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

By

OMAR FOO BIN ABDUL RAHMAN FOO

FINAL PROJECT REPORT

Submitted to the Department of Electrical & Electronic Engineering
in Partial Fulfillment of the Requirements

for the Degree

Bachelor of Engineering (Hons)

(Electrical & Electronic Engineering)

Universiti Teknologi PETRONAS
Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

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

by

Omar Foo Bin Abdul Rahman Foo

A project dissertation submitted to the
Department of Electrical & Electronic Engineering
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

Approved:

_____

Assoc. Prof. Dr. M. Naufal B. M. Saad
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2012

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.

_____

Omar Foo Bin Abdul Rahman Foo

ABSTRACT

The number of accidents in Malaysia that increases annually is a severe problem that needs to be addressed aggressively [1]. Of the many factors that cause accidents, the reduced concentration of driver during a driving session is one of the main factors that add up to the number of road accident and this is more so during bad weather such as during rain fall, fog or snow. There is an urgency to develop something to help and increase the safety of the driver by warning the driver of the driving situation ahead. This project addresses this urgency by developing a safety application that is used to alert the driver of road/environment abnormality and spread the information to other drivers nearby. The vehicle to vehicle communication is done by implementing a platform for Vehicle Ad Hoc Networks (VANETs). This networks works by automatically forming networks between vehicles in real time. Currently the communication is done using the standard WiFi network. This project is strictly being developed on the application layer using JAVA. For the next stage of this project, the application can be linked to vehicle sensors and external sensors where the data obtained can be processed and broadcasted to other vehicles.

ACKNOWLEDGEMENTS

I would like to thanks to my supervisors, Assoc. Prof. Dr. Mohamad Naufal 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 who are guiding me in the field of VANETs. The information that you gave me is really helpful for me in understanding the project.

I would like to give thanks to my brother, Ilyas Foo Bin Abdul Rahman Foo wich give guidance in the field of JAVA programming which tremendously helped this project to achieve completion.

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 Contents

I. LIST OF FIGURE

II. LIST OF TABLE

CHAPTER 1 INTRODUCTION

1. PROJECT BACKGROUND

Malaysian roads are becoming busier every day. Everyone can feel the tension on our road which has increased compared to a decade ago. This can be seen in the statistical data shown form Figure 1.

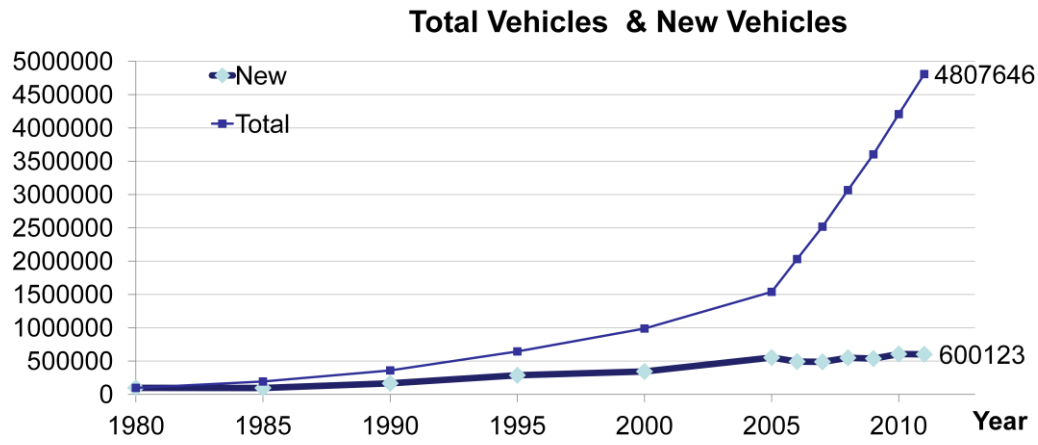

Figure 1: Total Vehicles vs Years - *Malaysia Automotive Info. (2011). Summary of Sales & Production Data*

As of 2011, the number of new car entering our road is 600,123 and a total of 4,807,646 cars is currently on our road [1]. With such numbers, the risk of driving increases as the road becomes more congested and vehicle becoming prone to accidents.

Advancement in safety technology enables vehicles producers to implements the safety technology such as Automatic Braking System and Adaptive Light Control. But such features are isolated in the individual vehicle alone [6]. It is important to have a method to enable a type of cooperation between vehicles. These mechanisms would enable the vehicles to exchange messages containing critical safety information to each other in which would improve the coordination between the local road users. To realize the cooperation, a communication method is needed. A new type of networking is being developed in the recent years which would support the communication between two or more vehicles in real time high speed environment. The network is called Vehicular Ad Hoc Networks (VANETs).

VANETs development gives a new insight to develop a range of application that includes the development of dynamic safety application. This application could in effect improve the driving experience and safety of the occupant.

This project focuses on the development of the said safety application in which helps the driver to become more alert of what is ahead of them from seconds to minutes. This is done by notifying the driver and those around him of the road condition. The applications will also be able to monitor the surrounding of the vehicle.

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*

This Final Year Project is done strictly on the first year domain. The basic function for the application is defined. Test scenarios will be generated and taken into consideration. The graphical user interface with features such as weather forecasting and GPS maps is also developed. By the end of this, users can view the application and interacted with using the GUI.

2. PROBLEM STATEMENT

Figure 2: An accident

Vehicular accident is a serious problem that plagues various countries in the world. In Malaysia alone, the statistic summary shows us the accidents that occur reach a disturbing level.

As in Appendix III, by the year 2010, the number of registered vehicles reaches 605,156 units. This number is expected to rise even further. This overcrowding of vehicles will prove to be a challenge to the effort of controlling accidents. With the accidents that are recorded in the statistics in Appendix I and II over the recent decade things are looking grim.

The number of accidents that occur annually in Malaysia is more than 300,000. These accidents accounted for RM 9 Billion in property damages, 391,751 non-fatal injuries and 22,260 with injuries. In addition, from a newspaper report, the average death caused by road accident is 17 persons each day [2]. Across the world, World Health Organization (WHO) reported that nearly 1.3 million people die as a result of a road traffic collision annually which averages to 3000 death each day [3].

It shows there are much improvement could be implemented to increase the road safety conditions. On general analysis, factors that contribute to the accidents are due to mechanical issues, road design, road condition and poor driver performance. With such, the implementation of comprehensive traffic management could further reduce the number of accidents towards the foreseeable future.

One of the ways to achieve this traffic management is to develop a safety application for road users. This application tackles two types of event, road based condition and also weather condition. The Figure 3, 4 & 5 below shows some of the most common weather condition.

Figure 1: Rain

Figure 2: Heavy Fog

Figure 3: Snow

This application must be versatile such it must be able to know the location of the vehicle and also perceive the environment around it. To do that, the application will be interfaced with GPS, sensors and internet. To enable communication, the application uses VANETs.

3. OBJECTIVE

Develop an active safety application which makes it possible to warn drivers in vehicles about the condition of the road and/or their environment.

- *Linux based warning system application of VANET using JAVA.*
- *To sense the conditions of roads and environment using sensors.*
- *To be able to transmit and receive data and display it to the driver.*

4. SCOPE OF STUDY

These are the scope of study for the Final Year Project.

- *Study of JAVA programming on Linux platform.*
- *Study of interfacing GPS with test machine.*
- *Study of serial communication to read from GPS.*
- *Study of GPS method of communication.*
- *Study UDP implementation using JAVA*

The project is developed strictly in the application layer and the communication is done using standard WiFi due to the unavailability of VANETs devices. It can be used as a test platform for VANETs devices. The communication layer is not delved into and treated as transparent layer.

