

GPS TRAFFIC NAVIGATION SYSTEM
BY
DANIEL KHOR SAY HOU

A REPORT
SUMMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF INFORMATION TECHNOLOGY (HONS)
COMMUNICATION AND NETWORKING
Faculty of Information and Communication Technology
(Perak Campus)

MAY 2012

DANIEL KHOR SAY HOU

MAY 2012

REPORT STATUS DECLARATION FORM

Title: GPS Traffic Navigation System

Academic Session: May 2012

I DANIEL KHOR SAY HOU
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:

10, JALAN PU 12/5A
TAMAN PUCHONG UTAMA
47100 PUCHONG. SELANGOR

GOH HOCK GUAN
Supervisor's name

Date: _____

Date: _____

GPS TRAFFIC NAVIGATION SYSTEM
BY
DANIEL KHOR SAY HOU

A REPORT
SUMMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF INFORMATION TECHNOLOGY (HONS)
COMMUNICATION AND NETWORKING
Faculty of Information and Communication Technology
(Perak Campus)

MAY 2012

DECLARATION OF ORIGINALITY

I declare that this report entitled “**GPS Traffic Navigation System**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : Daniel Khor Say Hou

Date : _____

ACKNOWLEDGEMENT

I would like to acknowledge and extend my heartfelt gratitude to my supervisor, Mr Goh Hock Guan, who has guided me patiently throughout the entire project timeline with his advice and meet me at some untimely meeting which mostly are random popping up in front of his office. Without his help and his idea for this project, God knows how this project would end up to.

I also owe a sincere and earnest thankfulness to Mr Liew Shiuh Deh, for the invaluable advises on the CeedTec MPC824-M3-EP embedded routers. Without his help, I doubt that I could get the OpenWrt custom patch application to be installed and that none of it would even work.

I would like to thank the laboratory officer, Mr Chan Wei Lu, for his help and patience for letting me stay in for extra hours despite it is already after his working hours for most of the days. And I would like to thank my senior, Mr Azmi, for giving me some ideas on the algorithm and some general encouragement for this project.

And lastly, it is a great pleasure to thank everyone else that has in general, to have direct or indirect of support to me for the completion of this project.

ABSTRACTS

This report presents the development of a GPS traffic navigation system for the Android platform as smartphone users has increases rapidly in these recent years. The GPS traffic navigation system developers have started to pay attention towards the need of smartphone user when nearly all newer smartphone has GPS ability installed. This provides a motivation for developer to design and provide the needs for smartphone user. As a result, several developers have developed several kind of GPS traffic navigation system that caters for the smartphone users. With the recent advancement in the GPS traffic navigation system development, some developers has come out with GPS traffic navigation service with extra service which the service has traffic information gathered and analysed. This will provide the service buyer a way to get over the traffic and avoid traffic congestion.

Therefore, the GPS traffic navigation system in this project relies on the traffic sensor that would broadcast the traffic information via the wireless mesh network that is setup. The Wi-Fi relay point is also a Wi-Fi access point for smartphone to connect instantly without any issue as the message is a broadcast which would flood the network for a period of time. This allow the GPS traffic navigation system installed in the smartphone to read the message and mark given GPS coordination as invalid which would trick the GPS traffic navigation system to reroute the user. Hence, this would provide user hassle free driving when knowing the GPS traffic navigation system would reroute with an action prompt in order to proceed. Though, to keep in mind is that this application is mainly cater for Android compatible smartphone.

TABLE OF CONTENTS

TITLE	ii
DECLARATION OF ORIGINALITY	iii
ACKNOWLEDGEMENT	iv
ABSTRACTS	v
TABLE OF CONTENTS	vi
TABLE OF FIGURES	x
LIST OF TABLES	xii
LIST OF ABBREVIATIONS	xiii
CHAPTER 1	1
1.1 Project Background	1
1.1.1 Summary and Overview of the Project	1
1.1.2 Problem Statement and Motivation	2
1.2 Project Scope and Objective	3
1.2.1 Project Scope	3
1.2.2 Objectives	5
1.3 Project Contribution	6
1.4 Report Chapters Overview	6
CHAPTER 2	8
2.1 Human Navigation History	8
2.2 Data Collection – Existing Solution	8
2.3 Literature Review	9
2.3.1 Whitepaper – HD Traffic and IQ Routes	10
2.3.2 Artificial Intelligence in GPS Navigation Systems	10
2.3.3 GPS Based Urban Guidance	11
2.3.4 Real-time Vehicle Route Guidance Using Vehicle-to-vehicle Communication	11
2.4 Critical Remarks on Previous Works	12

2.5	Summary	15
CHAPTER 3		16
3.1	Methodology	16
3.1.1	Waterfall Model	16
3.1.2	Incremental and Iterative Development	17
3.1.3	Extreme Programming	18
3.1.4	Selecting a Development Model	19
3.2	Requirement Specifications	20
3.2.1	System Requirements	20
3.2.2	System Performance Definition	21
3.2.3	System Verification Plan	21
3.3	Implementation Issues	22
3.4	Timeline	22
3.5	Technology Involve	27
3.6	Budget Plan	33
3.7	Summary	34
CHAPTER 4		36
4.1	Overview of System Design	36
4.2	Hardware Design	36
4.2.1	CeedTec MPC8241-M3-EP	36
4.2.2	Linksys WRT54G (WRT54Gv2 / WRT54GL)	38
4.3	Software Design	40
4.3.1	General Concepts	41
4.3.2	Androzie	42
4.3.3	Androzie “NavigationService” Background Service	43
4.3.4	Androzie Distance Calculation (Vincenty’s Formula)	44
4.3.5	Wi-Fi Monitoring Algorithm	45

4.3.6	“SmartRoute” Background Service	46
4.3.7	Create Log File Algorithm	47
4.3.8	Coverage Area Algorithm	48
4.4	Network Design	48
4.4.1	Wireless Mesh Network Topology	49
4.5	Summary and Concluding Remarks about Chapter 4	50
CHAPTER 5		51
5.1	Software Design Implementation	51
5.1.1	“SmartRoute” System	51
5.1.2	Creating Log File	53
5.1.3	Dummy Server Broadcaster	53
5.2	Network Design Explanations	54
5.2.1	OpenWrt Configuration	54
5.3	Software Used In System Implementation	56
5.3.1	Android Development Installation	57
5.4	Summary and Concluding Remarks For Chapter 5	59
CHAPTER 6		60
6.1	System Verification	60
6.2	Case Study	62
6.2.1	Environment Description	62
6.3	Objective Achievements	63
6.4	Issues	64
6.4.1	Android Application Lifecycle And Big Project	64
6.4.2	Wireless Connectivity	65
6.4.3	Maps and Waypoints	65
6.5	Summary and Concluding Remarks For Chapter 6	66
CHAPTER 7		67

7.1 Project Conclusion	67
7.2 Recommendation	67
REFERENCE	69
APPENDICE A – GANTT CHART	A1
APPENDICE B – SAMPLE LOG FILES	B1
APPENDICE C – BI-WEEKLY LOG	C1

TABLE OF FIGURES

FIGURE 1 - INITIAL ROUTE TO DESTINATION BEFORE RECEIVE TRAFFIC INFORMATION .	4
FIGURE 2 - MODIFIED ROUTE TO DESTINATION AFTER RECEIVE TRAFFIC INFORMATION	4
FIGURE 3 - SAFELY REACH DESTINATION	5
FIGURE 4 - WATERFALL MODEL	17
FIGURE 5 - ITERATIVE AND INCREMENTAL DEVELOPMENT LIFE CYCLE	17
FIGURE 6 - XP LIFECYCLE	18
FIGURE 7 - STORIES FROM CUSTOMER	19
FIGURE 8 - PROJECT I GANTT CHART PART1	23
FIGURE 9 - PROJECT I GANTT CHART PART2	23
FIGURE 10 - PROJECT I GANTT CHART PART3	24
FIGURE 11- PROJECT II TIMELINE OVERVIEW	24
FIGURE 12 - CEEDTEC MPC8241-M3-EP BLOCK DIAGRAM.....	38
FIGURE 13 - WRT54G BLOCK DIAGRAM (GENERIC) EXTRACTED FROM HTTP://WIKI.OPENWRT.ORG/TOH/LINKSYS/WRT54G	40
FIGURE 14- A SIMPLIFIED ILLUSTRATION OF THE ACTIVITY LIFECYCLE (SOURCE: HTTP://DEVELOPER.ANDROID.COM/TRAINING/BASICS/ACTIVITY- LIFECYCLE/STARTING.HTML)	41
FIGURE 15 - ANDROZIC NAVIGATION ROUTE OVERVIEW	43
FIGURE 16 - ANDROZIC NAVIGATIONSERVICE FLOW CHART.....	44
FIGURE 17 - VINCENTY'S FORMULA (SOURCE: HTTP://WWW.MOVABLE- TYPE.CO.UK/SCRIPTS/LATLONG-VINCENTY.HTML)	45
FIGURE 18 - WI-FI SCAN AND CONNECT ALGORITHM FLOW CHART	46
FIGURE 19 - SMARTROUTE ALGORITHM	47
FIGURE 20 - CREATE LOG FILE ALGORITHM FLOW CHART.....	47
FIGURE 21 - WIRELESS MESH NETWORK	49
FIGURE 22 - SCREEN CAP FOR NAVIGATION	52
FIGURE 23 - SMARTROUTE AND NAVIGATION STARTED.....	52
FIGURE 24 - CRASH ALERT DIALOG BOX.....	53
FIGURE 25 - DUMMY SERVER	54
FIGURE 26 - NODE1 WIRELESS CONFIGURATION	55
FIGURE 27 - NODE1 NETWORK CONFIGURATION	56
FIGURE 28- NODE1 OLSRD CONFIGURATION PAGE	56
FIGURE 29 - ANDROID SDK MANAGER	58

