14,18,25,30,12,29,,,,,,,2.6,1.3,2.3*35		
Data Num	**Value**	**Description**
1	A	Mode: M=Manual, forced to operate in 2D or 3D A=Automatic, 3D/2D
2	3	Mode: 1=Fix not available 2=2D 3=3D
3-14	14,18,25,30,12,29,,,,,,,	Pseudo-random's (PRN) of Satellite Vechicles (SV's) used in position fix (null for unused fields)
15	2.6	Position Dilution of Precision (PDOP)
16	1.3	Horizontal Dilution of Precision (HDOP)
17	2.3*35	Vertical Dilution of Precision (VDOP)

3.6.2 $GPRMC

This sentence shows the GPS position, velocity and time data. It also called as The Recommended Minimum (RMC). Table 6 shows details for this sentence. The data for Variation and East/West value are not available for this GPS receiver.

Table 6: $GPRMC sentence data [15]

$GPRMC,154220.000,A,0423.1711,N,10057.8293,E,0.00,160.48,280911,,,A*66		
Data Num	**Value**	**Description**
1	154220.000	UTC time of fix 15:42:20
2	A	validity – A-ok, V-invalid
3	0423.1711	current Latitude, 04° 23.1771"
4	N	North/South
5	10057.8293	current Longitude, 100° 57.8293"
6	E	East/West
7	0.00	Speed in knots
8	160.48	True course
9	280911	Date Stamp 28th November 2011
10	-	Variation
11	-	East/West
12	A*66	checksum

3.6.3 $GPGGA

This is the most important NMEA sentences. It provide most of the data needed for basic GPS application. That include the current UTC time, latitude, longitude, number of sattelites, HDOP, altitude and others. Table 7 shows the details of this sentence.

Table 7: $GPGGA sentence data [15]

$GPGGA,154220.000,0423.1711,N,10057.8293,E,1,06,1.3,90.6,M,-14.1,M,,0000*4E		
Data Num	**Value**	**Description**
1	154220.000	UTC time of fix 15:42:20
2	0423.1711	current Latitude, 04° 23.1771"
3	N	North/South
4	10057.8293	current Longitude, 100° 57.8293"
5	E	East/West
6	1	Fix Quality: - 0 = Invalid - 1 = GPS fix - 2 = DGPS fix
7	06	Number of Satellites
8	1.3	Horizontal Dilution of Precision (HDOP)
9	90.6	Altitude
10	M	Meters
11	-14.1	Height of geoid above WGS84 ellipsoid
12	M	Meters
13	Blank	Time since last Differential GPS update
14	Blank	Differential GPS reference station id
15	0000*4E	Checksum

3.7 Graphical User Interface Design

The neat Graphical User Interface (GUI) design is important in order to make the user easier to read the important data from the application. The software that will help to design it is *Microsoft Visual Studio 2010*. The programming language that used to develop the application is *Visual Basic .NET*. It is versatile development software because it can perform a lot of tasks compare to others.

3.8 Interface Map with the application

There are several suggested ways to implement Google Map in Visual Basic (VB) on Desktop Application. One of the ways is to implement by linking the form in VB to the physical address or its latitude and longitude in the Google Map site.

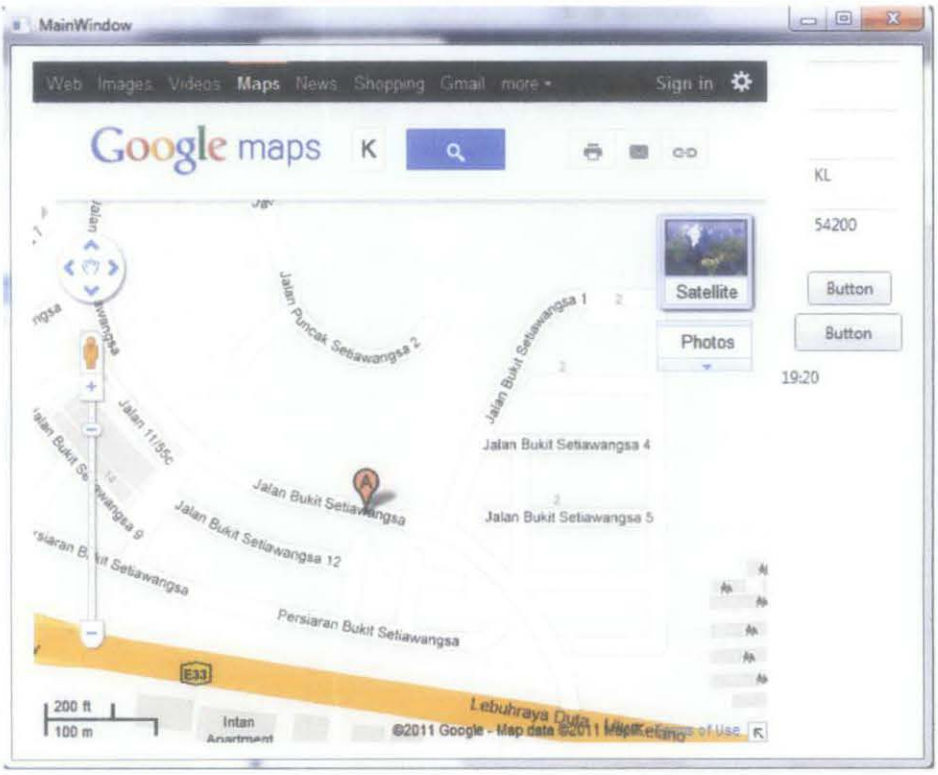

Figure 19: The output view from the application using physical address

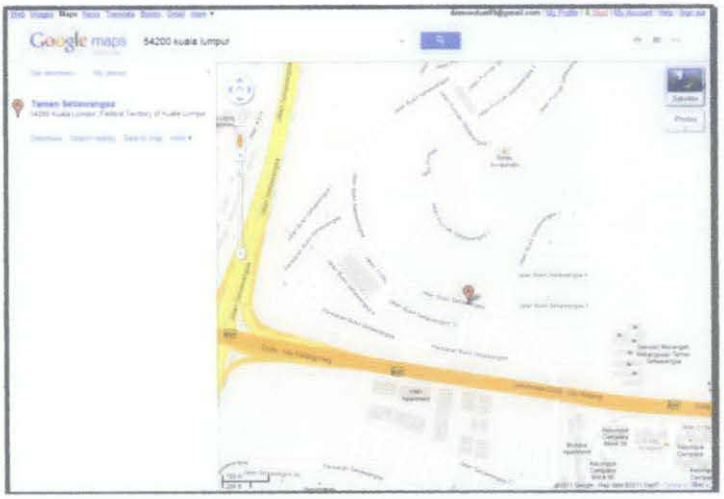

Figure 20: Google Map view via internet browser

There is a fast approach to interface Google Map with the application which by using web browser toolbox in VB. Figure 19 shows the output of the implementation. The output for Figure 19 is almost the same with Figure 20. The comparison shows us that using fast approach will have the output same with the normal Google Map and there is nothing we can do to customize the map output features as we like.

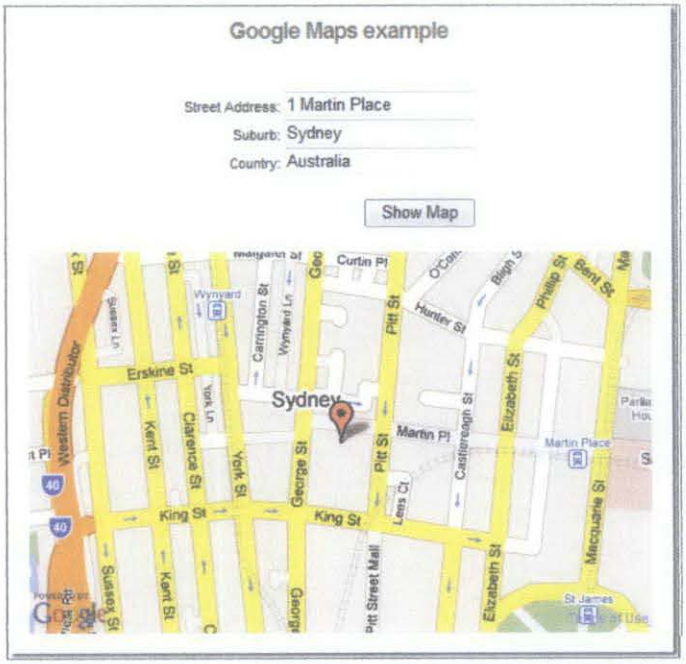

Figure 21: The output view from application using Google Maps API

Another way of doing it is by using Google Map Application Programming Interface (API). Google Maps API has greater control of the map. Google Map API offers custom markers on the map, several markers placement and can show us only the map that we needed without browser-like view. The design will be much neat and

a lot of customization can be implemented as shown in Figure 21. This approach is harder to develop but will make your map viewer much more dynamic. In this project, Google Maps API will be used for the application.

3.9 Network Interface Implementation

To do some demonstration for this project, network interface should be implemented to make this application able to broadcast the warning message to other vehicles. This can be achieved by implementing User Datagram Protocol (UDP) network protocol. UDP able to send messages to other hosts on an Internet Protocol (IP) network without requiring prior communications to set up special transmission channels or data paths. One of the reasons why UDP is used for this application is because of the ability to support packet broadcasts by sending packets to all hosts on local network. IEEE 802.11g or 802.11n will be used to broadcast the warning message.

3.10 Programming Validation and Functionality Check

After the network interface implementation phase done, the application need to be tested. The criteria that need to be taken into account are:

a) The stability of the application (application not crash)
b) Application shows correct data
c) User easy to interact with the application
d) Manage to broadcast warning message to others

This phase will help to detect if any vital error occurring the application. The error will be fixed immediately and the tests will be run again. The cycle of fixing and testing will keep going until there is no error detected during the test.

3.11 Transmit and Receive The Message

In order to transmit the message, the application needs to be connected to GPS receiver, internet connection, sensors to detect the abnormal condition and network adapter. The GPS receiver will provide the current coordinate of the vehicle, internet connection needed to obtain the weather information of the current place and sensors to trigger the detection of abnormal condition. The application will process those data and message will be broadcasted via network adapter. The block diagram for message transmission shows in Figure 22.