5. RELEVANCY OF PROJECT

The implementation of the warning system in vehicles with the capability to communicate and allow the drivers of vehicles to coordinate accordingly has a very promising future to facilitate higher safety standard for road users. It gives the road users early warning and allow them to exercise extra caution in during their driving session on the road.

6. FEASIBLITY OF PROJECT

This project requires testing and retesting as new features are added during development. Debugging will also consume much of the development time. If the timeline is followed closely, this project can achieve its intended objective on time.

CHAPTER 2 LITERATURE REVIEW

1. Intelligent Transportation System (ITS)

ITS is a system, in driver point of view, that is able to perceive the situation of driving and give assistance that enhance the driving experience. The system is supported by the on-board navigation system [4].

2. VANETs

To accomplish the ITS, there must be communication between fixed immobile communicator to the vehicles (V2R) and also between the vehicles (V2V) [5]. The fixed communicator is called road side unit (RSU). On the vehicle, the device is called on board unit (OBU). RSU devices can be installed onto any other construct that usually on the roadside such as lamp post and traffic light. OBU on the other hand is placed into the vehicles and functions while on the move [5]. The figure below explains this.

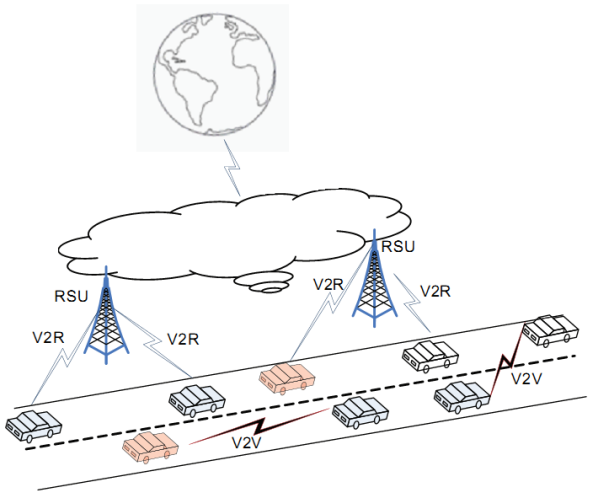

Figure 6: Data transmission flow [5].

The current wireless networks that are widely used would not be able to handle the heavy communication traffic generated by the high speed vehicles. Thus VANETs is being developed especially to handle the networking.

There are currently several application that uses VANETs.

- *Electronic Toll Collection*
- *Safety Warning Application*
- *Internet Access Node*
- *Group Communication Node*
- *Roadside Services Finder*

This final year project focuses on the development of safety warning application that is proposed for VANETs.

3. Safety Application for Vehicle

The application layer provides a method of communication with transparency of the layers below for the programs on the node, i.e. vehicles and roadside units. When VANETs has gained enough popularity and wide implemented, these program will be produced by manufacturers en mass. For this project, the program developed will monitor sensors and broadcasts safety messages in an event of emergency. It is also capable of knowing its own location, its nearest neighbor's location, road and weather condition ahead [7]. With that wealth of information, driver can act and coordinate themselves accordingly.

4. Global Positioning System (GPS)

The GPS is a satellite based navigation system. A user with a GPS receiver is able to know its location with an accuracy of several meters at any point on Earth. To be able to determine the location of an object, the GPS receiver must be able to "see" at least 3 GPS satellites [8]. The coordinate of an object is determined using triangulation method. Figure 7 shows how GPS receiver works.

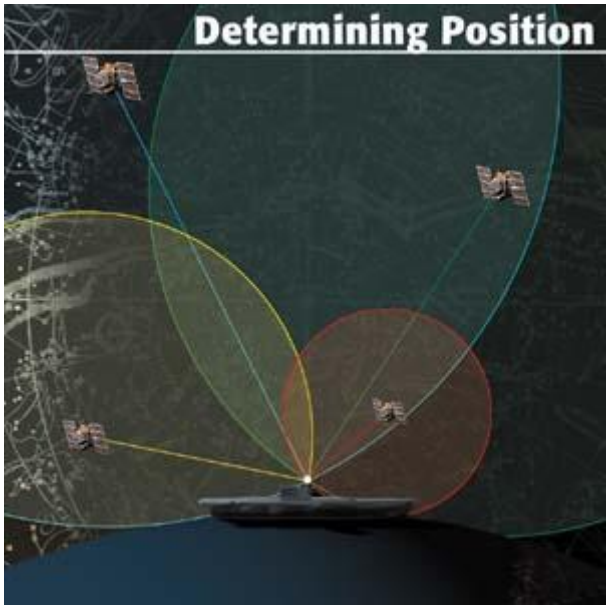

Figure 4: How GPS determine its position

With 3 satellites visible to the GPS receiver, the GPS receiver is able to calculate its position in 2-dimension, longitude and latitude. With 4 or more satellites visible to the GPS receiver, it will be able to calculate its location in 3-dimension, longitude, latitude, and altitude. The satellites also provide wealth of other navigation information which the GPS receiver could use to calculate properties such as speed, bearing, track and others [9].

The GPS receiver streams the data to reader in National Marine Electronics Association (NMEA) format. NMEA data uses ASCII characters. The data is sent serially, character by character across the RS-232 port. Table 1 lists the common GPS NMEA messages being used for navigation.

Table 1: Information provided by the GPS receiver

$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*

5. VANET in LINUX

On similar application of VANET, Cooperation Vehicle Infrastructure Systems (CVIS), vehicles networking for ITS was implemented on LINUX based platform.

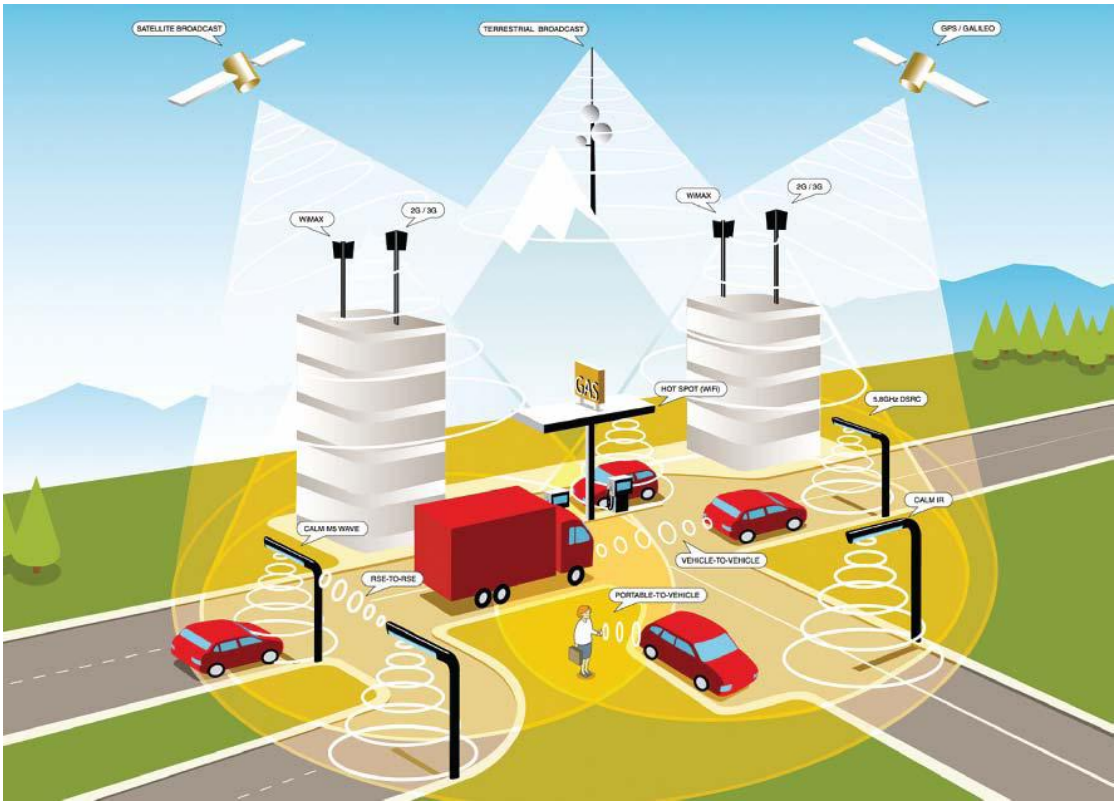

Figure 8: CVIS communication overview [12].