FIGURE 30 - NETSTRESS NETWORK BENCHMARKING TOOL RESULT	61
FIGURE 31 - NODE 2 ACTIVE OLSR NODES OLSR TOPOLOGY (LOWER NUMBER IS BETTER).....	61
FIGURE 32 - FIELD TEST MAP	63

LIST OF TABLES

TABLE 1 - CRITICAL REMARKS	12
TABLE 2 - PROJECT II WORK BREAKDOWN STRUCTURE (WBS).....	24
TABLE 3 - MOTOROLA MILESTONE SPECIFICATION	28
TABLE 4 - SUMMARISE OF CURRENT WORKING PROTOCOL	30
TABLE 5 - INITIAL BUDGET PLAN	33
TABLE 6 - AMENDED PLANNED COST	34
TABLE 7 - FINAL COST	34
TABLE 8 - MPC8241-M3-EP HARDWARE/SOFTWARE SPECIFICATION.....	37
TABLE 9 - LINKSYS WRT54Gv2 / WRT54GL HARDWARE/SOFTWARE SPECIFICATION	39

LIST OF ABBREVIATIONS

2PEED	2 Phase Piecewise Linear Speed Model
ADT	Android Development Tool
AVD	Android Virtual Devices
DOD	US Department of Defence
ETX	Expected Retransmission Count
GPSNAV	GPS Navigation System
GPS	Global Positioning System
GSM	Global System for Mobile Communication
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineering
IETF	Internet Engineering Task Force
IID	Incremental and Iterative Development
KM/H	Kilometers per Hour
OLSR	Optimised Link State Routing Protocol
OLSRd	Optimised Link Stated Routing Protocol Daemon
RAM	Random Access Memory
ROM	Read-Only Memory
SDK	Software Development Kit
SDLC	Software Development Life Cycle
UDP	User Datagram Protocol
V2R	Vehicle-to-Roadside
V2R2	Vehicle-to-Vehicle Real-Time Routing
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad-Hoc Network
XP	Extreme Programming

CHAPTER 1

1.1 Project Background

1.1.1 Summary and Overview of the Project

The advancement of technology has provided humanity the ease of living through the daily hassles from the first founding of rolling wheels to flying aircrafts. Since the ancient time, when the invention of wheels was founded, man has been using horse, donkey, buffalo and other animals as a means of domestic transport for travelling from one location to another location. This has further advance till the current time, the range of vehicle from bicycle to motorcycle; chariots to cars; charcoal power train to electric train; small creepers to gigantic cruisers [1], and every means of transport is to travel from one location to another location.

This lead the humans to always plan in few days in advance for which is the best route to be taken for travelling from here to there; planning always include the cost of travelling, the total distance required and the time required to reach a location. In order for an accurate location detection, the GPS (Global Positioning System) was than invented by the U.S. Department of Defence (D.O.D) in the year 1972 which in the later years in 1983, the GPS is made available to the civilian after a tragedy happened to a Korean civilian airliner was shot down by Russian fighters after accidentally intruded into the Soviet air space [2]. A GPS Navigation System (GPSNAV) has make route planning easier than before, with just a touch on the screen and information of the location can be display with estimated time to arrive and shortest path is an assurance.

However, in this project, to further enhance the ability of a GPSNAV, real time traffic information is gathered on the spot by the underlying traffic information network available. This also enables the user to skip the congested route with an alternative route suggested by the GPSNAV. In order for the GPSNAV to perform such function, the GPSNAV will receive the message via Wi-Fi, and decode the message; information regarding the route congested is decoded, the system will reroute the user with any available alternative route. This will ease the driver from the usual hectic traffic jam and at the same time, save fuels and reduces gas emission from the vehicle.

1.1.2 Problem Statement and Motivation

GPS navigation system has been widely used by the people to navigate from one place to another place through the uses of either an offline map that is included in most GPS devices or an online map that uses mobile data line in order to drive or walk to the destination point with the estimation based on shortest path and speed limit [3] to provide the best route available for the user.

However, road condition changes throughout the day as when traffic congestion during slow traffic cause by heavy rain fall or accident occurs along the route or road blocks setup by the local policemen on the suggested route will result in the estimated time to reach the destination that was calculated by the GPS navigation system is off the mark. Hence, causes the user some delay to reach the destination point. Many users in the world will find that the GPSNAV besides being just a navigation system but wishes that it will also guide the user to avoid traffic jam routes.

Though, there are already few commercial products that support intelligence routing base on traffic information and community efforts to provide data for data mining by the commercial product company [4], these products mostly are too expensive to purchase. TomTom International BV is selling it for £229.99 with bundle of 1 year subscription to the special service and £47.50 annually renewal fee [5]. However, this device and service is not globally available as it only available to certain country or region where Malaysia and other Asia countries are not in the list.

Therefore, the main point of this project is to create an enhance GPSNAV that able to guide and also re-route the user from traffic congested area caused by flood, road accidents, landslides or any conditions cause by natural disaster, whenever possible by using wireless relay points along the road.

The traffic condition can be obtained from several systems such as Integrated Transport Information System (ITIS) and Lembaga Lebuhraya Malaysia (LLM) monitoring camera, radio station that user report the road condition and other traffic monitoring system available in Malaysia. The progress of nationwide implementation by ITIS and LLM will not be covered in this project. We assume that the image or video will be processed autonomously and broadcast the information to the wireless relay and any user that is within the wireless relay coverage will automatically receive

the information and whenever if the given condition is possible, the GPSNAV will re-route the user depending on the road condition of the road further down with a prompt to the user. With this, the users will be aware of the traffic ahead of them and act as a time saver to arrive at any destination.

1.2 Project Scope and Objective

1.2.1 Project Scope

With the problems stated in Section 1.1.2, the aim of this project is to produce an Android based GPS Navigation System (GPSNAV) that is capable to receive traffic information via Wi-Fi. The traffic congestion packet is generated whenever traffic congestion occurs on a particular road segment which enable the GPSNAV to calculate a new route to the destination by comparing the new route time estimation and the current route time estimation with speed patterns of the selected route segment into account to determine whether is the alternate route feasible or not feasible; the GPSNAV will than prompt a message to inform the user of the differences and let the user to decide. Example of a traffic congested due to road construction scenario is shown in Figure 1, Figure 2 and Figure 3. It shows how a traffic congestion packet is receive, decoded, message prompt and reroute user to alternate road. However, considering few constraint factors involving time and resources of the project, the said output is not possible for a single person to handle everything from traffic congestion sensors, complete independent Android GPSNAV application with own digital map and setting up a mesh wireless network topology.

Therefore, the project scope will cover only a manageable scope that can be handled by one person. As such, the project is split into 2 parts, Part 1 and Part 2. The first part is to search for an open source Android GPSNAV application, than create a function to decode a message receive via Wi-Fi and mark the particular road segment in the digital map to be inaccessible. And the system will calculate a new route base on the new digital map information and reroute the user. Therefore, this part is to identify the problem and design a solution to resolve the problem. The later part is to have a practical experiment of the GPSNAV with on-the-road driving experience. A few wireless routers are setup to act as the message relay for the message is sent to the GPSNAV. Figure 1, 2 and 3 shows an example of this project idea.