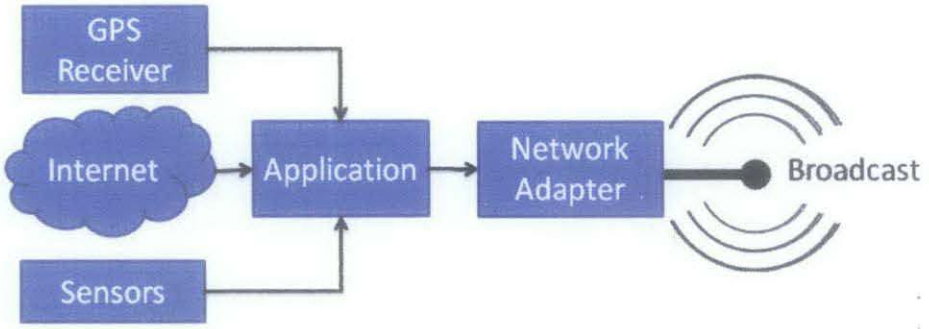

Figure 22: Block diagram transmit message

For receiving the warning message, the adapter will keep listening to any broadcast that transmitted in the selected port number. The GPS receiver provides the current coordinate in order to compare it with the received warning message coordinate. When application evaluated the warning message received, the warning message will be displayed on the screen with a beeping sound. Figure 23 shows the block diagram when receiving the warning message.

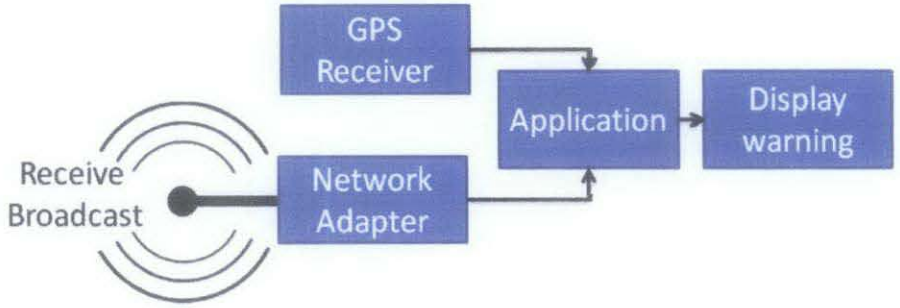

Figure 23: Block diagram receive message

CHAPTER 4

RESULT & DISCUSSION

Figure 24: The application Graphical User Interface

Throughout the project activities as described in the project methodology, the information has been acquired. In this section, we will be discussed about the application and how it works. Figure 24 shows the Graphical User Interface of the application.

The example of the situation when a vehicle is broadcasting a warning message is shown in Figure 25. The vehicle will broadcast the message whenever an abnormal environment condition detected and all the vehicles moving in the same direction will receive the warning. The received vehicle will alert the driver about the abnormal condition ahead.

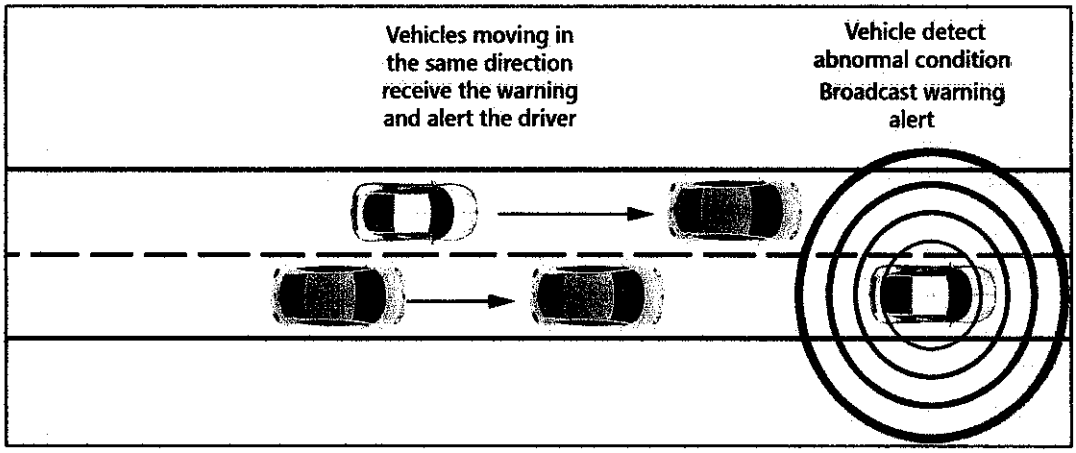

Figure 25: Vehicle Broadcast and receive the warning message

This application implemented in the vehicles and needs to be connected to the network. The GPS receiver needs to be mounted on the vehicle. Then, start the application by pressing "Connect/Disconnect" button from Communication Port section as in Figure 26. Whenever vehicle senses any abnormal conditions, it will immediately broadcast the warning message to other vehicles across the network.

When the other vehicles received the warning message, a beeping sound will emit indicating there is an abnormal environment condition ahead. The driver will be alerted of this situation and will drive more careful, thus the accident can be avoided.

The application is divided into several sections. Each section has its own step to achieve the objectives. Those sections are:

a) Communication Port selection
b) Date and Time
c) Satellite
d) Coordinate
e) Vehicle Velocity
f) Temperature & Weather Forecast
g) Wind speed
h) Map
i) Broadcast Message
j) Warning Message Receive

4.1 Communication Port selection

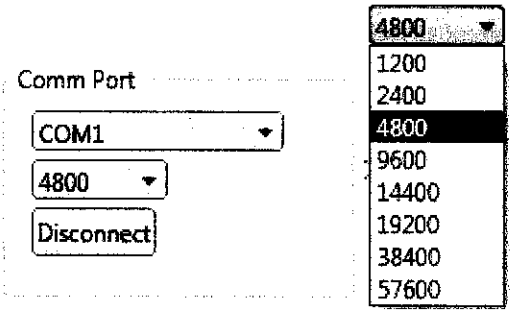

Figure 26: Communication Port Selection

To start the communication between application and the GPS receiver, the port number and baud rate of the serial port need to be selected according to the GPS receiver specification. The selection has to be made by choosing appropriate port and baud rate from the combo box that can be seen in Figure 26. This application will try to check which serial port that available and show it on the combo box selection. The GPS receiver connected on COM1 serial port with baud rate 4800. Table 8 shows the code fragment for communication port selection.

Table 8: Code fragment for Communication Port Selection

```
    Private Sub ShowComportSpeeds()
        combobaudrate.Items.Clear()
        combobaudrate.Items.Add("1200")
    combobaudrate.Items.Add("2400")
    combobaudrate.Items.Add("4800")
    combobaudrate.Items.Add("9600")
    combobaudrate.Items.Add("14400")
    combobaudrate.Items.Add("19200")
    combobaudrate.Items.Add("38400")
    combobaudrate.Items.Add("57600")
End Sub

Private Sub ShowComports()
    combocomport.Items.Clear()
    For Each sp As String In My.Computer.Ports.SerialPortNames
      combocomport.Items.Add(sp)
    Next
    If combocomport.Items.Count > 0 Then
          combocomport.SelectedIndex = combocomport.Items.Count – 1
          combobaudrate.SelectedIndex = 2
        End If
      End Sub
```

Table 9: Code fragment for Date Time

```
Public Sub time_utc(ByVal utc_input As String, ByVal utc_date As String)

Dim gpstime_array() As Char = utc_input.ToCharArray()
Dim gpsdate_array() As Char = utc_date.ToCharArray()
Dim utc_temp As String = "20" & gpsdate_array(4) & gpsdate_array(5) & "/" & gpsdate_array(2) & gpsdate_array(3) & "/" &
gpsdate_array(0) & gpsdate_array(1) & " " & gpstime_array(0) & gpstime_array(1) & ":" & gpstime_array(2) &
gpstime_array(3) & ":" & gpstime_array(4) & gpstime_array(5)
nowsendtime = utc_temp
Dim datetime_utc As DateTime = DateTime.Parse(utc_temp)
datetime_utc = datetime_utc.AddHours(system_gmt)
satellite_datetime = datetime_utc
nowtime_hour.Content = datetime_utc.ToString("hh")
nowtime_minutes.Content = datetime_utc.ToString("mm")
nowtime_ampm.Content = datetime_utc.ToString("tt")
nowday.Content = datetime_utc.ToString("dddd")
nowdate.Content = datetime_utc.ToString("dd")
now_month.Content = datetime_utc.ToString("MMMM")
now_year.Content = datetime_utc.ToString("yyy")
End Sub
```

4.3 Satellite

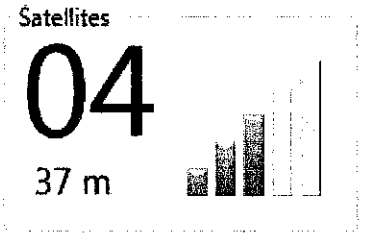

Figure 29: Satellite and accuracy display

In this section, the focus will be the information on the number of satellites locked and the accuracy of the coordinate. The number of satellites obtained from the $GPGGA sentence. This will help user to know how many satellites tracked by the GPS receiver. In Figure 29, there is a signal accuracy bar that shows how good the accuracy of the position is. The signal bar is depending on the value of the HDOP received. Table 10 shows the value of HDOP that will determine the signal bar display. Table 11 shows the code fragment to view the GPS signal accuracy bar [16].

Table 10: HDOP Signal Bar range [16]

Signal Bar	HDOP range
0	>= 20
1	>=9
2	>=7
3	>=6
4	>=3
5	>0

The flow on how to fetch the satellite number and HDOP of the signal are shown in Figure 30. The application will retrieve satellite number and HDOP from the GPS receiver data. Then, use the HDOP value to calculate the accuracy distance of the coordinate as in (2) and view it on the application's interface.