CVIS software architecture developed based on LINUX platform of Ubuntu. This software is easily and freely available [12].

The core of CVIS is developed on JAVA runtime environment [12]. Thus in lieu of the objective of this project, LINUX based road warning application of VANET is possible and within reach of the current technology.

CHAPTER 3 METHODOLOGY

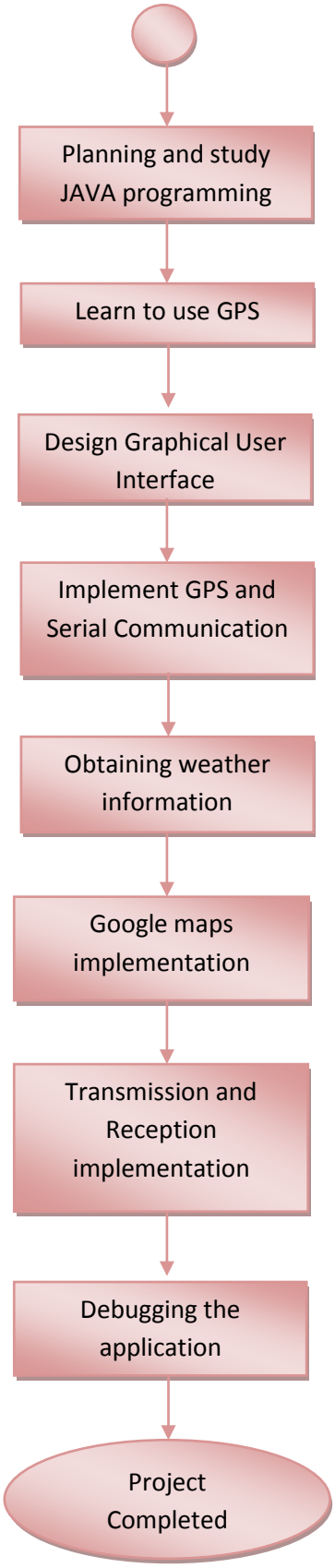

Figure 9: Prospected Methodology of the Project

1. **Planning and study JAVA programming**

   The project is planned as which device/code will be coded first. JAVA language is studied heavily as the project involves interfacing multiple device, and different software programming language.

2. **Learn to use GPS**

   The GPS receiver is not a plug and play type of device. The test machine needed to be configured so that it is able to accept the GPS data stream in such a way it is readable by users. The JAVA platform also needed to be configured to support serial communication in which the official JAVA does not support.

Figure 5: Byonics GPS receiver

The GPS itself is powered with 5V supply from the test machine. It is configured to use 4800 Baud rate. To read from the receiver, a machine needs to have a serial port and a serial terminal program which is configured to read ASCII character at 4800 baud.

This receiver sends streams of character forming 4 sentences starting with $GPGSA $GPRMC $GPGGA $GPGSV

The GPS receiver is the core part of this project. It provides the application with essential data such as coordinates, velocity and satellite time.

3. Design graphical user interface

In this phase, the basic GUI for the safety application is developed. It is developed using NetBean with JAVA application language. It is made to run on JAVA JDK 7.0 platform. Any machine have this platform is capable to run the program. It contains simple function of time and date display. Several sections are left empty which will be used when the right code is implemented.

4. Implement GPS and serial communication

This part is actual implementation of the GPS into the application. This is where the code to read the GPS receiver in the application is programmed. The data obtained is used for maps and several other functions. The sentences will be used in this application are $GPGSA, $GPRMC, $GPGGA

4.1. $GPGSA

It shows the number of current active satellites. The full details of this sentence are listed in Table 2.

Table 2: $GPGSA sentence

*$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)*

4.2. $GPRMC

This sentence gives the GPS receiver position and some other important values. It contains the most basic information. Thus it is also called the Recommended Minimum. Table 3 shows the details of this sentence.

Table 3: $GPRMC sentence

*$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*

4.3. $GPGGA

The sentence being used most for this application is this sentence. It provides data for basic GPS application. Table 4 shows the details of this sentence.

Table 4: $GPGGA sentence

*$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*

5. Obtaining weather information

The weather is obtained using Google Weather API. This API gives weather information obtained from Google database. The information is in XML format. Thus an XML reader is coded into the application to enable the use of the information for the application.

6. Google Map implementation

The map is implemented using Google Map API. The Google Map is displayed as a browser using HTML format. A code is implemented on the application so that a section of the application acts as a web browser. This browser is then used to read the HTML codes of the map and displays accordingly. By using Google Map API the map is customizable.

7. Transmission and reception implementation

To transmit the message, the application uses GPS receiver, internet connection, and sensor and network adapter. GPS receiver gives the coordinate of the vehicle, internet connection gives the data on weathers and a simulated sensor is used to trigger the broadcasts. A message will be sent via network adapter.

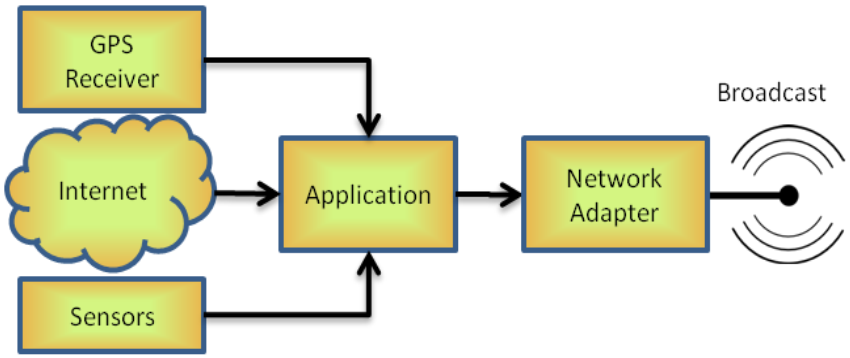

Figure 11: Block diagram transmit message

On the receiving vehicle, the network adapter listens to the receiver for any broadcast. The GPS receiver provides the self coordinates and compares it to the received warning messages. This coordinates will then be displayed in the map and a warning display will be issued on the application.

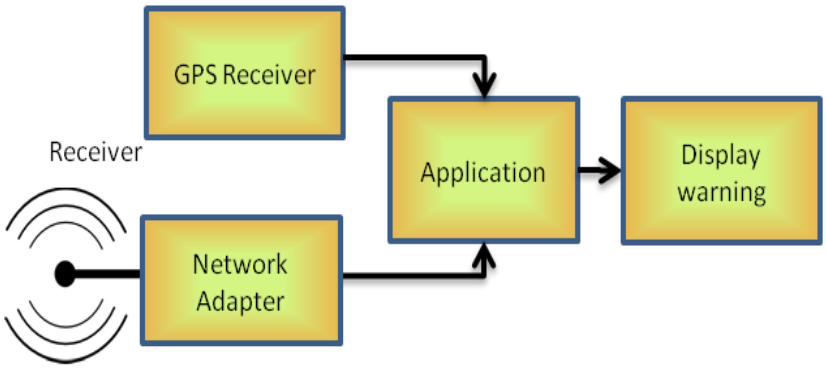

Figure 12: Block diagram receive message

8. Gantt Chart

	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14
Title selection / proposal														
Find resources														
Study WAVE architecture														
Extended proposal														
Proposal defense														
Training/Equipment draft														
Submission Interim Draft Report														
Submission Interim Report														

Figure 13: Gantt Chart Final Year Project 1

	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14
Interface GPS with application														
Test obtained GPS data														
Develop user interface														
Interface map with application														
Interface with network														
Minor improvement														
Debugging of application														

Figure 14: Gantt Chart Final Year Project 2

CHAPTER 4 FINDINGS

The programming of GUI of the application is completed. The JAVA codes are appended in Appendix IV.