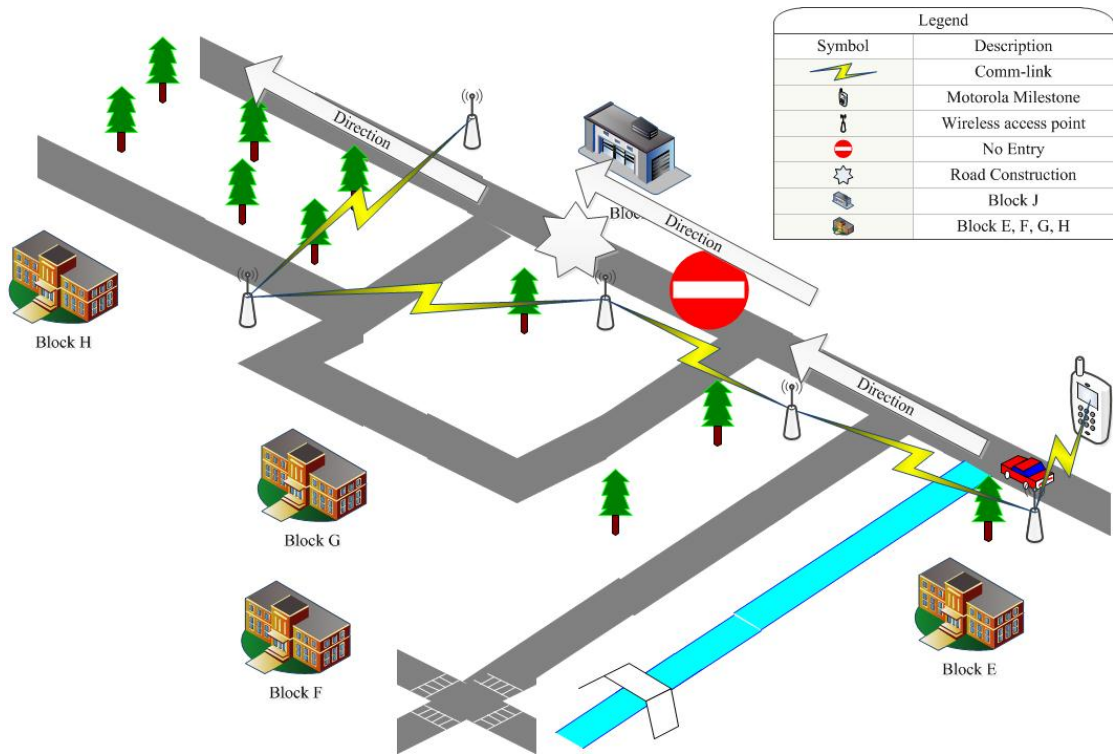


Figure 1 - Initial Route to Destination before Receive Traffic Information

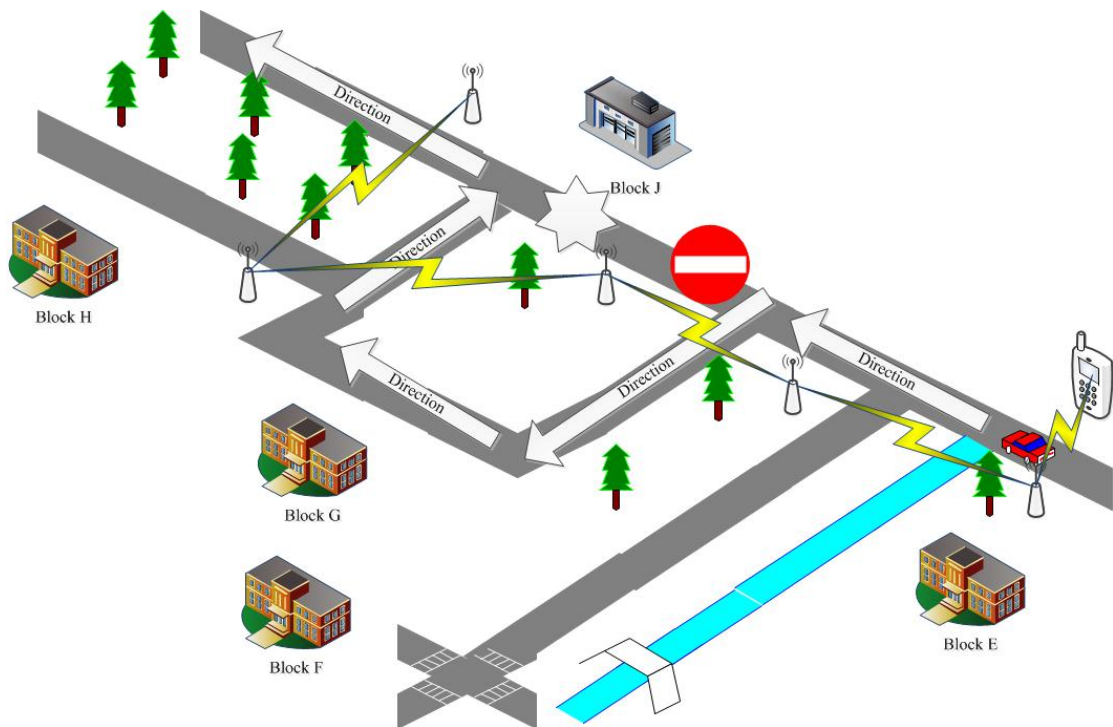


Figure 2 - Modified Route to Destination after Receive Traffic Information

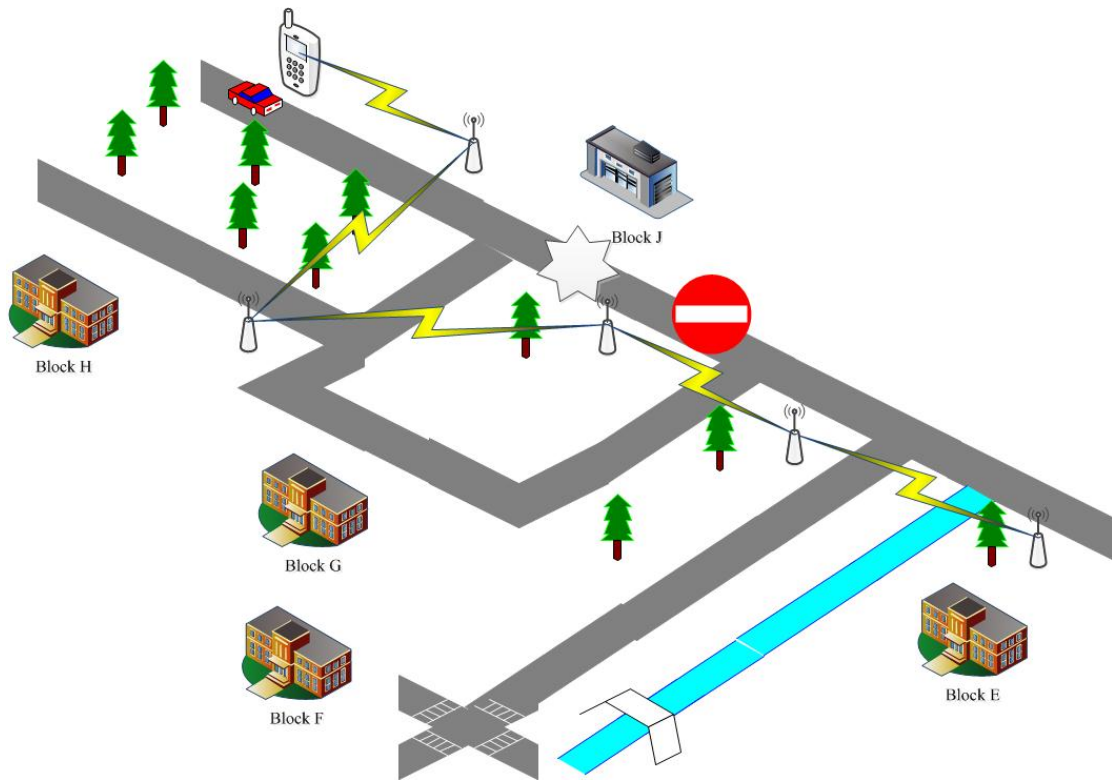


Figure 3 - Safely Reach Destination

1.2.2 Objectives

To achieve the project scope, objectives must be set in order for the project to proceed smoothly. The first objective is to search for an open source Android GPSNAV application; this is to understand the source code and attempt to compile the code for first time run. The application will be tested for routing to any location of the available digital map.

Upon successful routing with the given GPSNAV source code, the second objective is to create a program to send the traffic congested message by using UDP protocol; UDP is a connectionless protocol and less overhead in the packet as compare to TCP, hence, the reliability of the packet is dependent on the application layer which this project will implement for implementation testing purposes. The message contains the GPS coordinate, timestamp and UDP reliability mechanism. The message sending method is either broadcast or multicast.

When the type of message and sending method is determined, the following third objective is modify the source code to include a new function for decoding a message receive via Wi-Fi, the function will check the GPS coordinate, timestamp and reply an

acknowledge message to the sender. The system will mark the specific road segment inaccessible. Thus, this will force the system to reroute via alternative available route to target destination.

Hence, the fourth and last objective is to simulate real time traffic congestion by inputting traffic congested message to the system and let the system to reroute; simulation environment is in Android Virtual Devices (AVD) created via Eclipse Android Development Tool. After a simulation produces positive result, a practical run is performed on the field with multiple wireless router setups to test it on the road.

1.3 Project Contribution

The merit for working on this project with regards on its contribution towards the community is discussed in detail in this particular section. This project primary goal is to create a GPS traffic navigation system that is capable of rerouting user upon receiving traffic congestion information. The motivation for this is explained in details under the problem statement of this chapter. This alone contributes towards the reduction of user frustration on the road and the reduction of air pollutants (not to be congested in the traffic jam could reduce some carbon foot print).