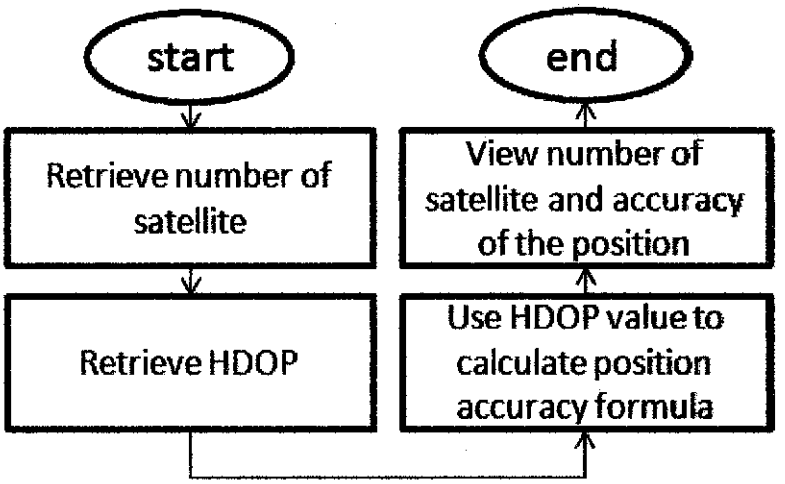

Figure 30: Flowchart for satellite and accuracy section

Table 11: Code fragment in determine the signal accuracy

```
Public Sub gps_signal_strength(ByVal valhdop As Decimal)
    Dim gpspic As String

    If valhdop >= 20 Then
      gpspic = "bar00.png"
    ElseIf valhdop >= 9 Then
      gpspic = "bar20.png"
    ElseIf valhdop >= 7 Then
      gpspic = "bar40.png"
    ElseIf valhdop >= 6 Then
      gpspic = "bar60.png"
    ElseIf valhdop >= 3 Then
      gpspic = "bar80.png"
    ElseIf valhdop > 0 Then
      gpspic = "bar100.png"
    Else
      gpspic = "bar00.png"
    End If

    Dim gps_signal_output As New BitmapImage
    gps_signal_output.BeginInit()
    gps_signal_output.UriSource = New Uri(CurDir() & "\Resources\" & gpspic)
    gps_signal_output.EndInit()
    image_gps_signal.Source = gps_signal_output

  End Sub
```

4.4 Coordinate

Figure 31: Current Coordinate display

This is one of the important sections for this application. It will keep showing the current coordinate of the vehicle. The coordinate obtained from $GPGGA message and will be used for sending warning message, viewing the map and retrieving weather condition.

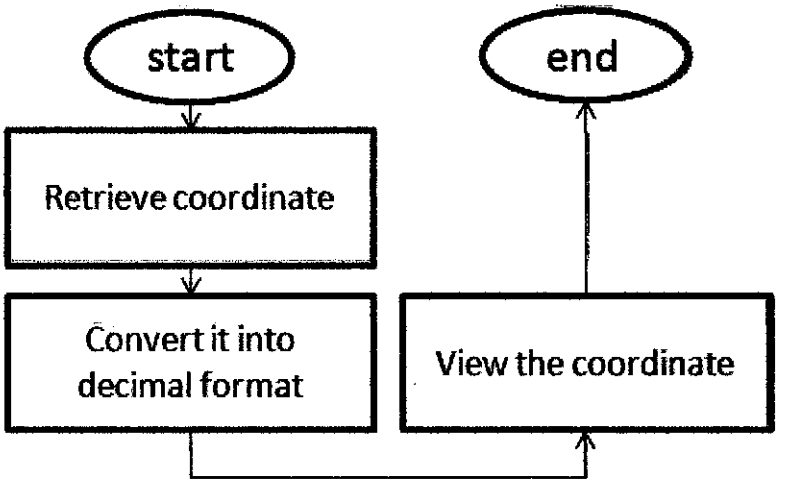

Figure 32: Flowchart to retrieve and view the coordinate

Figure 32 shows the flowchart in retrieving the current coordinate. It receives the coordinate in the degree and minute format. It needs to be converted into decimal format for the use of the Google Map and Google Weather in obtaining the information.

4.5 Vehicle Velocity

Figure 33: Velocity display

This feature will help user to know the velocity of the vehicle from the GPS receiver. The velocity is given in Knot unit and need to be converted into km/h. The conversion formula from Knot to km/h is shown in (3).

$$Velocity\ (km/h)\ =\ (velocity\ in\ Knot)\ x\ 1.852 \qquad (3)$$

4.6 Temperature & Weather Forecast

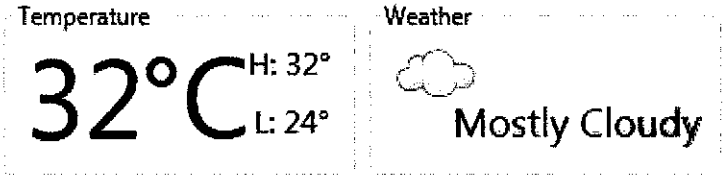

Figure 34: Temperature and Weather Display

The temperature and weather forecast is obtained by using Google Weather API that will give us a raw of weather data and need further processing it in order to retrieve interested information. The weather and temperature will show on the application as in Figure 34. The programming flow chart on how to retrieve the weather condition and temperature located in Figure 35.

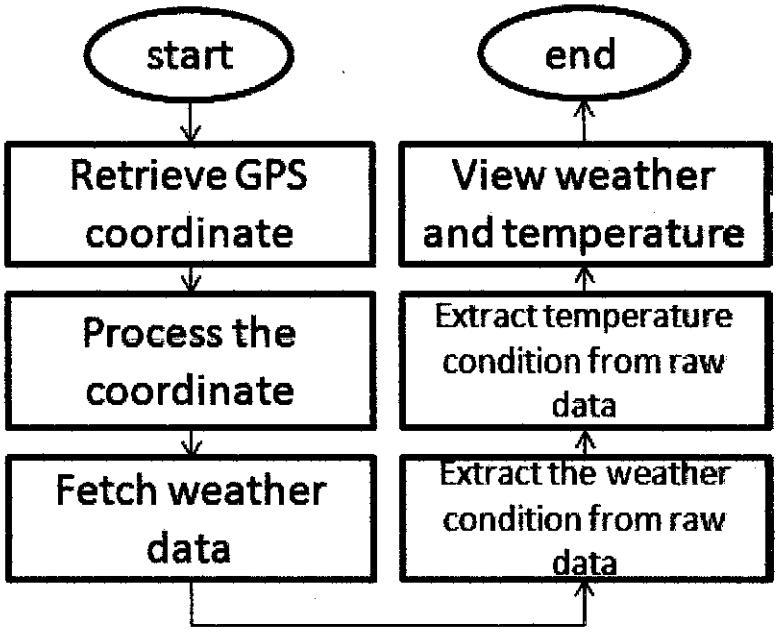

Figure 35: Flowchart to fetch temperature and weather condition

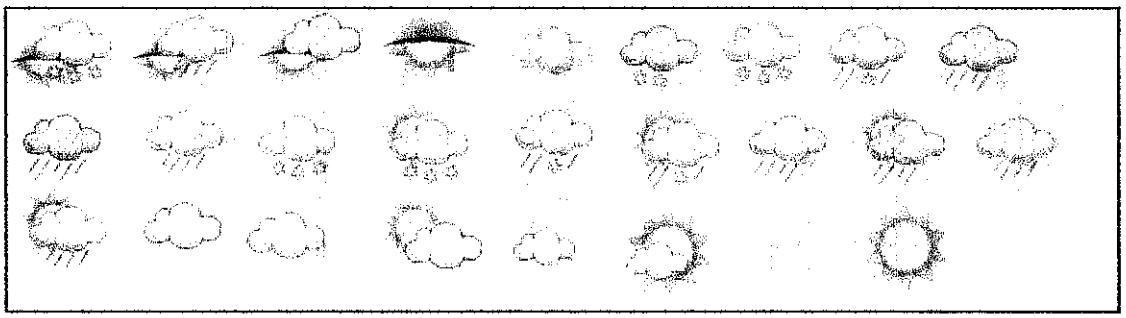

Figure 36: The pictures for weather condition icon

The weather update also will include a picture that will represent the current weather condition. It keeps changing the picture as the condition change. Figure 36 shows the list of weather pictures.

4.7 Wind Speed

Figure 37: Wind speed Display

The current wind speed condition is also taken from the Google Weather API. It shows direction and speed of the wind for current position of the vehicle. The information will show on the application as in Figure 37.

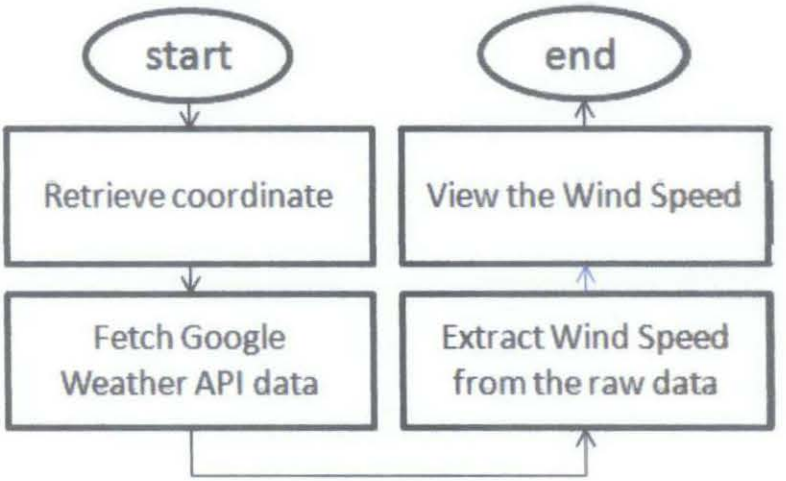

Figure 38: Flowchart to determine the wind speed for current location

The flow of the process in obtaining and viewing the wind speed condition shown in Figure 38. It will fetch the Google Weather API data from the current vehicle coordinate. Then the application will extract the wind speed data from the Google Weather API and view it on the screen.

4.8 Map

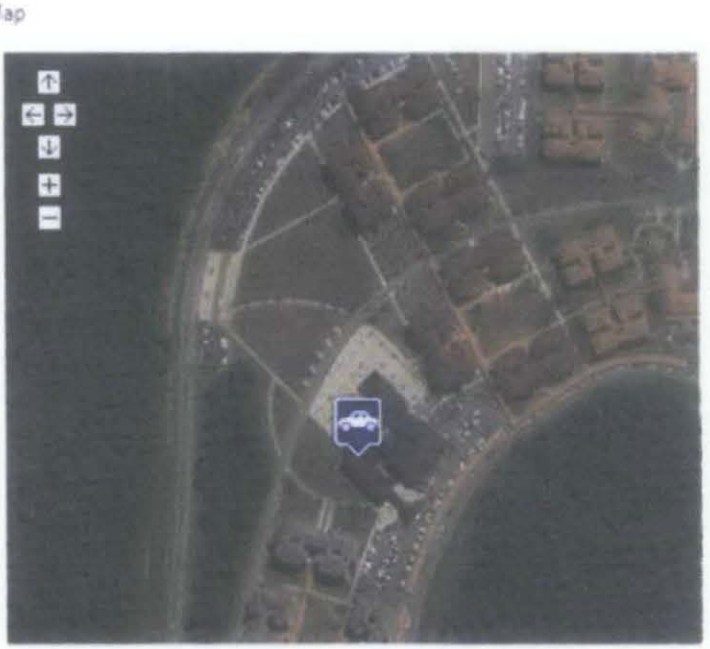

Figure 39: Map for current location

Since the Google Map API uses JavaScript, map cannot directly embed its code in Visual Basic. There is a need to create a webpage that will include all the JavaScript needed for the application. Figure *39* shows the current location of vehicle.