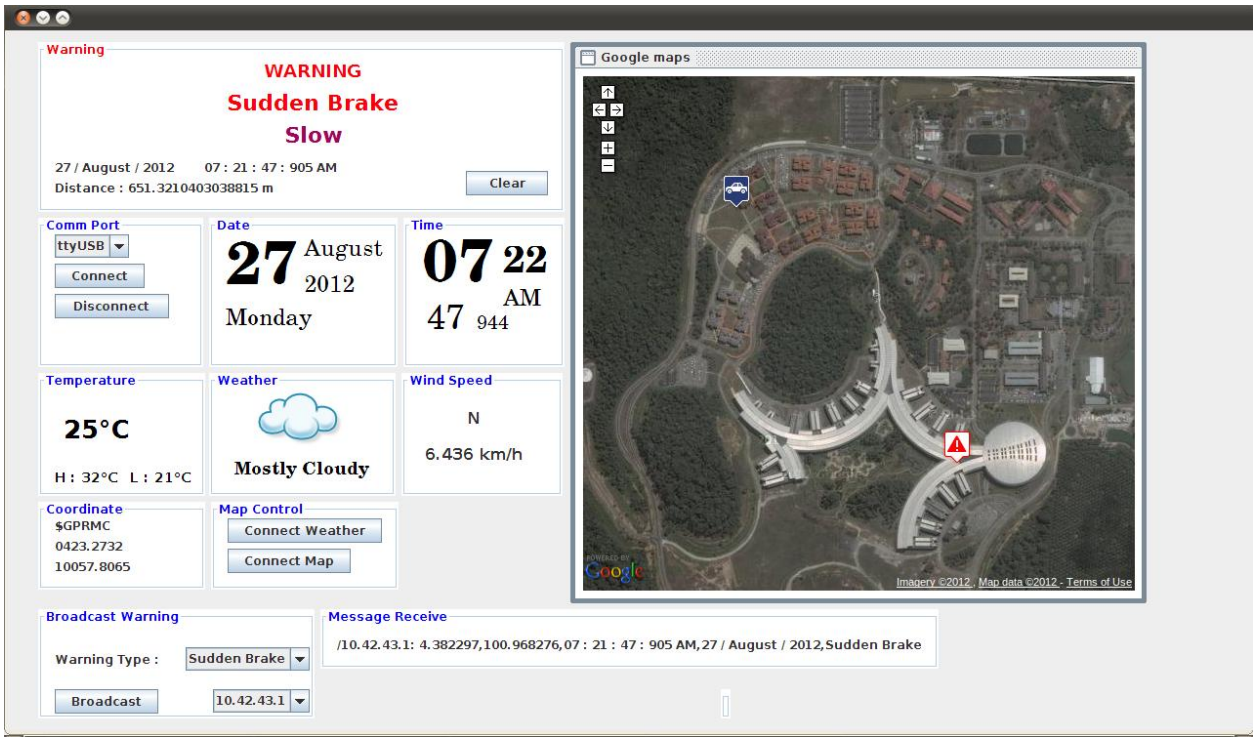

Figure 15: Completed GUI Sections

1. DATE & TIME SECTION

The GUI has internal timing which is set to run at every 10ms. When this time elapsed, the GUI fetches the current time and date from the computer it is running on. The codes are separated into 3 sections, the date fetch code, the time fetch code and the timing code. To simplify,

- *At each 10 ms*
- *Fetch current time and current date*
- *Display on GUI*

Figure 16: Date & Time Section

2. TEMPERATURE & WEATHER SECTION

This section works by implementing Google Weather API. The GUI retrieves information of the weather at certain coordinates from Google Weather in XML format. The XML contains multitudes of information including the weather forecasts up to 3 days ahead. Currently the program implements only the current weather information in relative to the location of the car. Then the GUI processes the information and displays the selected data which is relevant to the driver. To simplify,

- *Get weather information in XML*
- *Saves in program*
- *Select current weather condition and temperature*
- *Get appropriate icon*
- *Display on GUI*

Figure 17: Temperature & Weather Section

3. MAP SECTION

The map is produced by creating a browser environment in the application. The application then read the html codes of the map and generates the map. This code contains the coordinates generated by the GPS device. Several mathematical conversions have to be calculated before the coordinates is fed to the map.

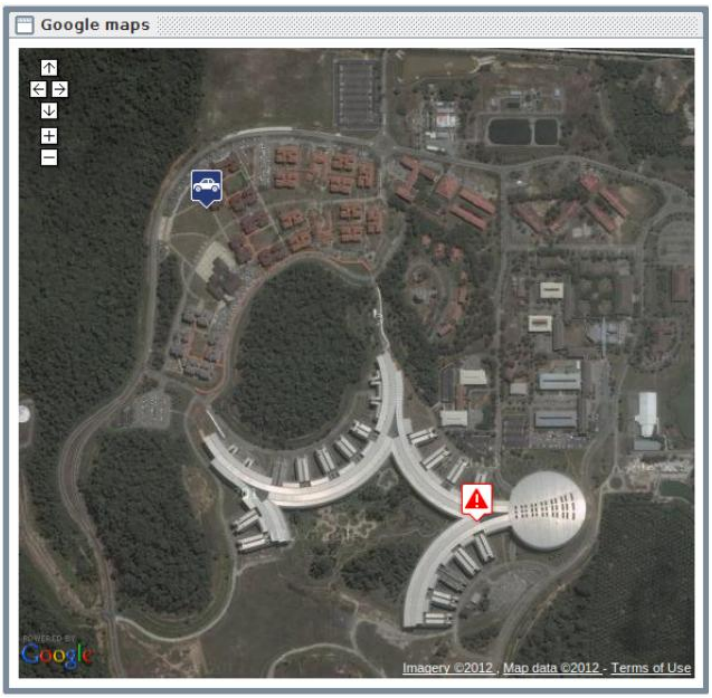

Figure 18: Map Section

4. GPS SECTION

The GPS device being used is Byonics GPS 2. This device sends request to satellites at the interval of 1 second. The reply send back to the device is a stream of NMEA formatted ASCII string of characters. This character is sent to the computer using serial communication port. The application then separates the streams of information into several variables each with different usage in the program. Out of many of the data received, only a few is used in the application. The list may expand.