As the supporting of the primary goal, the contribution of the side goal (wireless mesh network) is the usage of relaying important information wirelessly without the need of accessing GPRS/HSDPA network as this would possible to increase the speed of message being relayed. Though, this project is only a stepping stone for exploring more alternatives in GPS traffic navigation system.

1.4 Report Chapters Overview

In this section, a brief overview of the chapters that is available in this report. In the next chapter, Chapter 2 covers the existing solution and literature review of selected articles / journals. The following chapter, Chapter 3 covers the methodology that is the base timeline for this project and the specific requirements for this project. In Chapter 4, 5 and 6 covers the design phase, implementation phase and a collection of issue occur throughout the project timeline. And finally the last chapter, Chapter 7 would be a conclusion for this project and possible future exploration based on this project.

CHAPTER 2

2.1 Human Navigation History

Human navigation existed since the ancient time, back in thousands years ago. During the rise of mercantilism, boats were the easiest and most efficient vessel for transporting goods and used by the Egyptians at around 5500 years ago. The mariners who steered these vessels would require accurate navigation to reach the destination safely. They steer through river or ocean and remained close to shore and uses geographic landmarks to guide them. If they were to venture out to the open water, they could make crude approximations of time and latitude by observing the height of the sun or during night time, they would use the moon, stars, and planets as celestial guides but was notoriously inaccurate. This method was used for centuries until the Chinese created the first magnetized needle compass around 1100CE which prove to be a better and more accurate method.

Fast forwarding to the 15th century, the mariners' astrolabe allows the sailors of that time to determine their latitude while travelling far away from any sight of land. In the 18th century, the marine chronometer allows the sailors to measure latitude and longitude that could remain reliable at sea. In the early 19th century, navigators use radio waves as an alternative to celestial navigation and mainly were used by aircraft. An aircraft would purposely fly into the signals and listen for the continuous sound so that the pilot can adjust the plane accordingly and travel within the centre. About a decade later, radar was used mostly by submarines as it could identify hazards and locate enemy vessels via the radio wave pulse send and receive. As human navigation technology advances, boundless and limitless possibilities are achieved for ground, air and sea navigation system.

2.2 Data Collection – Existing Solution

GPS Navigation System is nothing new; it was first used by the United States military back when GPS was just invented and later was open to the general public for both personal and commercial usage. GPSNAV was widely used for road navigation especially when travelling to unfamiliar places via unfamiliar route. Various types of GPSNAV were developed by various GPSNAV developer with more functionality implementation than its predecessor. Most of these products are standalone devices before the smartphone era starts with their GPSNAV ported for smartphone version.

For an instance of smartphone GPSNAV, an intelligent GPSNAV for iPhone or iPad that was developed by TomTom that uses latest traffic information with its HD Traffic™ that intelligently reroute user whenever the chosen route is not feasible; this application also pack with other functions like TomTom Speed Cameras, Map Share™ and IQ Routes™ [5]. However, the uses of such functions require a high premium from the user in order to access the function.

Another instance of outsmarting the traffic, an intelligent GPSNAV [6] for iPhone, Android, Blackberry and Nokia that uses the data line of the smartphone to access the server where it gather information; information that is shared by the community that also uses this application. It requires the community to summit live information to the server and it notifies other user that the reported road is congested or accident occurs along the road. This application is free and cross mobile platform. However, the accuracy of the information shared by users is not check for its creditability. Hence, false information might be reported and misguided other users of the community.

2.3 Literature Review

With the advancement of GPS accuracy in tracking a moving object with multiple satellites and other supportive probes which further enhance the accuracy of GPS tracking, it is now possible to track smaller targets than before. In addition, the wireless technologies that provides information and data wirelessly without relying too much on wired technologies.

These technologies are widely implemented into the world to ease our daily life; examples of which include smart phone that uses nearly every wireless implementation from GSM, UMTS, Wi-Fi, GPS and other wireless technologies. However, these devices are often packaged as “all-in-one” (meaning it contains a wide variety of hardware) but typically lack support from applications that can take full advantage of the provided hardware, it would be equivalent to being a useless junk that cannot utilise the technologies to the fullest. Hence, in this section, we will review products and articles that have weaknesses that can be improved and further improvise the idea with better ideas. Therefore, we have filtered some commercial products’ whitepapers and articles which could be related to this project’s main objective.

The following articles and whitepaper are picked:

1. Whitepaper - HD Traffic and IQ Routes [7]
2. GPS Based Urban Guidance [8]
3. Artificial Intelligence in GPS Navigation Systems [9]
4. Real-time Vehicle Route Guidance Using Vehicle-to-vehicle Communication [10]

2.3.1 Whitepaper – HD Traffic and IQ Routes

In [7] states that the introduction of time-dynamic navigation concept on its X40 GO series that enable dynamic, historic and real-time traffic and travel-time information obtained from GPS and GSM probes technologies.

The main source of traffic data is gathered from the mobile telecommunication operator in the participating countries and TomTom owned GPS probes that were installed as base TomTom connected devices and TomTom WORK navigation. This guarantees enhanced data and service coverage that is not limited to highways but also covers secondary and arterial urban roads. This has greatly improved the accuracy of route planning for the travelling time, delay time measurements and time-specific data as this provide sufficient time-specific data for better routing from one place to another place by avoiding traffic congested routes.

2.3.2 Artificial Intelligence in GPS Navigation Systems

Although with the sufficient data collected from GPS and GSM probes as in [7], most likely these required a premium in order to use it. Therefore, in [9], the idea is to emulate how does a human plan the route base on the human past experience by assuming that the given person had a perfect memory of every trip they had traverse with the date and time remembered and the duration of each trip taken. Hence, [9] had mention that each person will start off with zero experience for each trip to take, as more experience is gain from day to day travelling, the better it would select the optimal route and accurately estimating the time to reach destination from the starting point.

Basically, the system is adapting to the user driving habits and the traffic environment by capturing the data and stores it for future uses. As the collected data saturated, the

better the system can estimate the route to the destination by using the default data and the new acquire data from user.

2.3.3 GPS Based Urban Guidance

In [8], the authors proposed a new speed pattern model known as two phase piecewise linear speed model (2PEED) by using estimated traffic condition to represent the speed pattern in a road network and a classification-based route guidance approach by using machine learning technique to provide a dynamic routing for drivers.

In this context, understanding the traffic condition is utmost important therefore the authors' identified 2 type of data collection techniques: fixed-based and mobile-based; The former collection techniques uses stationary sensors to collect traffic data but it is limited coverage due to high cost of implementation; The latter data collection technique uses mobile probes to collect traffic data and have better coverage than the former technique since it is more cost-effective, flexible and easy deployment. The authors also use machine learning technique to learn the current and the past speed patterns and traffic condition of related road segments to classify different speed pattern into different classes.

The 2PEED fully utilises the data collected from the probes and self-learning database in order to estimate the traffic more accurately as it contains different road segments speed pattern. Since they uses classification-based route guidance model method, the details allows traffic condition representation to distinguish different traffic conditions into different classifications so that the characteristic of dynamic traffic condition to provide accurate route guidance in a route network.

2.3.4 Real-time Vehicle Route Guidance Using Vehicle-to-vehicle Communication

The state-of-the-art route guidance systems make use of vehicular ad hoc network (VANET) to collect real-time traffic information to find better paths as mention in [10]. There are 2 types of approaches to collect real-time traffic information: infrastructure-based and infrastructure-free. The infrastructure-based approach employs the idea of vehicle-to-roadside (v2r) communication to collect real-time traffic information but this approaches is costly to implement and maintenance due to large road network of a typical urban city which incurred more cost for deploying the

devices while the infrastructure-free approach employs the idea of vehicle-to-vehicle (v2v) communication to have message exchange between vehicles links and it is less costly than v2r approach since it doesn't need to deploy too many sensors and related equipment over the roadside of the urban road network.

Hence, in this article, the authors' proposed a new infrastructure-free communication link which is known as vehicle-to-vehicle real-time routing (v2r2) that consist of 2 algorithms: v2r2 guidance algorithm and v2r2 detour algorithm where the former is to search for the quickest path to guide the vehicle and the latter is used to bypass void areas during the guidance process.