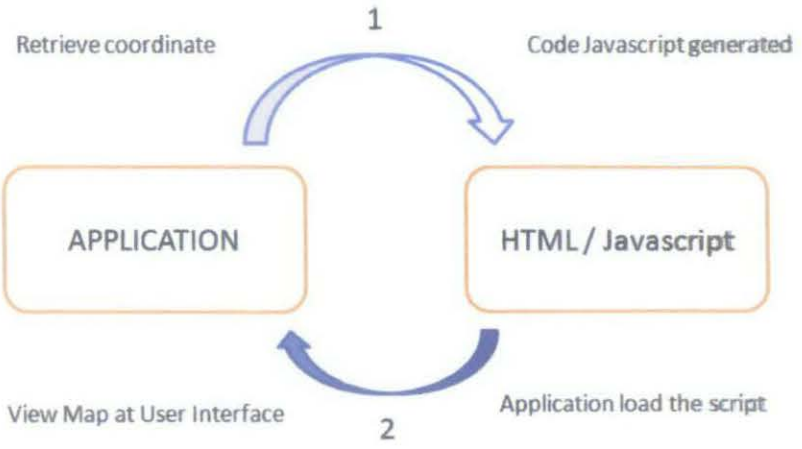

Figure 40: Flow how the application view the map

From the Figure 40, the application will fetch the coordinate that wants to be viewed. Then, the application will create a HTML file containing JavaScript that is able to view the map. The application will load the HTML file and view it at the user interface. Finally, the map location with markers is displayed at the user interface.

4.9 Broadcast Message

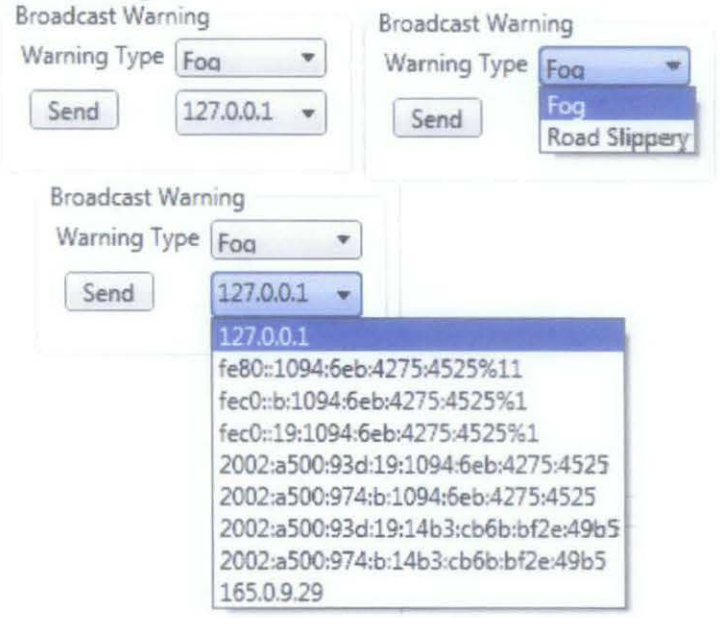

Figure 41: Broadcast Warning section

Since there is no sensor to detect abnormal environment condition, this broadcast warning section will help to trigger the application as if the sensor detects the abnormal environment condition. In Figure 41, there are two combo box that user need to configure. Those are type of warning and network interface selection. To start broadcast the message, user need to choose the type of warning from the combo box, the suggested type of warning are Fog and Road slippery selection. Then user needs to decide which network adapter that will be used to broadcast the message. The network interface selection is important because a machine might have more than one network interface and need to select the appropriate network interface instead using the default network interface. The information that will be broadcasted are the current coordinate, the current date time (in UTC) and the type of warning message. It will be sent by using UDP protocol and broadcast it to the network. The port number for this broadcast is set to "7777" by default.

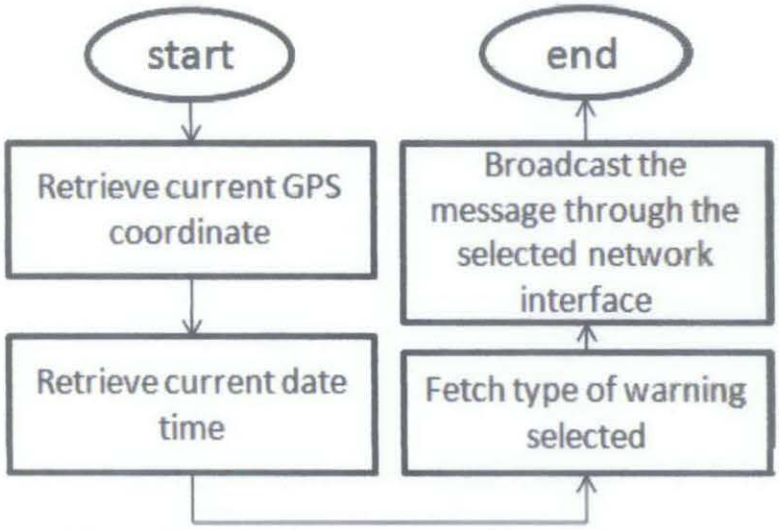

Figure 42: Flowchart how to broadcast the message

The flow of the broadcast process is shown in Figure 42. It will retrieve the GPS coordinate, current date time and type of warning. Then, all the info will be broadcasted through the selected network interface for other vehicles to receive the warning message.

4.10 Warning Message Receive

Figure 43: Message Receive Display

When the application fired up, it starts to listen for any kind of warning message at port "7777" that sent by other hosts in the network. Whenever a message received, the raw message will be shown in the Message Receive textbox as shown in Figure 43. The raw message will be submitted into the functions to process the data. The function for this process is shown in Table 12. After that, it will view the warning message with a "beep" sound to alert the user. Figure 44 shows the flowchart that the

application executes whenever it receives the warning message. The user can clear the warning message by just click at the "clear" button.

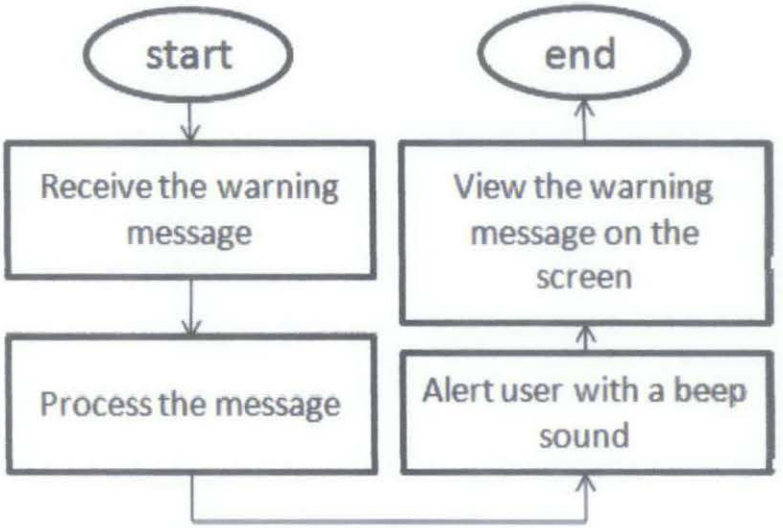

Figure 44: Flowchart process when receive warning message

Table 12: Code fragment to process received warning message

```
Public Sub udp_dataprocess(ByVal input As String)
    Dim udp_message_array() As String = Split(input, ",")
    Dim udp_datetime_receive As DateTime = DateTime.Parse(udp_message_array(3))


    udp_datetime_receive = udp_datetime_receive.AddHours(system_gmt)
    txt_display_warning.Content = "Warning"

    If InStr(udp_message_array(0), "Fog") Then
        txt_display_condition.Content = "Fog Ahead"
        txt_display_advice.Content = "Reduced Visibility"
        txt_display_date.Content = udp_datetime_receive.ToString("hh mm tt")
        txt_display_time.Content = udp_datetime_receive.ToString("dd/MM/yyyy")
    End If


    If InStr(udp_message_array(0), "Road Slippery") Then
        txt_display_condition.Content = "Wet Road Ahead"
        txt_display_advice.Content = "Road Slippery"
        txt_display_date.Content = udp_datetime_receive.ToString("hh mm tt")
        txt_display_time.Content = udp_datetime_receive.ToString("dd/MM/yyyy")
    End If

    Array.Clear(udp_message_array, 0, udp_message_array.Length - 1)
End Sub
```

The application is now able to update the data itself. As for temperature and weather condition, it will keep update the location for every 30 seconds while the date and time section will be updated every second. The timing for update is done by using task scheduler in Visual Basic.

This application provides configuration file that will able the user to change some vital configuration for the application. The example of the configuration file shown in Table 13. The user just needs to edit the value in <appSettings> section.

Table 13: Configuration file

```
<appSettings>
  <add key ="GMT" value="+8"/>
  <add key ="SYSTEMSOCKET" value="7777"/>

</appSettings>
```

The project is completed within the range of time and successfully implemented and demonstrated. The GPS receiver implemented with a laptop and connected to the internet by using broadband connection for weather forecast. The application will able to detect the abnormal condition, warn the driver of the abnormal condition and spread the warning message to other laptop.

CHAPTER 5

CONCLUSION & RECOMMENDATION

5.1 CONCLUSION

The number of vehicle produced keep increasing but the road improvement not following the trend and it will cause the vehicles more prone for accidents. The lack of awareness of the driver while driving a vehicle contributed a lot in the daily rate of road accident. A comprehensive and reliable warning system is a need for every vehicle in order to help driver to be more aware of his surroundings. The warning system need to be able to warn the driver if any abnormality detected ahead. Thus, it can helps in lack of human reflex time in avoiding the accident.

The application for the warning system needs to have several features that will be able to assist the driver in knowing his driving environment. It needs to be able to detect any abnormal condition around it and warn the driver. Then, the message need to be broadcasted to other vehicle so they could modify his vehicle to the appropriate speed. Map included in the application to help driver estimate the distance between his location and abnormal condition. The weather forecast, wind speed and temperature will help the driver in preparing any bad weather situation along his journey.

The daily rate of accident occurring right now will be significantly decreased if a comprehensive and reliable warning system is implemented in the vehicles. This project is a head start in developing any robust vehicle warning system that will make our driving experience much safer. This project will assist a lot of research in VANETs for them to implement the communication layer that they have been developing right now into the system.