- *Device Coordinate*

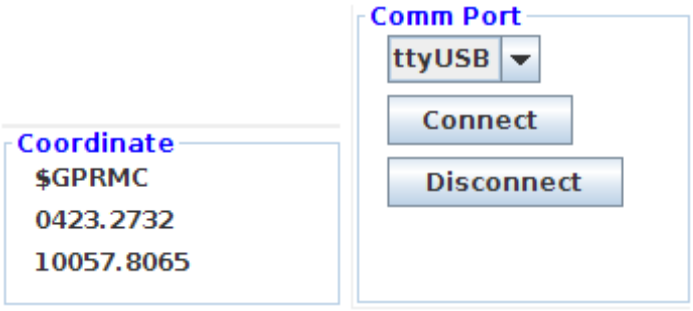

Figure 19: GPS Coordinate & Control

```
Input Stream... gnu.io.RXTXPort$SerialInputStream@461984
............
5$GPGGA,034132.000,0242.2444,N,10154.3678,E,2,08,1.3,94.6,M,-7.8,M,1.8,0000*55
Latitude = null
Longitude = null
$GPGSA,A,3,03,06,32,23,14,16,30,31,,,,,2.3,1.3,1.9*3F
Latitude = null
Longitude = null
$GPGSV,3,1,12,06,62,123,40,03,61,164,34,16,55,342,30,32,53,268,35*72
Latitude = null
Longitude = null
$GPGSV,3,2,12,30,33,009,26,19,30,188,25,20,29,288,22,14,27,118,24*7D
Latitude = null
Longitude = null
$GPGSV,3,3,12,31,15,044,26,23,11,331,23,11,10,215,26,22,08,151,*77
Latitude = null
Longitude = null
$GPRMC,034132.000,A,0242.2444,N,10154.3678,E,0.00,29.98,090712,,,D*52
Latitude = null
Longitude = null
$GPGGA,034133.000,0242.2444,N,10154.3678,E,2,08,1.3,94.6,M,-7.8,M,2.8,0000*57
Latitude = 0242.2444
Longitude = 10154.3678
$GPGSA,A,3,03,06,32,23,14,16,30,31,,,,,2.3,1.3,1.9*3F
Latitude = null
Longitude = null
$GPRMC,034133.000,A,0242.2444,N,10154.3678,E,0.00,29.98,090712,,,D*53
Latitude = null
Longitude = null
$GPGGA,034134.000,0242.2444,N,10154.3678,E,2,08,1.3,94.6,M,-7.8,M,3.8,0000*51
Latitude = 0242.2444
Longitude = 10154.3678
$GPGSA,A,3,03,06,32,23,14,16,30,31,,,,,2.3,1.3,1.9*3F
Latitude = null
Longitude = null
$GPRMC,034134.000,A,0242.2444,N,10154.3678,E,0.00,29.98,090712,,,D*54
Latitude = null
```

Figure 20: GPS Data Streams

5. TRANSMISSION SECTION

To achieve transmission, the application implements a P2P communication method using UDP. This enables the application to be both the broadcaster and the receiver. In any event the application needs to broadcasts a message it is able to do so and at the same time receives the message itself. This is due to the application is being used to both listening and sending data.

Each of the sending part of the code and also the receiving part of the code ran on different threaded. This threading method in effect makes the application into 2 running programs, one to receive and another to send.

As there are no sensors to be interfaced with the application, a broadcast warning section is used to simulate a message being sent. In future development, this section is replaced with sensors which selects the event that occurs and broadcasts it automatically.

The message being sent is configured to
/ Sender IP: Latitude, Longitude, Time, Date, Warning
Additional information can be appended to the message with ease.

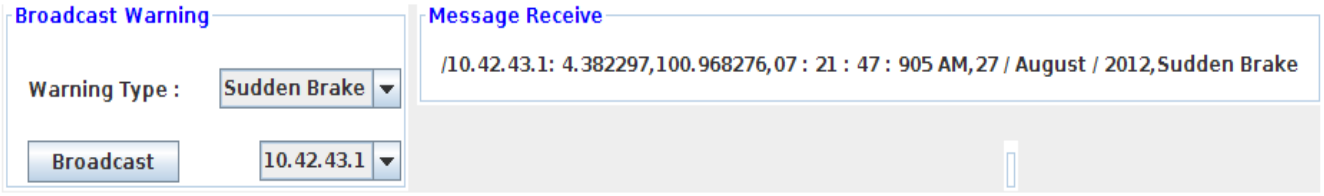

Figure 21: Broadcast Control & Message Data Streams

When a listening vehicle receives this message, the application takes the information and processes it to display the warning, and also link it with the map.

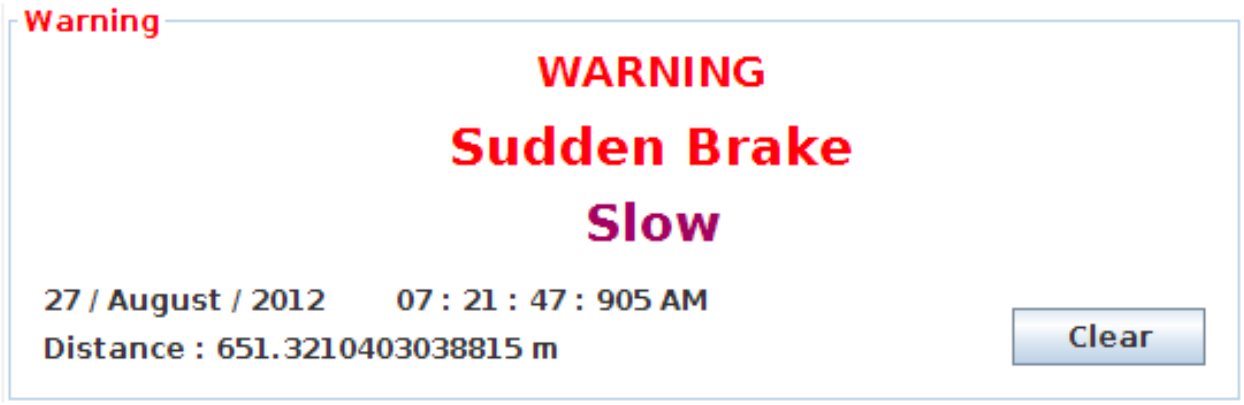

Figure 22: Warning Display

CHAPTER 5 CONCLUSION AND RECOMMENDATION

1. CONCLUSION

The effort to reduce accidents occurrence across the world are increasing. This application is one of them. The implementation of this application on the road users could help the driver to become aware of what is ahead of them several seconds earlier which could save their lives. The development of this application is a success with all the objectives achieved. This application will assist the researches being done in VANETs as a platform to implement the communication layer on.

.

2. RECOMMENDATION

- *Implementation of sensors with the application.*
- *Add functionality to determine the direction of event in relatives to driver.*
- *Add functionality to determine the intended recipient in relatives to driver and event.*

REFERENCES

[1] Malaysia Automotive Info, *Summary of Sales & Production Data*. [Online]. Available: http://www.maa.org.my/info_summary.htm. [Accessed: July. 28, 2012].

[2] Y. Sahat, "17 maut di jalan raya setiap hari," *Utusan Malaysia*, para. 1, June 19, 2011. [Online]. Available: http://www.utusan.com.my/utusan/info.asp?y=2011&dt=0619&pub=Utusan_Malaysia&sec=Dalam_Negeri&pg=dn_04.htm. [Accessed Aug. 13, 2012].

[3] World Health Organization, *Global Plan for the Decade of Action for Road Safety 2011-2020*, 2011. [Online]. Available from http://www.who.int/roadsafety/decade_of_action/plan/en/index.html. [Accessed: Aug. 1, 2010].

[4] R. Bishop, *Intelligent Vehicle Technology and Trends*, 1st ed., 685 Canton Street Norwood, MA 02062 USA: Artech House Publishers, 2005, p. 344.

[5] X. Wang, *Mobile Ad-Hoc Networks: Applications*, ed., Janeza Trdine 9, 51000 Rijeka, Croatia: InTech, January, 2011 , p. 514.