2.4 Critical Remarks on Previous Works

Table 1 - Critical Remarks

Articles	Advantages	Weakness	Comments
Whitepaper - HD Traffic and IQ Routes	<ul style="list-style-type: none"> i. High accuracy for real time traffic information with high probe data penetration from GSM and GPS probes. ii. Speed profiles base on consistent and complete historical speed maps for all navigable roads in the digital map compiled from aggregating anonymous GPS probe data. iii. Avoiding traffic congestion by showing detour or alternative route to destination. 	<ul style="list-style-type: none"> i. Heavily dependent on GSM and GPS probes, if any of these were to break down than the data accuracy will be reduced tremendously. ii. Currently most of the probes are only available in Western Europe and North America; implementing this elsewhere requires a large GSM/GPS probe base. iii. Human privacy issues where people simply refuse to broadcast their position even if it is anonymous may influence available probe population in more sensitive cultures. 	<p>This method of obtaining multi-source data is very costly as it requires paying a premium to the telecommunication providers. In addition, the majority of the current GSM/GPS probe population are mostly in the Western Europe and North America.</p> <p>Implementing this method in Malaysia would require the construction of a basic infrastructure to support this system first which would be costly.</p>

<p>Artificial Intelligence in GPS Navigation Systems</p>	<ul style="list-style-type: none"> i. Self-learning mechanism to learn from the user driving behaviour and traffic environment to improve future route planning. ii. All learnt experience are saved and stored in local database for the system to analyse for providing better route planning. 	<ul style="list-style-type: none"> i. Stereotype learning base as the system only learns from a particular user since there could have alternative route from another user. ii. High capacity storage required for the system to store all learnt experiences and high processing speed required to process the information. These resulted in high power consumption and it is not suitable for mobile users unless the user have it connect to continuous supply of power. 	<p>The scenario presented in this article justifies the learning of a road user during different time and road. However, it doesn't justify the requirement for processing power, storage capacity and power consumption by processes.</p> <p>It is true that processing power is increasing steadily over the years and price for storage capacity is dropping by a factor of two every six months to a year [9] but power efficiency is still an issue with current mobile devices as power resource is scarce when no wall plug available for recharging the power.</p>
<p>GPS Based Urban Guidance</p>	<ul style="list-style-type: none"> i. Real-time traffic information obtains from mobile-based data collection technique for the ease of deployment and low cost. The dynamic nature of mobile-based able to collect traffic information from 	<ul style="list-style-type: none"> i. The dynamic nature itself has the disadvantages as it is not possible to cover every route available in the city. ii. Accuracy is directly reflected by the amount of training data that was supplied and 	<p>This method is good when high amount of training data is available but this might not be practical for on-the-road usage if the user initiate the navigation outside of the traffic information parameters where the</p>

	<p>every route that has being traverse by the users.</p> <p>ii. Each road segment are classified and labelled after the machine learning technique processes complete at the time of query. Hence, the speed pattern is determined for each segment which provide assistant to the route guidance.</p>	<p>stored in the system.</p>	<p>system will use the “old” training data for the classification route guidance to the destination and no rerouting was perform when new traffic information is received. This would result in user possibility of entering a congested route which it wasn’t congested previously.</p>
<p>Real-time Vehicle Route Guidance Using Vehicle-to-vehicle Communication</p>	<p>i. Vehicle-to-vehicle real-time traffic information passing for system to perform routing.</p> <p>ii. Does not require any infrastructure and it is a scalable network.</p>	<p>i. Not every vehicle is equipped with such devices, hence it literally ‘breaks’ the network. Or the equipped vehicle is too far away from each other and the traveling speed is different.</p> <p>ii. It is hard to determine the destination node by solely route query (RQ) and route reply (RR) through vehicle-to-vehicle communication since each vehicle traverse at different speed.</p>	<p>This method could be the best among all if and only if every vehicle is equipped with such device and traverse at same constant speed to construct a VANET that make the packets movable without disconnection. But in reality, not every vehicle is equip with such device and driving at same constant speed. This make it difficult for the device to send RQ and RR packets to the destination node as it would have huge delays if the connection is disconnected.</p>

2.5 Summary

From the selected 4 related studies articles, there are weaknesses in the design that could have being improvised. Therefore, in this project, we can provide the general public an affordable solution to traffic jam issues without sacrificing the user trust on privacy since this project GPSNAV doesn't collect any user location information. However, there might have times when the user is outside of the coverage area, the system can use previously learn pattern to give user a better route planning based on the past and the present of the same time of the day.

Because the GPSNAV is affordable and everyone has the possibility of owning one, upon a term of service agreement with the users, the GPSNAV can gather aggregate speed patterns of a particular road segment of a specific time. This provides a good database for future references and estimation. And since there are so many users, it is possible to widen the coverage by using VANET concept of having more aggregate cells.

CHAPTER 3

3.1 Methodology

Software Development Life Cycle (SDLC) in general has 5 phases which are planning, analysis, design, implementation and maintenance. As defining a realistic goal for a project as the project runs, documenting milestone of the software being develop is crucial for checking the progress of the development against the define goal. By using a SDLC to keep things in checked, it makes things more manageable as complexity of the project increases while producing quality software within the allocated time frame and project budget. These methodologies that follow the standard SDLC are referred to as heavy methodologies or plan driven.

However, in recent years, newer methodologies are unlike the classical ones that focus on long development cycle but rather, it focuses on short iterations which makes the newer methodologies agile and quick to adapt to changes. These newer methodologies are referred as light methodologies. Hence, in this section, this project discusses one type of methodologies from both heavy methodologies and light methodologies:

- 1) Waterfall Model
- 2) Incremental and Iterative Development (IID)
- 3) Extreme Programming (XP)

3.1.1 Waterfall Model

Waterfall model, as depicted in Figure 4, is the oldest among all methodology, is a heavy methodology as it only proceeds in a one-way direction or downward fashion (which is known here as waterfall). It consists of 5 phases which are requirements, design, implementation, verification and maintenance.

The advantage of waterfall model is that it requires the completion of a single phase before proceeding to the next phase. Therefore, if there is any fault encountered during one of the initial phases, it can be corrected immediately. The disadvantage of this model is huge amount of time is laid waste as each phases required the completion of the preceding phases in order to work. Hence, if any delays from the preceding phases will have a high impact on the overall timeline of the project.