5.2 RECOMENDATION

For future implementation of this project, the application needs to be more than just warning application. The implementation of this application with the existing GPS devices for vehicle such as Garmin GPS will make it more dynamic. Most of the new vehicles have built-in GPS devices and if those devices integrated with this kind of safety warning application and sensors, it can make the navigation vehicle more robust. Besides that, the access of the internet connection for vehicle in the future will widely implement with the existing of 3G connection and WIMAX will boost the flexibility of the application. Thus, it will allow in enhancing the dynamic usage of the application for safety purposes.

The other development on VANETs is needed to improve our project such as the interpretation of driver behavior detection if he is tired, sleepy or drunk during driving. If all those kind of development can be integrated into a system, it will help to create robust and reliable warning application for safety of the road users.

REFERENCES

[1] Malaysia Automotive Info. "*Summary of Sales & Production Data,*" Internet: http://www.maa.org.my/info_summary.htm, 2011 [June 7th 2011].

[2] F. Küçükay, J. Bergholz, "Driver assistant systems," Int. Conf. on Automotive Technologies, Istanbul, Turkey, 2004.

[3] Y. Sahat. "*17 maut di jalan raya setiap hari.*" Internet: http://www.utusan.com.my/utusan/info.asp, June. 19, 2011 [Aug. 20, 2010]

[4] World Health Organization. "*Global Plan for the Decade of Action for Road Safety 2011-2020*" Internet: http://www.who.int/roadsafety/decade_of_action/plan/en/index.html, 2011 [Dec. 21, 2011]

[5] R. Bishop, *Intelligent Vehicle Technology and Trends*, Norwood, MA: Artech House, 2005, pp. 14-15.

[6] Z. Y. Rawashdeh and S. M. Mahmud, "Communication in Vehicle Networks," in *Mobile Ad-Hoc Networks: Applications*, 1st ed, X. Wang, Rijeka, Croatia: InTech, 2011, pp. 20-21

[7] Y. Qian, and N. Moayeri, "Design Secure and Application-Oriented VANETs", Proceedings of IEEE VTC'2008-Spring, Singapore, May 11-14, 2008

[8] K. Bilstrup, A. Bohm, K. Lidstrom, M. Jonsson, T. Larsson, L. Stranden, and H. Zakizadeh, "Vehicle alert system," in Proc. of the 14th World Congress on Intelligent Transport Systems, Beijing, China, Oct. 2007.

[9] J. Stefan, Navigating With GPS, *Circuit Cellar*, Issue. 123, Oct.

[10] GARMIN. "*Garmin | What is GPS?,*" Internet: http://www8.garmin.com/aboutGPS [Oct. 24, 2011]

[11] P. Bennet, ""*The NMEA Faq*", Internet: http://vancouver-webpages.com/peter/nmeafaq.txt, Jan. 30, 2008 [Dec. 28, 2011]

[12] J. Person. "*Writing GPS Applications*" Internet: http://drdobbs.com/embedded-systems/229100152, Jan. 01, 2005 [Oct. 24, 2011].

[13] BYONICS. "*Byonics GPS Receivers*", Internet: http://www.byonics.com/tinytrak/gps.php, [Dec. 29, 2011]

[14] VK6ABA Site. *"In Making converter to use the Byonics GPS2 on a PC,"* Internet: http://home.iprimus.com.au/kat2009/gps2.htm, [**Oct. 28, 2011**]

[15] G. Baddeley,. *"Glenn Baddeley - GPS - NMEA Sentence Information,"* Internet: http://home.mira.net/~gnb/gps/nmea.html, May 25,2011 [Oct. 28, 2011]

[16] J. Person. *"Writing Your Own GPS Applications: Part 2,"* Internet: http://www.developerfusion.com/article/4652/writing-your-own-gps-applications-part-2/2/, 2008 [Dec. 20, 2011]