[6] Prof. Dr.-Ing. Ferit Küçükay,Dipl.-Ing. Janine Bergholz, *Driver Assistant Systems*, Institute of Automotive Engineering, TU Braunschweig, Hans-Sommer- Str. 4, 38106 Braunschweig, 2004

[7] K. Bilstrup, A. Bohm, K. Lidstrom, M. Jonsson, T. Larsson, L. Stranden, and H. Zakizadeh, *Vehicle Alert System*, Centre of Research on Embedded System, Halmstad University, Sweden; SP Technical Research Institute of Sweden, Boras, Sweden; Volvo Technology Corporation, Goteborg, Sweden, 2007

[8] J. Stefan, "Navigating with GPS," *CIRCUIT CELLAR, The Magazine for Computer Applications*, Issue 123 , October 2000.[Online]. Available:www.opengpstracker.org/Stefan123.pdf. [Accessed 25 Jul 2012]

[9] GARMIN, *What is GPS?*. [Online]. Available : http://www8.garmin.com/aboutGPS/. [Accessed: Oct. 24, 2011].

[10] J. Person, "Writing GPS Applications," *Dr Dobb's Embedded Systems*, 1 Jan 2005.[Online]. Available:http://www.drdobbs.com/embedded-systems/writing-gps-applications/229100152#rl3. [Accessed 24 Apr 2012]

[11] J. Person, "Writing Your Own GPS Applications: Part 2," *developerFusion*, 22 Oct 2008.[Online]. Available: http://www.developerfusion.com/article/4652/writing-your-own-gps-applications-part-2/. [Accessed 24 Apr 2012]

[12] H. Stratil, et al, "Cooperative Vehicle Infrastructure Systems," *Reference Execution Platform*. pp. 63, 30 Jun 2009. [Online]. [Accessed 8 Aug 2011].

Appendix I

Statistic Road Accident 2001 - 2010

Statistic Road Accident 2001-2010

http://www.jkjr.gov.my

PERANGKAAN KEMALANGAN JALAN RAYA BAGI TAHUN 2001-2010											
TAHUN		**2001**	**2002**	**2003**	**2004**	**2005**	**2006**	**2007**	**2008**	**2009**	**2010**
JENIS KEMALANGAN INJURI	*Kemalangan Maut*	*5,230*	*5,378*	*5,634*	*5,674*	*5,604*	*5,711*	*5,672*	*5,952*	*6,218*	*6,260*
	Kemalangan Parah	*6,942*	*6,696*	*7,163*	*7,444*	*7,600*	*7,375*	*7,384*	*7,020*	*6,978*	*6,002*
	Kemalangan Ringan	*30,684*	*30,259*	*31,357*	*33,413*	*25,928*	*15,596*	*13,979*	*12,893*	*12,072*	*10,408*
JUMLAH KEMALANGAN INJURI		**42,856**	**42,333**	**44,154**	**46,531**	**39,132**	**28,682**	**27,035**	**25,865**	**25,268**	**22,670**
JUMLAH KEMALANGAN TANPA INJURI (ROSAK SAHAJA)		*222,319*	*237,378*	*254,499*	*280,283*	*289,136*	*312,550*	*336,284*	*347,182*	*372,062*	*391,751*
JUMLAH KEMALANGAN		*265,175*	*279,711*	*298,653*	*326,814*	*328,268*	*341,232*	*363,319*	*373,047*	*397,330*	*414,421*
JENIS INJURI	*Mati*	*5,854*	*5,891*	*6,286*	*6,228*	*6,188*	*6,287*	*6,282*	*6,527*	*6,745*	*6,872*
	Cedera Parah	*8,689*	*8,425*	*9,040*	*9,229*	*9,397*	*9,254*	*9,273*	*8,866*	*8,849*	*7,781*
	Cedera Ringan	*35,974*	*35,236*	*37,415*	*38,631*	*31,429*	*19,884*	*18,444*	*16,901*	*15,823*	*13,616*
JUMLAH INJURI		**50,517**	**49,552**	**52,741**	**54,088**	**47,014**	**35,425**	**33,999**	**32,294**	**31,417**	**28,269**
INDEKS KEMATIAN JALAN RAYA	*Setiap 100 ribu penduduk*	**24.60**	**24.10**	**25.10**	**24.30**	**23.70**	**23.60**	**23.10**	**23.60**	**23.80**	**24.20**
	Setiap 10 ribu kenderaan berdaftar	**5.17**	**4.88**	**4.88**	**4.51**	**4.18**	**3.98**	**3.73**	**3.63**	**3.55**	**3.40**
	Setiap 1 Bilion VKT	**23.93**	**22.71**	**22.77**	**21.1**	**19.58**	**18.69**	**17.6**	**17.2**	**17.28**	**17.28**
Sumber : PDRM / MIROS											

Appendix II

General Road Accident Data in Malaysia (1995 – 2010)

General Road Accident Data in Malaysia (1995 – 2010)
MALAYSIAN INSTITUTE OF ROAD SAFETY RESEARCH (MIROS)
http://www.miros.gov.my/web/guest/road

Year	Population	Vehicles Registered	Vehicle Involved	Road Length	Road Accidents	Road Casualties	Road Deaths	Vehicle Ownership (*Person per vehicle*)
1995	*20,096,700*	*6,802,375*	*275,430*	*62,221*	*162,491*	*52,152*	*5,712*	*3.0*
1996	*21,169,000*	*7,686,684*	*325,915*	*64,511*	*189,109*	*53,475*	*6,304*	*2.8*
1997	*21,665,600*	*8,550,469*	*373,526*	*66,108*	*215,632*	*56,574*	*6,302*	*2.5*
1998	*22,179,500*	*9,141,357*	*366,932*	*66,741*	*211,037*	*55,704*	*5,740*	*2.4*
1999	*22,711,900*	*9,929,951*	*390,674*	*67,069*	*223,166*	*52,937*	*5,794*	*2.3*
2000	*23,263,600*	*10,598,804*	*441,386*	*68,770*	*250,429*	*50,200*	*6,035*	*2.2*
2001	*23,795,300*	*11,302,545*	*483,351*	*74,217*	*265,175*	*50,473*	*5,849*	*2.1*
2002	*24,526,500*	*12,068,144*	*507,995*	*74,641*	*279,711*	*49,552*	*5,891*	*2.0*
2003	*25,048,300*	*12,819,248*	*555,634*	*79,667*	*298,653*	*52,741*	*6,286*	*2.0*
2004	*25,580,000*	*13,828,889*	*596,533*	*71,814*	*326,815*	*54,091*	*6,228*	*1.8*
2005	*26,130,000*	*15,026,660*	*581,136*	*71,814*	*328,264*	*47,012*	*6,200*	*1.7*
2006	*26,640,000*	*15,790,732*	*635,024*	*72,781*	*341,252*	*35,425*	*6,287*	*1.7*
2007	*27,170,000*	*16,813,943*	*668,173*	*73,032*	*363,319*	*33,999*	*6,282*	*1.6*
2008	*27,730,000*	*17,971,901*	*671,078*	*73,419*	*373,071*	*32,274*	*6,527*	*1.5*
2009	*28,310,000*	*19,016,782*	*705,623*	*100,002*	*397,330*	*31,417*	*6,745*	*1.5*
2010	*28,910,000*	*20,188,565*	*760,433*	*111,378*	*414,421*	*28,269*	*6,872*	*1.4*

Appendix III

Summary Of New Passenger & Commercial Vehicles Produced And Registered In Malaysia For The Year 1980 To Ytd March 2011