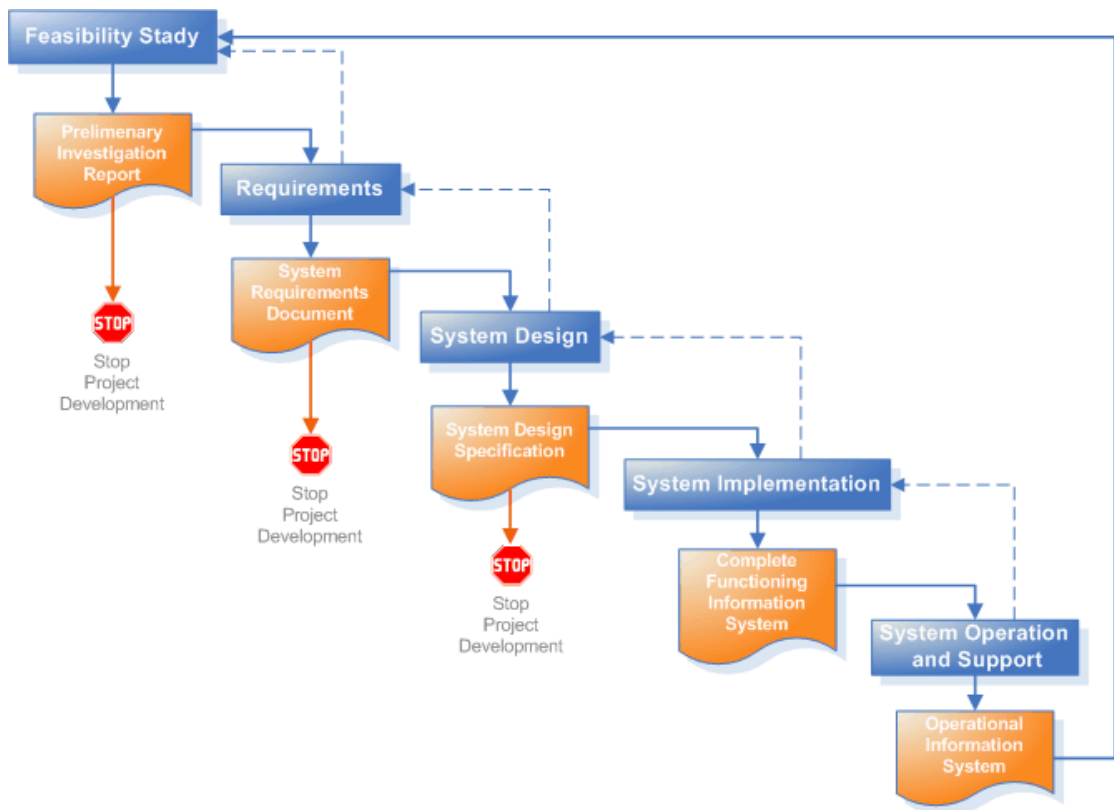


Figure 4 - Waterfall Model

3.1.2 Incremental and Iterative Development

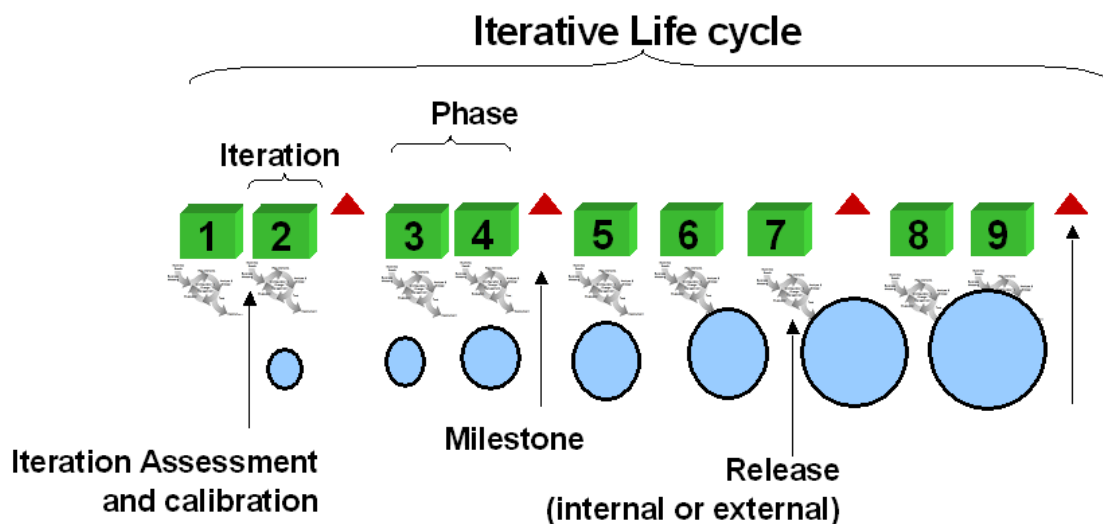


Figure 5 - Iterative and Incremental Development Life Cycle

Figure 4 depicts a high-level view of Iterative and incremental development lifecycle, IID belongs to heavy methodologies. The cycle starts off with repeat assessment and calibration of the project and initial start of building the system. As phases move

forward, additional function will be implement, the cycle repeats itself for designing, building and testing till the desired results is achieve.

The advantage of this methodology is that at each iteration phases, functionalities of the system are tested before increasing functionality as it simplifies the debugging process since the issue would be isolated from the phases before. The disadvantage of this approach is that it would be time consuming and lengthy before the actual product is valid for the client.

3.1.3 Extreme Programming

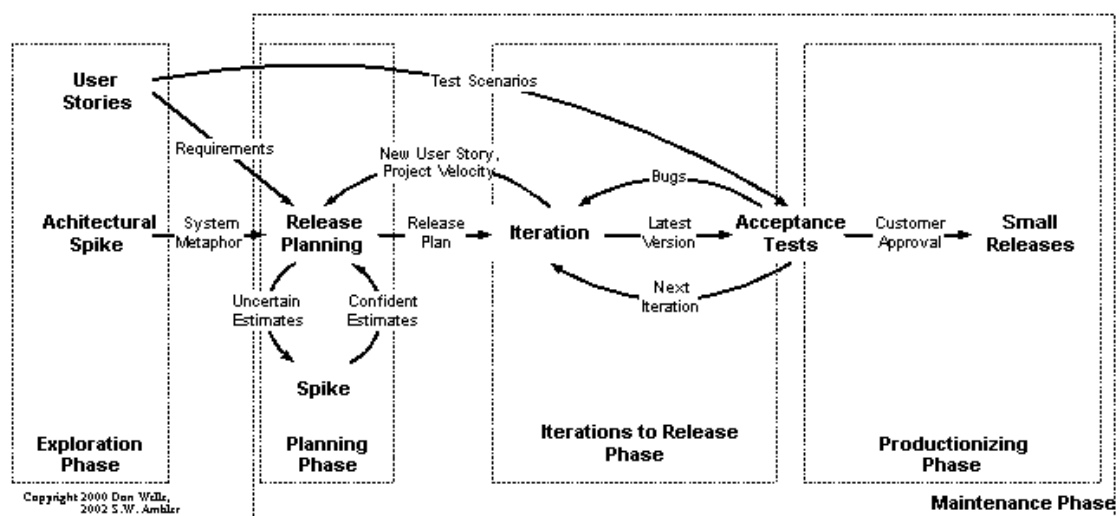


Figure 6 - XP Lifecycle

However, for a light methodologies, extreme programming as depicts in Figure 5, is similar to IID but with a major difference. In XP, a develop product is release to customer or internal team in between each iteration while in IID, the develop product is release to customer or internal team after certain milestone of the project. XP practices planning, pair programming, test driven development, refactoring, continuous integration and on-site customers. And for XP to be categorise in light methodology is the fact that it takes few weeks for an iteration and flexible towards change of requirement rather than months and fixed requirements in heavy methodology.

The advantage of this model is it focuses on small, incremental release to decrease the risk on the project while the disadvantage is it is geared towards toward a single project for developing and maintaining by a single team.

3.1.4 Selecting a Development Model

After discussing the 2 methodology above, this project adopts extreme programming methodology. In this project, user groups as defined in XP model, includes the author for programming and tracking the progress of this project, supervisor or moderator to monitor the development process and as a mentor to the author and they also act as customer to create and prioritise stories that is required to be implement in this project. The selection of this method is reason by the constraint time and resources that restrict many possible outcome of this project, hence, to keep things simple and straight forward, XP is chosen as out methodology.

In the exploration phase, customer stories of how the GPSNAV should work as depicts in Figure 6 during the initial requirements modelling. In Chapter 1, Figure 1-3 shows a sketch of the precise requirement of the customer story as well as given idea for the planning phase.

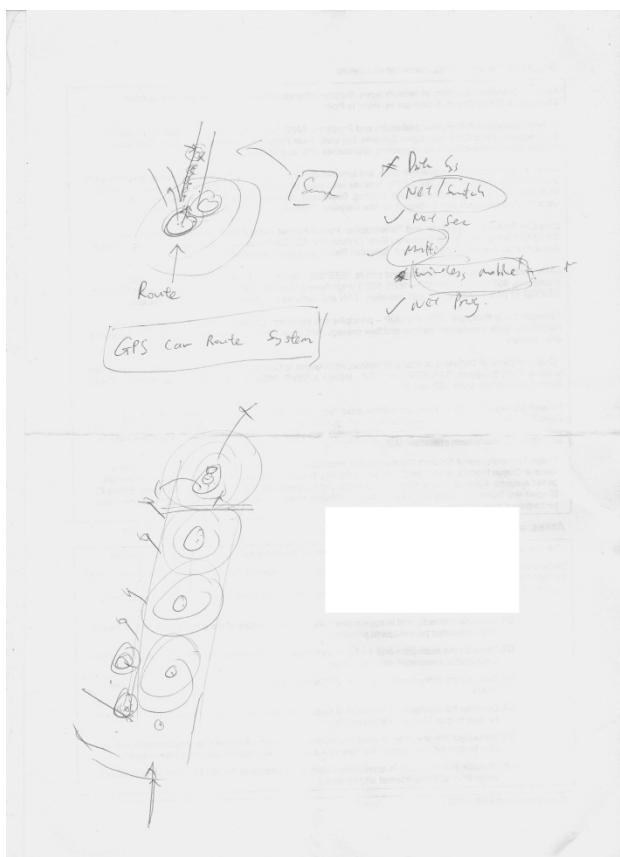


Figure 7 - Stories from Customer

In the planning phase, ideas from Figure 1-3 from Chapter 1 indicate the rough ideas of how should the project proceed from here onwards. In the iterations to release

phase; modelling, programming, testing and integration is where this project major development efforts by accurately identify the task so that estimating customer story more accurately. Iteration happens as more idea for improvement is feasible for this project and testing the product before it is shift to the next phase.

In the productionizing phase, this is where the final result is release to the customer. System documentation, operations documentation, support documentation and user documentation will be written and incorporate with Project 2 report. A demonstration of the system will be shown to customer and to acquire customer approval before final deployment of the GPSNAV.

3.2 Requirement Specifications

As previously mentioned in Chapter 1, under the project scope and objectives, this project is to build a GPS navigation system that is capable of rerouting when the on-going route is not possible or not feasible to proceeds. Hence, in this section, the paper will discuss the system requirement that is necessary to write the Android application and field testing the GPSNAV, the project definition in system performance and verification plan for the system.

3.2.1 System Requirements

The system requirements is break into 2 parts where the first part is the requirement for writing the Android program while the latter part is the requirement for field testing the GPSNAV. Hence, to begin with the first part, the following list is the requirement to write the Android program:

- Eclipse
- Android Development Tool (an Eclipse plugin)
- Windows or Linux operating system (both are compatible with Eclipse)

Eclipse is an open source IDE that comprise of multiple programming languages with extensible plugins, where the ADT is an extension for Eclipse. The ADT uses Android SDK which is a prerequisite for any Android application writing, it also serves as a management for the Android API and version of Android that can be developed.

The latter part of this section, compose of the hardware required for field test the GPSNAV after successful developing it. Therefore, the following list is the hardware requirement for running a field test:

- CEEDTec Board WW721-MAIN rev4C
- 802.11g Wireless Mini PCI module model: WMIA-165G/E
- Motorola Milestone (Android v2.3.7)
- Moving vehicle (Bicycle, Motorcycle or Car)

The CEEDTec board is a custom router board that is easily customisable with a GSM module and a Wi-Fi module. Firmware installed is OpenWrt which is a bare firmware with lots of customisable options (will be discuss under Technology section). The 802.11g Wireless Mini PCI module model: WMIA-165G/E is the Wi-Fi module for adding on the CEEDTec board, this is to make the CEEDTec board a wireless router that is possible to make it as an access point and relay point. The Motorola Milestone will be installed with the GPSNAV for testing it in real time while the author will be using any type of moving vehicle.