APPENDIX A

APPLICATION PROGRAMMING CODE

MainWindow.xaml.vb

(MICROSOFT VISUAL STUDIO 2010)

```
Imports System.Collections.Generic
Imports System.Linq
Imports System.Text
Imports System.Windows
Imports System.Windows.Controls
Imports System.Windows.Data
Imports System.Windows.Documents
Imports System.Windows.Input
Imports System.Windows.Media
Imports System.Windows.Media.Imaging
Imports System.Windows.Navigation
Imports System.Windows.Shapes
Imports System.IO.Ports
Imports System.Threading
Imports System.Windows.Threading
Imports System.IO
Imports System.Net.Sockets
Imports System
Imports System.Net
Imports System.ComponentModel
Imports Microsoft.Win32
Imports System.Configuration
Imports System.Xml.XPath
Imports System.Xml



Namespace Serial_Communication_WPF
  Public Class MainWindow



    Public Sub time_timer(ByVal sender As Object, ByVal e As EventArgs)
      updatetime()
      previous_globallatitude = globallatitude
      previous_globallongitude = globallongitude
      CommandManager.InvalidateRequerySuggested()
    End Sub

    Public Sub temperature_timer(ByVal sender As Object, ByVal e As EventArgs)
      Try
        temperature(globallongitude, globallatitude)
      Catch ex As Exception

      End Try

      CommandManager.InvalidateRequerySuggested()
    End Sub


    Private Sub gettime()
      ' nowtime.Content = DateTime.Now.ToString("hh:mm:ss tt")
      nowday.Content = DateTime.Now.ToString("dddd")
      nowdate.Content = DateTime.Now.ToString("dd MMM yyy")
    End Sub
```

```
    Private Sub updatetime()
        gps_signal_strength(hdop)
        txthdop.Content = Math.Round(hdop)
        time_elapsed()
    End Sub


    Private Sub ShowComportSpeeds()
        combobaudrate.Items.Clear()
        combobaudrate.Items.Add("1200")
        combobaudrate.Items.Add("2400")
        combobaudrate.Items.Add("4800")
        combobaudrate.Items.Add("9600")
        combobaudrate.Items.Add("14400")
        combobaudrate.Items.Add("19200")
        combobaudrate.Items.Add("38400")
        combobaudrate.Items.Add("57600")
    End Sub

    Private Sub ShowComports()
        combocomport.Items.Clear()
        For Each sp As String In My.Computer.Ports.SerialPortNames
            combocomport.Items.Add(sp)
        Next
        If combocomport.Items.Count > 0 Then
            combocomport.SelectedIndex = combocomport.Items.Count - 1
            combobaudrate.SelectedIndex = 2
        End If
    End Sub

    Private Sub showipaddress()
        Dim ipaddress() As Net.IPAddress = System.Net.Dns.GetHostAddresses("")
        combonetwork.Items.Add("127.0.0.1")
        combonetwork.SelectedIndex = 0
        For Each fsfs As Net.IPAddress In ipaddress
            combonetwork.Items.Add(fsfs.ToString)
        Next

    End Sub

#Region "variables"
    Private mcFlowDoc As New FlowDocument()
    Private para As New Paragraph()
    'Serial
    Private serial As New SerialPort()
    Private recieved_data As String
    Private new_buffer As Boolean = 0
    Private buffer As String
    Private globallatitude As String
    Private globallongitude As String
    Private firstdate As DateTime
    Private seconddate As DateTime
    Private satellite_datetime As DateTime
    Private nowsendtime As String
    Public initialnetwork As String = "127.0.0.1"
    Dim system_gmt As String = ConfigurationManager.AppSettings("GMT")
    Private hdop As Double = 0
    Private previous_globallatitude As String
    Private previous_globallongitude As String
    Private receive_warninglatitude As String
    Private receive_warninglongitude As String
```

```
#End Region

    Public Sub Window_Loaded(ByVal sender As System.Object, ByVal e As System.Windows.RoutedEventArgs) _
    Handles MyBase.Loaded

        Dim dt_timer As DispatcherTimer = New DispatcherTimer()
        Dim dt_timer_temperature As DispatcherTimer = New DispatcherTimer()
        Dim dt_timer_udpreceiver As DispatcherTimer = New DispatcherTimer()
        AddHandler dt_timer_udpreceiver.Tick, AddressOf receivemessage
        AddHandler dt_timer.Tick, AddressOf time_timer
        AddHandler dt_timer_temperature.Tick, AddressOf temperature_timer
        dt_timer_temperature.Interval = New TimeSpan(0, 0,
ConfigurationManager.AppSettings("temp_update_second"))
        dt_timer.Interval = New TimeSpan(0, 0, 1)
        dt_timer_udpreceiver.Interval = New TimeSpan(0, 0, 0, 0, 1)
        dt_timer_temperature.Start()
        dt_timer.Start()
        dt_timer_udpreceiver.Start()
        showipaddress()
        ShowComportSpeeds()
        ShowComports()
        Connect_btn.Content = "Connect"
        gps_signal_strength(Convert.ToDecimal(txtsatellite.Content))
        firstdate = DateTime.Now
        time_elapsed()
        updatetime()



    End Sub

    Private Sub Connect_Comms(ByVal sender As Object, ByVal e As RoutedEventArgs)
        If Connect_btn.Content = "Connect" Then
            'Sets up serial port
            serial.PortName = combocomport.Text
            serial.BaudRate = Convert.ToInt32(combobaudrate.Text)
            serial.Open()

            'Sets button State and Creates function call on data recieved
            Connect_btn.Content = "Disconnect"

            AddHandler serial.DataReceived, New System.IO.Ports.SerialDataReceivedEventHandler(AddressOf Recieve)
        Else
            Try
                ' just in case serial port is not open could also be acheved using if(serial.IsOpen)
                serial.Close()
                Connect_btn.Content = "Connect"
            Catch
            End Try
        End If
    End Sub

#Region "Recieving"

    Private Delegate Sub UpdateUiTextDelegate(ByVal text As String)
```

```
    Private Sub Recieve(ByVal sender As Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs)
        ' Collecting the characters received to our 'buffer' (string).
        'recieved_data = serial.ReadExisting()
        Dim data_r As String = serial.ReadExisting()
        Dispatcher.Invoke(DispatcherPriority.Send, New UpdateUiTextDelegate(AddressOf WriteData), data_r)

    End Sub

    Private Sub WriteData(ByVal buff As String)
        ' Assign the value of the recieved_data to the RichTextBox.
        'para.Inlines.Add(buff)
        ' mcFlowDoc.Blocks.Add(para)
        ' Commdata.Document = mcFlowDoc

        If InStr(buff, "$") Then

            process_gps_data(buffer)
            buffer = buff

        Else
            buffer = buffer & buff
        End If
    End Sub

#End Region

#Region "GPS_data_processing"
    Private Sub process_gps_data(ByVal gps_input As String)
        Dim gps_input_array() As String = Split(gps_input, ",")
        Dim templatitude As Double
        Dim templongitude As Double
        'Using writer As StreamWriter = New StreamWriter("log.txt", True)
        'writer.WriteLine(gps_input)
        'End Using

        Try
            If InStr(gps_input_array(0), "$GPGGA") Then

                templatitude = gps_input_array(2)
                templongitude = gps_input_array(4)
                txtsatellite.Content = gps_input_array(7)
                templatitude = Format(Convert.ToDouble(coordinate_todecimals(templatitude)), "0.000000")
                templongitude = Format(Convert.ToDouble(coordinate_todecimals(templongitude)), "0.000000")
                txtgpslatitude2.Content = gps_input_array(3) & " " & templatitude
                txtgpslongitude2.Content = gps_input_array(5) & " " & templongitude

                If InStr(gps_input_array(3), "S") Then
                    globallatitude = templatitude * -1
                Else
                    globallatitude = templatitude
                End If

                If InStr(gps_input_array(5), "W") Then
                    globallongitude = templongitude * -1
                Else
                    globallongitude = templongitude
                End If



            ElseIf InStr(gps_input_array(0), "$GPRMC") Then

                time_utc(gps_input_array(1), gps_input_array(9))
```

```
            If (gps_input_array(7) = "") Then
               txtvelocity.Content = txtvelocity.Content
            Else
               txtvelocity.Content = Math.Floor(gps_input_array(7) * 1.852)
            End If

         ElseIf InStr(gps_input_array(0), "$GPGSA") Then

            If (gps_input_array(16) = "") Then
               txt_accuracy.Content = txt_accuracy.Content
            Else
               hdop = gps_input_array(16)
               txt_accuracy.Content = Math.Round(hdop * 6) & " m"
            End If
         End If




      Catch ex As IndexOutOfRangeException


      End Try

      Array.Clear(gps_input_array, 0, gps_input_array.Length - 1)

   End Sub




#End Region


#Region "functions"
   Private Function coordinate_todecimals(ByVal inputcoordinate As String) As Double

      'Pos="5601.0318"
      Dim PosDb As Double = inputcoordinate
      Dim Deg As Double = Math.Floor(PosDb / 100)
      Dim DecPos As Double = Math.Round(Deg + ((PosDb - (Deg * 100)) / 60), 5)
      Return DecPos '=56.0172

   End Function


   Public Function GetDistanceBetweenPoints(ByVal lat1 As Double, ByVal long1 As Double, ByVal lat2 As Double,
ByVal long2 As Double) As Double
      Dim distance As Double = 0
      Dim dLat As Double = ((lat2 - lat1) / (180 * Math.PI))
      Dim dLong As Double = ((long2 - long1) / (180 * Math.PI))
      Dim a As Double = ((Math.Sin((dLat / 2)) * Math.Sin((dLat / 2))) _
            + (Math.Cos(lat2) _
            * (Math.Sin((dLong / 2)) * Math.Sin((dLong / 2)))))
      Dim c As Double = (2 * Math.Atan2(Math.Sqrt(a), Math.Sqrt((1 - a))))
      'Calculate radius of earth
      Dim radiusE As Double = 6378135
      ' Equatorial radius, in metres
      Dim radiusP As Double = 6356750
      ' Polar Radius
```

```
        'Numerator part of function
        Dim nr As Double = Math.Pow((radiusE _
                * (radiusP * Math.Cos((lat1 / (180 * Math.PI))))), 2)
        'Denominator part of the function
        Dim dr As Double = (Math.Pow((radiusE * Math.Cos((lat1 / (180 * Math.PI)))), 2) + Math.Pow((radiusP *
Math.Sin((lat1 / (180 * Math.PI)))), 2))
        Dim radius As Double = Math.Sqrt((nr / dr))
        'Calaculate distance in metres.
        distance = (radius * c)
        Return distance
    End Function


    Public Function distance(ByVal lat1 As Double, ByVal lon1 As Double, ByVal lat2 As Double, ByVal lon2 As
Double) As Double
        Dim dlon As Double = deg2rad(lon1) - deg2rad(lon2)
        Dim dlat As Double = deg2rad(lat1) - deg2rad(lat2)
        Dim dist As Double

        dist = (Math.Sin(dlat / 2)) ^ 2 + Math.Cos(lat1) * Math.Cos(lat2) * (Math.Sin(dlon / 2)) ^ 2
        dist = 2 * Math.Asin(Math.Min(1, Math.Sqrt(dist)))
        dist = 6367 * dist

        Return dist
    End Function

    Private Function deg2rad(ByVal deg As Double) As Double
        Return (deg * Math.PI / 180.0)
    End Function

    Private Function rad2deg(ByVal rad As Double) As Double
        Return rad / Math.PI * 180.0
    End Function


#End Region


#Region "weather & temperature"
    Public Sub temperature(ByVal getlong As String, ByVal getlati As String)

        Dim readweather As XmlReader
        Dim downxml As String
        Dim src As String
        Dim webclient As New Net.WebClient
        Dim inputlatitude As Long = getlati * 1000000
        Dim inputlongitude As Long = getlong * 1000000

        src = "http://www.google.com/ig/api?weather=,,," & inputlatitude & "," & inputlongitude

        Try
            downxml = webclient.DownloadString(src)
            Using writer As StreamWriter = New StreamWriter("weather.xml", False)
                writer.WriteLine(downxml)
            End Using

        Catch ex As System.Net.WebException
            MessageBox.Show("Internet connection unavailable. " & _
                    "Unable to acquire latest weather info.", "Error!", _
                    MessageBoxButton.OK, MessageBoxImage.Error)
            Exit Sub
        End Try
```

```
    Dim xml_index As Integer = 0
    Dim windcon() As String
    readweather = XmlReader.Create("weather.xml")
    While readweather.Read()
      xml_index = xml_index + 1

      If xml_index = 14 Then
        Labelweather.Content = readweather("data")
        weather_picture(readweather("data"))
      End If

      If xml_index = 16 Then
        LabelTemp.Content = readweather("data")