SUMMARY OF NEW PASSENGER & COMMERCIAL VEHICLES PRODUCED AND REGISTERED IN MALAYSIA FOR THE YEAR 1980 TO YTD MARCH 2011

Year	*Passenger Cars*	*Commercial Vehicles*	*4x4 Vehicles*	*Total Vehicles*
1980	*80,420*	*16,842*	-	*97,262*
1985	*63,857*	*26,742*	*4,400*	*94,999*
1990	*106,454*	*51,420*	*7,987*	*165,861*
1995	*224,991*	*47,235*	*13,566*	*285,792*
2000	*282,103*	*33,732*	*27,338*	*343,173*
2005	*416,692*	*97,820*	*37,804*	*552,316*
2006	*366,738*	*90,471*	*33,559*	*490,768*
2007	*442,885*	*44,291*	-	*487,176*
2008	*497,459*	*50,656*	-	*548,115*
2009	*486,342*	*50,563*	-	*536,905*
2010	*543,594*	*61,562*	-	*605,156*
YTD March 2011	*142,546*	*15,887*	-	*158,433*

Appendix IV
JAVA Codes of the Project

Main Program Codes

```
/**...*/
public WarningSystemUI()
{

    initComponents();
    RUNTIME();
    WEATHER();

}
```

Time and Date Program Codes

```
public void TIME()
{
    Date now = new Date();      // get current time
    SimpleDateFormat jam = new SimpleDateFormat("HH");
    SimpleDateFormat minit = new SimpleDateFormat("mm");
    SimpleDateFormat saat = new SimpleDateFormat("ss");
    SimpleDateFormat msaat = new SimpleDateFormat("SSS");
    SimpleDateFormat ampm = new SimpleDateFormat("a");
    try
    {
        String hourString = jam.format(now);
        String minuteString = minit.format(now);
        String secondString = saat.format(now);
        String msecondString = msaat.format(now);
        String ampmString = ampm.format(now);
        hour.setText(hourString);
        hour.setForeground(Color.black);
        minute.setText(minuteString);
        minute.setForeground(Color.black);
        second.setText(secondString);
        second.setForeground(Color.black);
        msecond.setText(msecondString);
        msecond.setForeground(Color.black);
        pmam.setText(ampmString);
        pmam.setForeground(Color.black);
    }
    catch (IllegalArgumentException iae)
    {
        hour.setForeground(Color.red);
        hour.setText("Error: " + iae.getMessage());
        minute.setForeground(Color.red);
        minute.setText("Error: " + iae.getMessage());
        second.setForeground(Color.red);
        second.setText("Error: " + iae.getMessage());
        msecond.setForeground(Color.red);
        msecond.setText("Error: " + iae.getMessage());
        pmam.setForeground(Color.red);
        pmam.setText("Error: " + iae.getMessage());
    }

}
```

```
public void DDMMYY()
{
    Date today = new Date();
    SimpleDateFormat tarikh = new SimpleDateFormat("dd");
    SimpleDateFormat bulan = new SimpleDateFormat("MMMMM");
    SimpleDateFormat tahun = new SimpleDateFormat("YYYY");
    SimpleDateFormat hari = new SimpleDateFormat("EEEEEE");
    try
    {
        String dateString = tarikh.format(today);
        String monthString = bulan.format(today);
        String yearString = tahun.format(today);
        String dayString = hari.format(today);
        date.setForeground(Color.black);
        date.setText(dateString);
        month.setForeground(Color.black);
        month.setText(monthString);
        year.setForeground(Color.black);
        year.setText(yearString);
        day.setForeground(Color.black);
        day.setText(dayString);
    }

    catch (IllegalArgumentException iae)
    {
        date.setForeground(Color.red);
        date.setText("Error: " + iae.getMessage());
        month.setForeground(Color.black);
        month.setText(monthString);
        year.setForeground(Color.black);
        year.setText(yearString);
        day.setForeground(Color.black);
        day.setText(dayString);
    }

    catch (IllegalArgumentException iae)
    {
        date.setForeground(Color.red);
        date.setText("Error: " + iae.getMessage());
        month.setForeground(Color.red);
        month.setText("Error: " + iae.getMessage());
        year.setForeground(Color.red);
        year.setText("Error: " + iae.getMessage());
        day.setForeground(Color.red);
        day.setText("Error: " + iae.getMessage());
    }
}
```

Weather and Temperature Program Codes

```
public void WEATHER()
{
    xml_name = new ArrayList();
    xml_content = new ArrayList();

    try
    {
        double getlat  = 4.620229;
        double getlong = 101.112556;
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser saxParser = factory.newSAXParser();
        long latitude = (long) (getlat * 1000000);
        long longtitude = (long) (getlong * 1000000);
        String url;

        DefaultHandler handler = new DefaultHandler()
        {
            public void startElement(String uri, String localName,String qName,Attributes attributes) throws SAXException
            {
                xml_name.add(xml_index,qName);
                xml_content.add(xml_index,attributes.getValue("data"));
                System.out.println("Start Element : " + qName );
                System.out.println("\tData : " + attributes.getValue("data"));

                xml_index++;
            }

            public void endElement(String uri, String localName,String qName) throws SAXException
            {
                /* */
            }

        };

        url = "http://www.google.com/ig/api?weather=,," + latitude + "," + longtitude;
        System.setProperty("java.net.useSystemProxies", "true");
        System.out.println("Requesting " + url);
        saxParser.parse(url, handler);

        System.out.println("\n\nDisplay content of xml array");
        for(int i=0; i<30; i++)
        {
            System.out.println("\nindex : " + i);
            System.out.println("Name : " + xml_name.get(i));
            System.out.println("Content : " + xml_content.get(i));
        }

    }
    catch (ParserConfigurationException | SAXException | IOException e)
    {
        e.printStackTrace();
    }
    WEATHER_DISP();
    TEMPERATURE_DISP();
}
```

```
public void WEATHER_DISP()
{
    // index 11 is weather condition

    String w_cond = (String) xml_content.get(11);
    String w_picname;
    String w_picpath;
    w_pic = new ImageIcon();

        if (w_cond == "Mostly Cloud")
        w_picname = "17.gif";
        else if (w_cond == "Partly Sunny")
        w_picname = "16.gif";
        else if (w_cond == "Scattered Thunderstorm")
        w_picname = "11.gif";
        else if (w_cond == "Showers")
        w_picname = "09.gif";
        else if (w_cond == "Scattered Showers")
        w_picname = "09.gif";
        else if (w_cond == "Rain And Snow")
        w_picname = "12.gif";
        else if (w_cond == "Overcast")
        w_picname = "16.gif";
        else if (w_cond == "Light Snow")
        w_picname = "13.gif";
        else if (w_cond == "Freezing Drizzle")
        w_picname = "13.gif";
        else if (w_cond == "Chance Of Rain")
        w_picname = "09.gif";
        else if (w_cond == "Sunny")
        w_picname = "01.gif";
        else if (w_cond == "Clear")
        w_picname = "01d.gif";
        else if (w_cond == "Mostly Sunny")
        w_picname = "02d.gif";
        else if (w_cond == "Partly Cloudy")
        w_picname = "03d.gif";
        else if (w_cond == "Mostly Cloudy")
        w_picname = "04.gif";
        else if (w_cond == "Chance Of Storm")
        w_picname = "11.gif";
        else if (w_cond == "Rain")
        w_picname = "10.gif";

        w_pic = getweathericon("weatherimages/" + w_picname);
        w_pic.setDescription(w_cond);
        imgweather.setIcon(w_pic);
        imgweather.setText("");
        imgweather.setForeground(Color.black);
        txtweather.setText(w_cond);
        txtweather.setForeground(Color.black);


}
```

```
public void TEMPERATURE_DISP()
{
    // index 12 is temperature in farenheit
    // index 13 is temperature in celcius

    String temperature = (String) xml_content.get(13);
    celcius.setText(temperature + "°C");
    celcius.setForeground(Color.black);
    c_high.setText("H : " + "°C");
    c_high.setForeground(Color.black);
    c_low.setText("L : " + "°C");
    c_low.setForeground(Color.black);

}
```