3.2.2 System Performance Definition

The system performance definition as defined for this project is “How the GPS navigation system can react quickly enough for the change of digital map”. The base idea for this question is that how would the GPSNAV be quick to process from the moment the Wi-Fi message is receive to that moment the GPSNAV prompt the user. To evaluate the swiftness, the coverage of the wireless router in this project presents a limiting factor as to how far can the wireless router cover in an assume open space area. The next evaluation base on the same question is the responsiveness of the application towards the change of digital map. That is being said, the responsiveness of the application widely depends on the processor speed of the smartphone and the application algorithm in determining the new path.

3.2.3 System Verification Plan

The verification exercise of this project is to verify the system performance that was previously defined in 3.2.2. A series of test is carried out to determine the performance of the GPSNAV. The first test, simulate an Android smartphone via AVD and input Wi-Fi message to the virtual device. Timing of the programing is

recorded in a log file within the program for verification purposes. The second test and last test to cooperate together is to determine the effective coverage area for the wireless router and the speed of rerouting user from Wi-Fi message receive to marking of digital map.

3.3 Implementation Issues

Challenges and issues faces by this project can be difficult to resolve and it would be hard to implement. Of such issues, the first issue would arise is the digital map issue which require certain knowledge of creating digital map for the Android application to comprehend the information on the digital. The next issue is writing the Android application itself as knowledge of writing it is important for how good an application is written. Due to the fact that the author had not learn any of the Android programming language although slight similarities to Java programming language as no subject of this matter was offered by the university.

Apart from the GPSNAV application issues, issue with the wireless router would present a greater threat to the successfulness of the project implementation. Assuming the GPSNAV is capable of reading Wi-Fi messages and re-route the user, but without any Wi-Fi coverage and the GPSNAV didn't receive any Wi-Fi message, than rerouting would not be possible given that the current route is not feasible.

3.4 Timeline

The following figures are Gantt charts that are associated for Project I in Year 2 Semester 3 only. As for Project II in Year 3 Semester 1, Figure 10 shows a Project II timeline summary while the Gantt chart is included in Appendix. In addition to the timeline and Gantt chart for Project II, the Work Breakdown Structure (WBS) as shown in Table is covers only for Project II since it's the planning phase, iteration to release phase and production phase which has details information of the whole timeline.