      End If

      If xml_index = 23 Then
        txttempl.Content = "L: " & Math.Round((readweather("data") - 32) * 5 / 9) & "°"
      End If

      If xml_index = 24 Then
        txttemph.Content = "H: " & Math.Round((readweather("data") - 32) * 5 / 9) & "°"
      End If

      If readweather.Name = "wind_condition" Then
        windcon = Split(readweather("data"), " ")
        txtwind1.Content = windcon(1)
        txtwind2.Content = Format(Convert.ToDouble(windcon(3) * 1.609344), ".0")
      End If

    End While


  End Sub

  Public Sub weather_picture(ByVal weathercon As String)

    Dim weatherpicname As String
    If weathercon = "Mostly Cloud" Then
      weatherpicname = ".png"
    ElseIf weathercon = "Partly Sunny" Then
      weatherpicname = "16.png"
    ElseIf weathercon = "Scattered Thunderstorm" Then
      weatherpicname = "11.png"
    ElseIf weathercon = "Showers" Then
      weatherpicname = "09.png"
    ElseIf weathercon = "Scattered Showers" Then
      weatherpicname = "09.png"
    ElseIf weathercon = "Rain And Snow" Then
      weatherpicname = "12.png"
    ElseIf weathercon = "Overcast" Then
      weatherpicname = "16.png"
    ElseIf weathercon = "Light Snow" Then
      weatherpicname = "13.png"
    ElseIf weathercon = "Freezing Drizzle" Then
      weatherpicname = "13.png"
    ElseIf weathercon = "Chance Of Rain" Then
      weatherpicname = "09.png"
    ElseIf weathercon = "Sunny" Then
      weatherpicname = "01.png"
    ElseIf weathercon = "Clear" Then
      weatherpicname = "01d.png"
    ElseIf weathercon = "Mostly Sunny" Then
```

```
        weatherpicname = "02d.png"
    ElseIf weathercon = "Partly Cloudy" Then
        weatherpicname = "03d.png"
    ElseIf weathercon = "Mostly Cloudy" Then
        weatherpicname = "04.png"
    ElseIf weathercon = "Chance Of Storm" Then
        weatherpicname = "11.png"
    ElseIf weathercon = "Rain" Then
        weatherpicname = "10.png"
    ElseIf weathercon = "Rain Showers" Then
        weatherpicname = "10.png"
    ElseIf weathercon = "Chance Of Snow" Then
        weatherpicname = "13.png"
    ElseIf weathercon = "Cloudy" Then
        weatherpicname = "04.png"
    ElseIf weathercon = "Mist" Then
        weatherpicname = "09.png"
    ElseIf weathercon = "Storm" Then
        weatherpicname = "11.png"
    ElseIf weathercon = "Thunderstorm" Then
        weatherpicname = "11.png"
    ElseIf weathercon = "Chance Of Tstorm" Then
        weatherpicname = "11.png"
    ElseIf weathercon = "Sleet" Then
        weatherpicname = "15.png"
    ElseIf weathercon = "Snow" Then
        weatherpicname = "13.png"
    ElseIf weathercon = "Icy" Then
        weatherpicname = "13.png"
    ElseIf weathercon = "Dust" Then
        weatherpicname = "15.png"
    ElseIf weathercon = "Fog" Then
        weatherpicname = "15.png"
    ElseIf weathercon = "Smoke" Then
        weatherpicname = "15.png"
    ElseIf weathercon = "Haze" Then
        weatherpicname = "15.png"
    ElseIf weathercon = "Flurries" Then
        weatherpicname = "15.png"
    ElseIf weathercon = "Light rain" Then
        weatherpicname = "09.png"
    ElseIf weathercon = "Snow Showers" Then
        weatherpicname = "14.png"
    ElseIf weathercon = "Ice/Snow" Then
        weatherpicname = "13.png"
    ElseIf weathercon = "Windy" Then
        weatherpicname = "04.png"
    ElseIf weathercon = "Scattered Snow Showers" Then
        weatherpicname = "14.png"
    Else
        weatherpicname = "no.png"
    End If

    Dim logo As New BitmapImage
    logo.BeginInit()
    logo.UriSource = New Uri(CurDir() & "\weather_icon\dotvoid\icons_60x50\" & weatherpicname)
    logo.EndInit()
    imageweather.Source = logo


End Sub
```

```
#End Region

#Region "view map"


    Public Sub viewmap(ByVal getlong As String, ByVal getlati As String, ByVal warn_latitude As String, ByVal
warn_longitude As String, ByVal mapzoomlvl As String)

        Dim iLongitude As String = getlong
        Dim iLatitude As String = getlati
        Dim currdir As String = CurDir() & "\output.html"
        Dim read() As String = System.IO.File.ReadAllLines("map.html")

        Using writer As StreamWriter = New StreamWriter("output.html", False)
          writer.WriteLine(read(0))
        End Using

        For value As Integer = 1 To 19
          Using writer As StreamWriter = New StreamWriter("output.html", True)
            writer.WriteLine(read(value))
          End Using
        Next

        ' map.setCenter(new GLatLng(3.1475, 101.69333), 18);
        Dim mapcenter As String
        mapcenter = "map.setCenter(new GLatLng(" & iLatitude & "," & iLongitude & ")," & mapzoomlvl & ");"
        Using writer As StreamWriter = New StreamWriter("output.html", True)
          writer.WriteLine(mapcenter)
        End Using

        For value As Integer = 21 To 32
          Using writer As StreamWriter = New StreamWriter("output.html", True)
            writer.WriteLine(read(value))
          End Using
        Next


        ' Current position in map
        Dim marker1 As String
        Dim marker1_1 As String
        Dim marker1_icon As String

        marker1_icon = "markerOptions = { icon:carIcon };"
        marker1 = "var marker = new GMarker(new GLatLng(" & iLatitude & "," & iLongitude & "),markerOptions);"
        marker1_1 = "map.addOverlay(marker);"
        Using writer As StreamWriter = New StreamWriter("output.html", True)
          writer.WriteLine(marker1_icon)
          writer.WriteLine(marker1)
          writer.WriteLine(marker1_1)
        End Using

        marker1_icon = "markerOptions = { icon:warnIcon };"
        marker1 = "var marker = new GMarker(new GLatLng(" & warn_latitude & "," & warn_longitude &
"),markerOptions);"
        marker1_1 = "map.addOverlay(marker);"
        Using writer As StreamWriter = New StreamWriter("output.html", True)
          writer.WriteLine(marker1_icon)
          writer.WriteLine(marker1)
          writer.WriteLine(marker1_1)
        End Using
```

```
        For value As Integer = 43 To 60
            Using writer As StreamWriter = New StreamWriter("output.html", True)
                writer.WriteLine(read(value))
            End Using
        Next
        WebBrowserGmap.Navigate(currdir)


    End Sub



#End Region


#Region "subroutine"


    Public Sub gps_signal_strength(ByVal hdopval As Decimal)
        Dim gpspic As String

        If hdopval >= 20 Then
            gpspic = "bar00.png"
        ElseIf hdopval >= 9 Then
            gpspic = "bar20.png"
        ElseIf hdopval >= 7 Then
            gpspic = "bar40.png"
        ElseIf hdopval >= 6 Then
            gpspic = "bar60.png"
        ElseIf hdopval >= 3 Then
            gpspic = "bar80.png"
        ElseIf hdopval > 0 Then
            gpspic = "bar100.png"
        Else
            gpspic = "bar00.png"
        End If

        Dim gps_signal_output As New BitmapImage
        gps_signal_output.BeginInit()
        gps_signal_output.UriSource = New Uri(CurDir() & "\Resources\" & gpspic)
        gps_signal_output.EndInit()
        image_gps_signal.Source = gps_signal_output

    End Sub

    Public Sub time_elapsed()

        seconddate = DateTime.Now
        Dim difftime As TimeSpan
        difftime = seconddate - firstdate
        labeltime2.Content = difftime.Hours & ":" & difftime.Minutes & ":" & difftime.Seconds

    End Sub


    Public Sub time_utc(ByVal utc_input As String, ByVal utc_date As String)

        Dim gpstime_array() As Char = utc_input.ToCharArray()
        Dim gpsdate_array() As Char = utc_date.ToCharArray()
        Dim utc_temp As String = "20" & gpsdate_array(4) & gpsdate_array(5) & "/" & gpsdate_array(2) &
gpsdate_array(3) & "/" & gpsdate_array(0) & gpsdate_array(1) & " " & gpstime_array(0) & gpstime_array(1) & ":" &
```

```
gpstime_array(2) & gpstime_array(3) & ":" & gpstime_array(4) & gpstime_array(5)
        nowsendtime = utc_temp
        Dim datetime_utc As DateTime = DateTime.Parse(utc_temp)
        datetime_utc = datetime_utc.AddHours(system_gmt)
        satellite_datetime = datetime_utc
        nowtime_hour.Content = datetime_utc.ToString("hh")
        nowtime_minutes.Content = datetime_utc.ToString("mm")
        nowtime_ampm.Content = datetime_utc.ToString("tt")
        nowday.Content = datetime_utc.ToString("dddd")
        nowdate.Content = datetime_utc.ToString("dd")
        now_month.Content = datetime_utc.ToString("MMMM")
        now_year.Content = datetime_utc.ToString("yyy")
    End Sub

#End Region

#Region "UDP broadcast"

    Dim GLOIP As IPAddress
    Dim GLOINTPORT As Integer
    Dim SocketNO As Integer = ConfigurationManager.AppSettings("SYSTEMSOCKET")
    Dim receivingUdpClient As UdpClient = New UdpClient(New IPEndPoint(IPAddress.Any, SocketNO))

    Private Sub receivemessage()
      receive()
    End Sub


    Public Sub receive()

      If (receivingUdpClient.Available > 0) Then

        Try
          Dim RemoteIpEndPoint As New IPEndPoint(IPAddress.Any, 0)
          Dim receiveBytes As [Byte]()
          receiveBytes = receivingUdpClient.Receive(RemoteIpEndPoint)
          Dim returnData As String = Encoding.ASCII.GetString(receiveBytes)

          txtmessagereceive.Text = returnData

          My.Computer.Audio.Play("alert.wav", AudioPlayMode.Background)
          udp_dataprocess(returnData)
        Catch

        End Try

      End If

    End Sub

    Private Sub btnsend_Click(ByVal sender As System.Object, ByVal e As System.Windows.RoutedEventArgs)
Handles btnsend.Click
        Try
          Dim message_output As String
          Dim nicuse As String = combonetwork.Text
          Dim localend As IPEndPoint = New IPEndPoint(IPAddress.Parse(nicuse), 0)
          Dim myudpClient As New UdpClient(localend)

          message_output = combo_sending_type.Text & "," & globallatitude & "," & globallongitude & "," &
nowsendtime & ","
          GLOINTPORT = ConfigurationManager.AppSettings("SYSTEMSOCKET")
          myudpClient.Connect(IPAddress.Broadcast, GLOINTPORT)
          Dim bytCommand As [Byte]() = Encoding.ASCII.GetBytes(message_output)
          Dim pRet As Integer = myudpClient.Send(bytCommand, bytCommand.Length)
```

```
            myudpClient.Close()
        Catch
        End Try
    End Sub


    Public Sub udp_dataprocess(ByVal input As String)
        Dim udp_message_array() As String = Split(input, ",")
        Dim udp_datetime_receive As DateTime = DateTime.Parse(udp_message_array(3))
        receive_warninglatitude = udp_message_array(1)
        receive_warninglongitude = udp_message_array(2)

        udp_datetime_receive = udp_datetime_receive.AddHours(system_gmt)
        txt_display_warning.Content = "Warning"

        If InStr(udp_message_array(0), "Fog") Then
            txt_display_condition.Content = "Fog Ahead"
            txt_display_advice.Content = "Reduced Visibility"
            txt_display_date.Content = udp_datetime_receive.ToString("hh mm tt")
            txt_display_time.Content = udp_datetime_receive.ToString("dd/MM/yyyy")
        End If


        If InStr(udp_message_array(0), "Road Slippery") Then
            txt_display_condition.Content = "Wet Road Ahead"
            txt_display_advice.Content = "Road Slippery"
            txt_display_date.Content = udp_datetime_receive.ToString("hh mm tt")
            txt_display_time.Content = udp_datetime_receive.ToString("dd/MM/yyyy")
        End If

        'txt_display_distance.Content = GetDistanceBetweenPoints(globallatitude, globallongitude, 4.38594,
100.96556)receive_warninglatitude, receive_warninglongitude
        txt_display_distance.Content = Math.Round(distance(globallatitude, globallongitude, receive_warninglatitude,
receive_warninglongitude) * 1000).ToString & " M"

        viewmap(globallongitude, globallatitude, receive_warninglatitude, receive_warninglongitude,
ConfigurationManager.AppSettings("gmap_zoom_level"))
        Array.Clear(udp_message_array, 0, udp_message_array.Length - 1)
    End Sub




#End Region



    Private Sub btn_display_clear_Click(ByVal sender As System.Object, ByVal e As
System.Windows.RoutedEventArgs) Handles btn_display_clear.Click
        txt_display_advice.Content = ""
        txt_display_condition.Content = ""
        txt_display_date.Content = ""
        txt_display_time.Content = ""
        txt_display_warning.Content = ""
        txt_display_distance.Content = ""
    End Sub

End Class
End Namespace
```