Serial Communication Program Codes

```
public void serialEvent(SerialPortEvent event)
{
    switch (event.getEventType())
    {
        case SerialPortEvent.BI:
        case SerialPortEvent.OE:
        case SerialPortEvent.FE:
        case SerialPortEvent.PE:
        case SerialPortEvent.CD:
        case SerialPortEvent.CTS:
        case SerialPortEvent.DSR:
        case SerialPortEvent.RI:
        case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
        break;

        case SerialPortEvent.DATA_AVAILABLE:
       //     gpsData.giveReader(inputStream);
            readBuffer = new byte[1];

            try
            {
                while (inputStream.available()>0)
                {
                    int numBytes = inputStream.read(readBuffer);
                    //System.out.print(new String(readBuffer));
                    warning.GETGPSDATA(new String(readBuffer));
                }
            }
            catch (IOException e)
            {
                System.out.println("IO Exception in SerialEvent()");
                return;
            }

        break;
    }

    if (inputStream != null) try {
        inputStream.close();
    } catch (IOException ex) {
        Logger.getLogger(SerialRead.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Map Program Codes

```
public static JComponent createContent() {
  JPanel contentPane = new JPanel(new BorderLayout());
  JPanel webBrowserPanel = new JPanel(new BorderLayout());
  webBrowserPanel.setBorder(BorderFactory.createTitledBorder("Native Web Browser component"));
  final JWebBrowser webBrowser = new JWebBrowser();
  webBrowser.navigate("http://www.google.com");
  webBrowserPanel.add(webBrowser, BorderLayout.CENTER);
  contentPane.add(webBrowserPanel, BorderLayout.CENTER);
  // Create an additional bar allowing to show/hide the menu bar of the web browser.
  JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 4, 4));
  JCheckBox menuBarCheckBox = new JCheckBox("Menu Bar", webBrowser.isMenuBarVisible());
  menuBarCheckBox.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
      webBrowser.setMenuBarVisible(e.getStateChange() == ItemEvent.SELECTED);
    }
  });
  buttonPanel.add(menuBarCheckBox);
  contentPane.add(buttonPanel, BorderLayout.SOUTH);
  return contentPane;
}

/* Standard main method to try that test as a standalone application. */
public static void main(String[] args) {
  NativeInterface.open();

  UIUtils.setPreferredLookAndFeel();
  SwingUtilities.invokeLater(new Runnable() {
    public void run() {
      JFrame frame = new JFrame("DJ Native Swing Test");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.getContentPane().add(createContent(), BorderLayout.CENTER);
      frame.setSize(800, 600);
      frame.setLocationByPlatform(true);
      frame.setVisible(true);
    }
  });

}
```

V2V Program Codes Server Side

```
package warningsystem;

import java.net.*;

public class MulticastServer {

    private DatagramSocket socket;
    private InetAddress group;

    public MulticastServer() {
        try {
            socket = new DatagramSocket(4445);
            group = InetAddress.getByName("230.0.0.1");//230
            System.out.println("Server initialized");
        } catch (Exception e) {
            System.out.println("Unable to initiate server!");
            //e.printStackTrace();
        }
    }

    public void send(String input) {
        try {
            byte[] buf = input.getBytes();
            DatagramPacket packet = new DatagramPacket(buf, buf.length, group, 4446);
            socket.send(packet);
        } catch (Exception e) {
            System.out.println("Network is unreachable!");
            //e.printStackTrace();
        }
    }
}
```

V2V Program Code Client Side

```
package warningsystem;

import java.io.*;
import java.net.*;
import java.util.StringTokenizer;

public class MulticastClient extends Thread {

    private boolean Alive = false;
    private int MAX_BYTES = 512;
    private MulticastSocket socket;
    private InetAddress address;
    WarningSystemUI warning;
    String[] warndata = {"0","0","0","0","0","0","0"};

    public MulticastClient() {
        try {
            socket = new MulticastSocket(4446);
            address = InetAddress.getByName("230.0.0.1");//230
            socket.joinGroup(address);
            System.out.println("Client initialized");
        } catch (Exception e) {
            System.out.println("Unable to start listening client!");
        }
        Alive = true;
        start();
    }

    public void stopListening() {
        try {
            Alive = false;
            socket.leaveGroup(address);
            socket.close();
            System.out.println("Client stopped listening");
        } catch (Exception e) {
            System.out.println("Error while stopping listening client");
        }
    }

    public void instancePassing(WarningSystemUI w)
    {
        warning = w;
    }

    public void run() {
        try {
            DatagramPacket packet;

            while (Alive) {
                byte[] buf = new byte[MAX_BYTES];
                packet = new DatagramPacket(buf, buf.length);
                socket.receive(packet);
                String received = new String(packet.getData(), 0, packet.getLength());
                warndata = token(received);
                warning.DISPLAYWARNING(warndata[0], warndata[1], warndata[2], warndata[3], warndata[4],packet.getAddress()+": "+received);
                System.out.println("Data received from server (" + packet.getAddress() + "): " + received);
            }


        } catch (IOException e) {
            System.out.println("Unable to start listening client!");
        }
    }

    public String[] token(String str)
    {
        StringTokenizer st = new StringTokenizer(str,",");

        String[] receiveddata = new String[6];
        System.out.println("********* GPS DATA RECEIVED *********");
        receiveddata[0] = st.nextToken(); // Latitude
        receiveddata[1] = st.nextToken(); // Longitude
        receiveddata[2] = st.nextToken(); // Time
        receiveddata[3] = st.nextToken(); // Date
        receiveddata[4] = st.nextToken(); // Condition

        return receiveddata;
    }
}
```

V2V Program Code Broadcast and Distance Calculation

```
private void buttonbroadcastActionPerformed(java.awt.event.ActionEvent evt) {
   server.send(gps_coord_lat+","+gps_coord_lon+","+send_time+","+send_date+","+condition);
}
```

```
public void DISPLAYWARNING(String ClientLat, String ClientLon, String ClientTime, String ClientDate, String ClientCondition, String Client
{
    w_coord_lat = ClientLat;
    w_coord_lon = ClientLon;
    warndataset.setText(ClientDataset);
    warndate.setText(ClientDate);
    warntime.setText(ClientTime);
    envcondition.setText(ClientCondition);


    if(ClientCondition.equals("Fog"))
    {advice.setText("Reduced Visibility");}

    if(ClientCondition.equals("Rain"))
    {advice.setText("Slippery Road");}

    if(ClientCondition.equals("Snow"))
    {advice.setText("Icy Road");}

}
```

```
public String GETDISTANCE()
{
    double equatorial = 6378135; // radius meter
    double polarial   = 6356750; // radius meter

    double self_lat = Double.parseDouble(gps_coord_lat);
    double self_lon = Double.parseDouble(gps_coord_lon);
    double warn_lat = Double.parseDouble(w_coord_lat);
    double warn_lon = Double.parseDouble(w_coord_lon);

    double diff_lat = (warn_lat-self_lat)/(180*Math.PI);
    double diff_lon = (warn_lon-self_lon)/(180*Math.PI);
    double a = ((Math.sin((diff_lat / 2)) * Math.sin((diff_lat / 2))) + (Math.cos(warn_lat) * (Math.sin((diff_lon / 2)) * Math.sin((diff_l
    double c = (2 * Math.atan2(Math.sqrt(a), Math.sqrt((1 - a))));

    double numerator   = Math.pow((equatorial * (polarial * Math.cos((self_lat / (180 * Math.PI))))), 2);
    double denumerator = (Math.pow((equatorial * Math.cos((self_lat / (180 * Math.PI)))), 2) + Math.pow((polarial * Math.sin((self_lat / (

    double radius = Math.sqrt(numerator/denumerator);
    double distance = (radius*c);

    return String.valueOf(distance);

}
```