APPENDIX B

APPLICATION GRAPHICAL USER INTERFACE CODE

MainWindow.xaml

(MICROSOFT VISUAL STUDIO 2010)

```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d"
x:Class="Serial_Communication_WPF.MainWindow"
    Title="LaporPO" Height="578" Width="1094" Icon="/VANET_application;component/Globe%201.ico"
ResizeMode="NoResize" WindowStartupLocation="CenterScreen" DataContext="{Binding}">
  <Window.Resources>
    <XmlDataProvider x:Key="ComPorts" Source="CommsData.xml" XPath="/Comms/Ports" />
    <XmlDataProvider x:Key="ComSpeed" Source="CommsData.xml" XPath="/Comms/Baud" />
  </Window.Resources>
  <Canvas Margin="341.5,0,2,0" Width="740">
    <GroupBox Header="Comm Port" Width="183" Height="118" Canvas.Left="-338" Canvas.Top="20">
                    <Grid Margin="2,0.04,2,2">
                        <ComboBox x:Name="combobaudrate" HorizontalAlignment="Left"
Text="Select..." ItemsSource="{Binding Source={StaticResource ComSpeed}}" Width="70" Margin="8,32,0,0"
Height="20" VerticalAlignment="Top" />
                        <ComboBox x:Name="combocomport" MinWidth="130" ItemsSource="{Binding
Source={StaticResource ComPorts}}" Height="20" HorizontalAlignment="Left" Width="70" Margin="8,8,0,0"
VerticalAlignment="Top" />
                        <Button Content="Connect" Click="Connect_Comms" x:Name="Connect_btn"
Margin="8,56,0,0" Height="24" VerticalAlignment="Top" HorizontalAlignment="Left"/>
                    </Grid>
                </GroupBox>
                <GroupBox Header="Coordinate" Height="87" Canvas.Left="-338" Canvas.Top="142" Width="183">
                    <Grid Margin="2,3.04,2,2">
                        <Label Content="latitude" Height="33" HorizontalAlignment="Right"
x:Name="txtgpslatitude2" VerticalAlignment="Top" Width="151" Margin="8,0,0,0" FontSize="18.667"
HorizontalContentAlignment="Right" />
                        <Label Content="longitude" Height="33" HorizontalAlignment="Right"
x:Name="txtgpslongitude2" VerticalAlignment="Top" Width="151" Margin="8,26,0,0" FontSize="18.667"
HorizontalContentAlignment="Right" />
                    </Grid>
                </GroupBox>
                <GroupBox Header="Temperature" Height="90" Canvas.Left="-338" Canvas.Top="233"
Width="183">
                    <Grid>
                        <Label FontFamily="Segoe UI" FontSize="56" FontWeight="Normal"
Foreground="Black" HorizontalContentAlignment="Left" x:Name="LabelTemp" VerticalContentAlignment="Top"
Margin="2,-11,84,0" Height="81" VerticalAlignment="Top" Content="00" />
                        <Label Content="L: 00°" x:Name="txttempl" Width="54" FontSize="16"
HorizontalAlignment="Right" Margin="0,29.04,-2,8" />
                        <Label FontFamily="Segoe UI" FontSize="56" FontWeight="Normal"
Foreground="Black" HorizontalContentAlignment="Left" x:Name="LabelTemp2" VerticalContentAlignment="Top"
Margin="61,-12,25,0" Height="81" VerticalAlignment="Top" Content="°C" />
        <Label Content="H: 00°" Height="33" Margin="116,0,-3,0" Name="txttemph" VerticalAlignment="Top"
FontSize="16" />
      </Grid>
                </GroupBox>
                <GroupBox Header="Date" Height="118" Canvas.Top="20" Width="184" Canvas.Left="-151">
                    <Grid>
        <Label Content="00" Margin="8,-7.807,99,34" Name="nowdate" FontSize="53.333"
HorizontalContentAlignment="Center" VerticalContentAlignment="Bottom" />
        <Label Content="day now" Height="58" HorizontalAlignment="Left" Margin="10,51.04,0,0" Name="nowday"
VerticalAlignment="Top" Width="147" FontSize="26.667" />
```

```
        <Label Content="Month" Height="33" HorizontalAlignment="Left" Margin="73,8,0,0" x:Name="now_month"
VerticalAlignment="Top" Width="91" FontSize="18" />
        <Label Content="Year" HorizontalAlignment="Left" Margin="73,31.04,0,34" x:Name="now_year"
Width="91" RenderTransformOrigin="1.429,0.386" FontSize="18" />
    </Grid>
            </GroupBox>
            <GroupBox Header="Velocity" Height="87" Canvas.Left="37" Canvas.Top="142" Width="183">
                <Grid Margin="2,3.04,2,2">
                    <Label Content="0" x:Name="txtvelocity" HorizontalAlignment="Left"
Width="99" Margin="-2,-22,0,0" FontSize="64" HorizontalContentAlignment="Right" VerticalAlignment="Top"
Height="84" />
                    <Label Content="km/h" x:Name="txtvelocity2" HorizontalAlignment="Right"
Width="73" FontSize="28" VerticalContentAlignment="Center" Margin="0,0,1,1" VerticalAlignment="Bottom" />
                </Grid>
            </GroupBox>
            <GroupBox Header="Weather" Height="90" Canvas.Top="233" Width="183" Canvas.Left="-150">
                <Grid>
                    <Image DataContext="{Binding}" Height="50" HorizontalAlignment="Left"
x:Name="imageweather" Stretch="Fill" VerticalAlignment="Top" Width="60"
d:LayoutOverrides="HorizontalAlignment" />
                    <Label FontFamily="Segoe UI" FontSize="20" Foreground="Black"
HorizontalContentAlignment="Right" x:Name="Labelweather" VerticalContentAlignment="Top" Margin="30,26,1,-5"
Content="weather" />
                </Grid>
            </GroupBox>
            <GroupBox Header="Satellites " Height="118" Canvas.Left="37" Canvas.Top="20" Width="184">
                <Grid>
                    <Label Content="00" x:Name="txtsatellite" HorizontalAlignment="Left"
Width="89" FontSize="64" Margin="0,-22,0,30" />
        <Image Height="84" HorizontalAlignment="Left" Margin="78,6,0,0" Name="image_gps_signal"
Stretch="Fill" VerticalAlignment="Top" Width="91" />
        <Label Height="38" Margin="6,57,100,0" Name="txt_accuracy" VerticalAlignment="Top" FontSize="18" />
        <Label FontSize="18" Height="38" HorizontalAlignment="Right" Margin="0,57,84,0" Name="txthdop"
VerticalAlignment="Top" Width="40" Visibility="Hidden" />
    </Grid>
            </GroupBox>
    <Label Content="time" x:Name="labeltime2" Width="42" Height="26" Canvas.Left="499.5" Canvas.Top="0"
HorizontalContentAlignment="Left" />
    <GroupBox Header="Time" Height="87" Canvas.Top="142" Width="183" Canvas.Left="-151">
        <Grid Margin="2,3.04,2,2">
            <Label Content="00" HorizontalAlignment="Left" x:Name="nowtime_hour" Width="77" Margin="-
4,-11,0,6" FontSize="48" d:LayoutOverrides="HorizontalAlignment" />
            <Label Content="00" HorizontalAlignment="Left" x:Name="nowtime_minutes" Width="72"
Margin="56,-11,0,6" FontSize="48" />
            <Label Content="PM" HorizontalAlignment="Left" x:Name="nowtime_ampm" Width="77"
Margin="108,-1,0,-4" FontSize="37.333" />
            <Label Content=":" Margin="49.006,0,0,-7" VerticalAlignment="Bottom" FontSize="34.667"
Height="70" HorizontalAlignment="Left" Width="24.994"/>
        </Grid>
    </GroupBox>
    <Label Canvas.Left="422.5" Canvas.Top="0" Content="uptime" Height="26" HorizontalContentAlignment="Right"
Name="Label1" Width="79" />
    <GroupBox Header="Map" Height="374" Canvas.Left="225" Canvas.Top="20" Width="440">
        <Grid Width="420">
            <WebBrowser x:Name="WebBrowserGmap" Margin="0,0,16,8"/>
        </Grid>
    </GroupBox>
    <GroupBox Header="Broadcast Warning" Height="90" Canvas.Left="38" Canvas.Top="233" Width="183">
        <Grid Margin="2,3.04,2,2">
            <ComboBox x:Name="combo_sending_type" Height="20.04" Margin="83,0.96,6,0"
VerticalAlignment="Top">
                <ComboBoxItem Content="Fog"/>
                <ComboBoxItem Content="Road Slippery"/>
            </ComboBox>
```

```
                <Label Content="Warning Type" Height="30" HorizontalAlignment="Left" Margin="-1.016,-3.5,0,0"
x:Name="Label1_Copy" VerticalAlignment="Top" Width="86" />
                <ComboBox x:Name="combonetwork" Margin="83,26.5,6,12.46"/>
                <Button Content="Send" HorizontalAlignment="Left" Height="20.04" VerticalAlignment="Top"
Width="46" x:Name="btnsend" Margin="8,26.5,0,0" RenderTransformOrigin="0.832,1.447" />
        </Grid>
    </GroupBox>
    <GroupBox Canvas.Left="-340" Canvas.Top="325" Header="Windspeed" Height="103" Width="183">
      <Grid>
        <Label Content="0.0" FontFamily="Segoe UI" FontSize="40" FontWeight="Normal" Foreground="Black"
HorizontalContentAlignment="Right" x:Name="txtwind2" VerticalContentAlignment="Top" Margin="1,21.5,82,-2.46"
/>
        <Label Content="km/h" FontFamily="Segoe UI" FontSize="28" FontWeight="Normal" Foreground="Black"
HorizontalContentAlignment="Left" x:Name="Label2" VerticalContentAlignment="Top" Margin="82,34.5,14.21,-4.46"
d:LayoutOverrides="Width" />
        <Label Content="NW" FontFamily="Segoe UI" FontSize="40" FontWeight="Normal" Foreground="Black"
HorizontalContentAlignment="Center" x:Name="txtwind1" VerticalContentAlignment="Top" Margin="45,-
11.5,48.713,26.54" d:LayoutOverrides="Width" /></Grid>
    </GroupBox>
    <GroupBox Height="144" Canvas.Left="225" Canvas.Top="398" Width="440" Header="Warning ">
        <Grid Margin="3.25,0,2,-0.96">
                <Label Content="
" Margin="134,-2.672,129.756,0" VerticalAlignment="Top" HorizontalContentAlignment="Center" FontSize="24"
Name="txt_display_warning" />
                <Label Content="
" Margin="8,22.748,8,57.316" d:LayoutOverrides="Height" HorizontalContentAlignment="Center" FontSize="24"
Name="txt_display_condition" />
                <Label Height="25.356" Width="81" HorizontalAlignment="Left" Margin="6,0,0,8"
VerticalAlignment="Bottom" Name="txt_display_time" />
                <Label Content="
" Margin="8,49.748,8,30.316" d:LayoutOverrides="Height" HorizontalContentAlignment="Center" FontSize="24"
Name="txt_display_advice" />
        <Label Height="25.356" HorizontalAlignment="Left" Margin="71,0,0,8" Name="txt_display_date"
VerticalAlignment="Bottom" Width="81" />
        <Button Content="clear" HorizontalAlignment="Right" Height="22.316" Margin="0,0,8,8"
VerticalAlignment="Bottom" Width="57.75" Name="btn_display_clear" />
        <Label Height="25.356" HorizontalAlignment="Left" Margin="265,0,0,8" Name="txt_display_distance"
VerticalAlignment="Bottom" Width="81" />
        <Label Content="Distance:" Height="27" HorizontalAlignment="Left" Margin="181,89,0,0"
Name="label_distance" VerticalAlignment="Top" Width="84" HorizontalContentAlignment="Right" />
      </Grid>
    </GroupBox>
    <GroupBox Canvas.Left="-150" Canvas.Top="327" Header="Message Receive" Height="103" Width="183">
        <Grid>
                <TextBox x:Name="txtmessagereceive" TextWrapping="Wrap" Text="TextBox" Margin="8"/>
        </Grid>
    </GroupBox>
  </Canvas>
</Window>
```