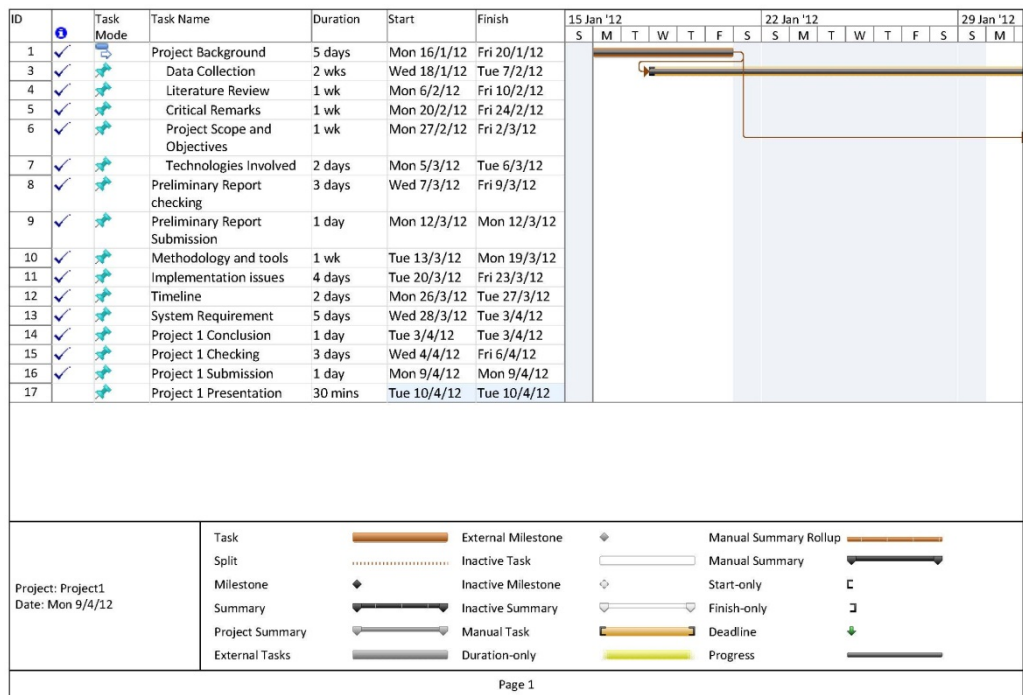


Figure 8 - Project I Gantt Chart Part1

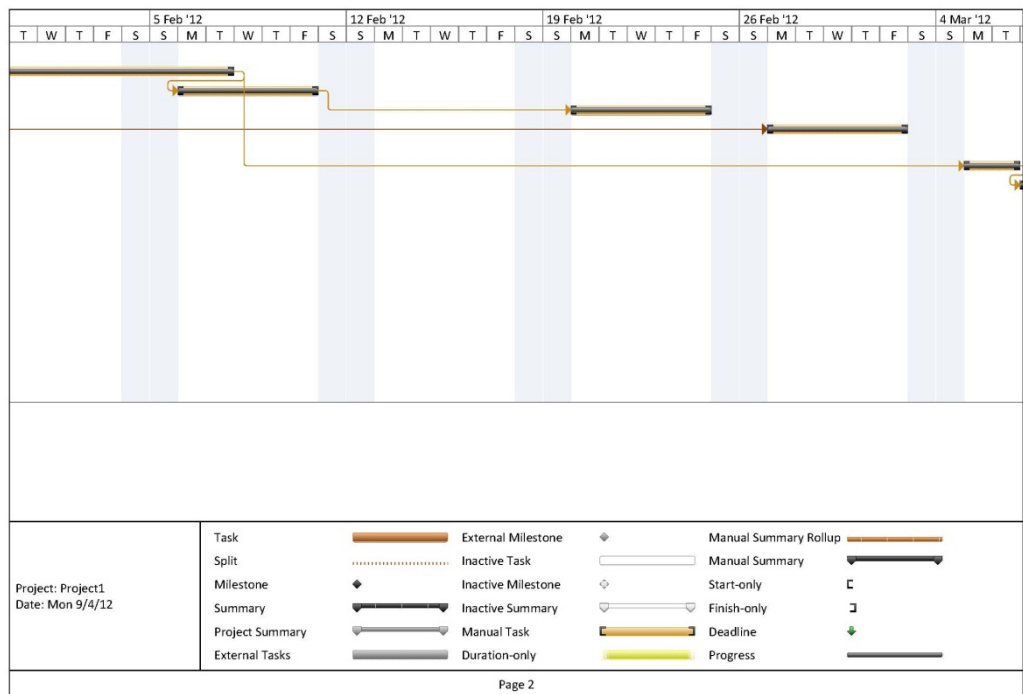


Figure 9 - Project I Gantt Chart Part2

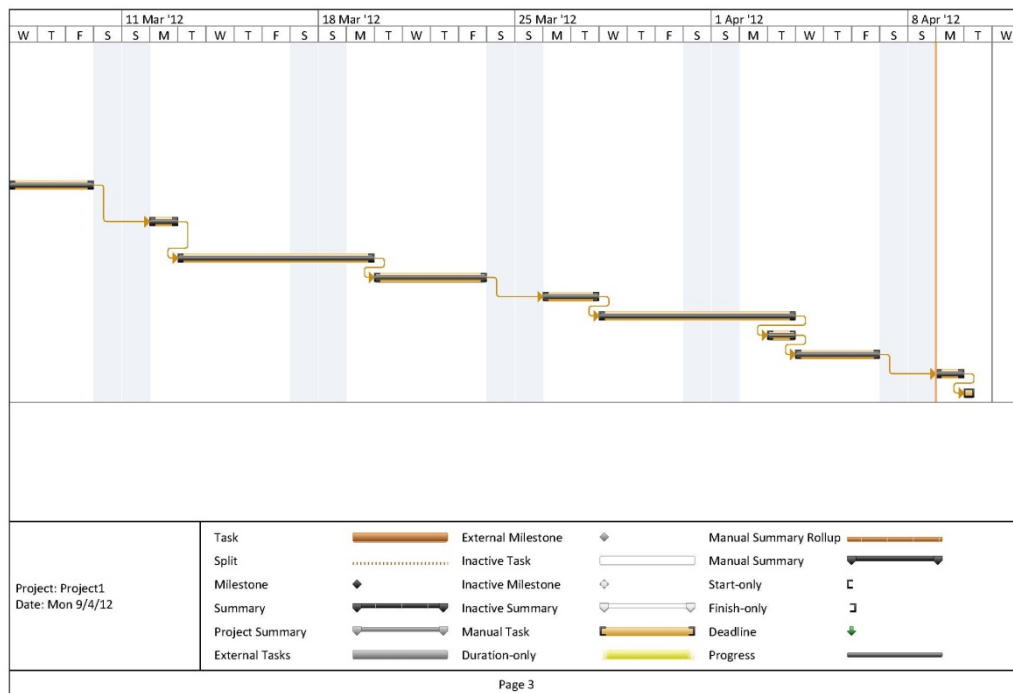


Figure 10 - Project I Gantt Chart Part3

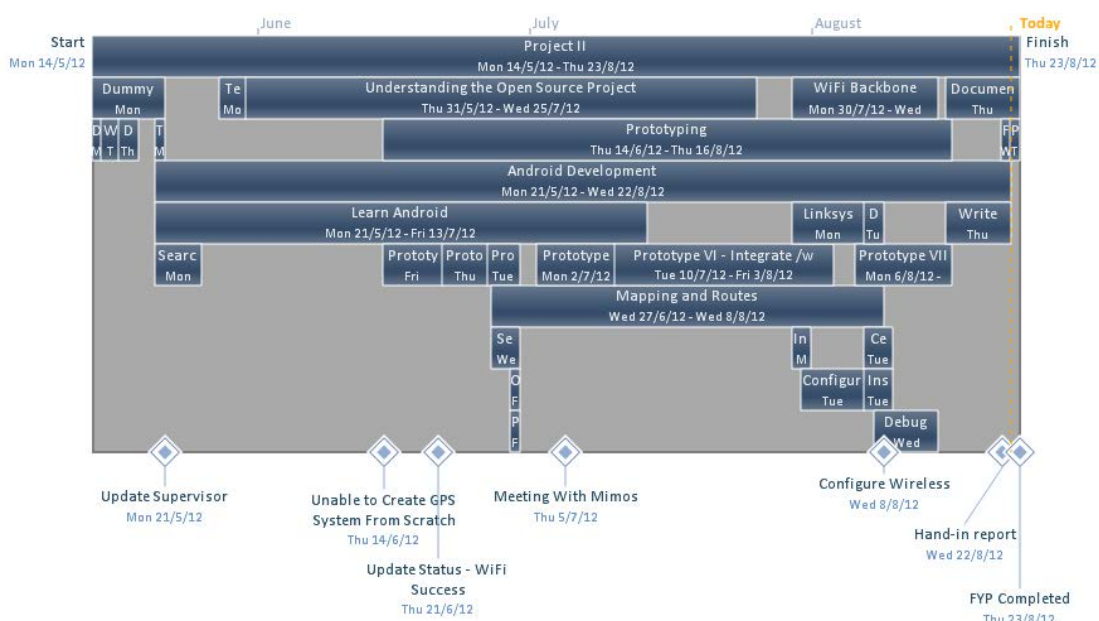


Figure 11- Project II Timeline Overview

Table 2 - Project II Work Breakdown Structure (WBS)

WBS	Task Name	Duration	Start	Finish
	Project II	74 days	Mon 14/5/12	Thu 23/8/12

1	Dummy Server	6 days	Mon 14/5/12	Mon 21/5/12
1.1	Draw Flow Chart	1 day	Mon 14/5/12	Mon 14/5/12
1.2	Write the Code	2 days	Tue 15/5/12	Wed 16/5/12
1.3	Debugging	2 days	Thu 17/5/12	Fri 18/5/12
1.4	Test on LAN	1 day	Mon 21/5/12	Mon 21/5/12
1.5	Update Supervisor	0 days	Mon 21/5/12	Mon 21/5/12
2	Android Development	68 days	Mon 21/5/12	Wed 22/8/12
2.3	Learn Android	8 wks	Mon 21/5/12	Fri 13/7/12
2.1	Search Open Source GPS Navigation System	1 wk	Mon 21/5/12	Fri 25/5/12
2.2	Test Compile Source Codes	3 days	Mon 28/5/12	Wed 30/5/12
2.4	Understanding the Open Source Project	8 wks	Thu 31/5/12	Wed 25/7/12
2.18	Meeting With Mimos	0 days	Thu 5/7/12	Thu 5/7/12
2.17	Mapping and Routes	31 days	Wed 27/6/12	Wed 8/8/12
2.17.4	Search for Mapping Software	3 days	Wed 27/6/12	Fri 29/6/12
2.17.1	Obtaining Kampar Map	1 day	Fri 29/6/12	Fri 29/6/12
2.17.2	Plotting Waypoints	1 day	Fri 29/6/12	Fri 29/6/12

2.17.3	Draw Routes	2 days	Tue 7/8/12	Wed 8/8/12
2.14	Prototyping	44.5 days	Thu 14/6/12	Thu 16/8/12
2.14.1	Prototype I - GPS System	1 wk	Mon 16/7/12	Fri 20/7/12
2.14.2	Unable to Create GPS System From Scratch	0 days	Thu 14/6/12	Thu 14/6/12
2.14.3	Prototype II - UDP Socket Listener	4.5 days	Fri 15/6/12	Thu 21/6/12
2.14.4	Update Status - WiFi Success	0 days	Thu 21/6/12	Thu 21/6/12
2.14.5	Prototype III - Scan Access Point	3 days	Thu 21/6/12	Tue 26/6/12
2.14.6	Prototype IV - Background Service Test	3.5 days	Tue 26/6/12	Fri 29/6/12
2.14.7	Prototype V - Background Service UDP + Prototype I	6.5 days	Mon 2/7/12	Tue 10/7/12
2.14.8	Prototype VI - Integrate /w Open Source Project /w Buttons	18 days	Tue 10/7/12	Fri 3/8/12
2.14.11	Integration Button Successful	0 days	Fri 3/8/12	Fri 3/8/12
2.14.9	Prototype VII - Integrate /w Open Source Project /w binding Service	1.7 wks	Mon 6/8/12	Thu 16/8/12
2.14.10	Integration Successful	0 days	Thu 16/8/12	Thu 16/8/12
2.15	Field Test	1 day	Wed 22/8/12	Wed 22/8/12
3	WiFi Backbone	12 days	Mon 30/7/12	Wed 15/8/12
3.2	Linksys Routers	6 days	Mon 30/7/12	Mon 6/8/12

3.2.1	Install OpenWrt	2 days	Mon 30/7/12	Tue 31/7/12
3.2.3	Configure Wireless	1 wk	Tue 31/7/12	Mon 6/8/12
3.3	CeedTec Routers	3 days	Tue 7/8/12	Thu 9/8/12
3.3.1	Install Packages	3 days	Tue 7/8/12	Thu 9/8/12
3.3.2	Configure Wireless	0 days	Wed 8/8/12	Wed 8/8/12
3.5	Debug and Test Wireless Topology	5 days	Wed 8/8/12	Tue 14/8/12
3.6	Update on Wireless Topology	0 days	Wed 15/8/12	Wed 15/8/12
5	Documentation	6 days	Thu 16/8/12	Thu 23/8/12
5.1	Write Report	5 days	Thu 16/8/12	Wed 22/8/12
5.2	Hand-in report	0 days	Wed 22/8/12	Wed 22/8/12
5.3	Presentation	1 day	Thu 23/8/12	Thu 23/8/12
5.4	FYP Completed	0 days	Thu 23/8/12	Thu 23/8/12

3.5 Technology Involve

In the current era, technology has given a very high impact to nearly every industry in the world, such as automobile industry, mobile phone industry, business and finance industry, agriculture industry and many more to be listed. Of course, without humans' creativity mind, innovation and curiosity attitude, there would be neither technology advancement nor any breakthrough in the human history that could bring so much benefit to the humanity.

Hence, in order to complete the project, which I believe the project would bring benefit to humanity, the following technologies are used or reference for their ideas and implementations.

Android

Android Incorporate was founded in Palo Alto, California, United States by Andy Rubin, Rich Miner, Nick Sears and Chris White in October 2003 and was officially acquired by Google in August 2005 [11]. The Android Beta Software Development Kit (SDK) was released in November 2007 [11] and the Android era officially began on 22nd October 2008 shortly after nearly a year when the T-Mobile G1 was launched in the United States [12] pre-installed with the first version of Android, the Android 1.0. Subsequent updates are followed after the initial release with more functionality and utilising the newer technologies available at the time of release. Till date, Android has reach version 4.0.3 (codename: Ice Cream Sandwich).

Android uses Java programming language as the fundamental for writing its' application. Hence, the Android SDK tools is used to compile the code along with any data and resource files into an Android package (.apk) that is use as an installer for the Android based mobile devices [12].

Motorola Milestone

In November 2009, after a year of the first G1's premiere Android 1.0, the Android 2.0 (codename: Éclair) was pre-installed in Motorola Milestone as the first debut for this Android version. The Motorola Milestone used in this project is a rooted device with a custom rom (CyanogenMod 7) installed to enable overclocking from 600 MHz to 1000 MHz and change the Android default VMHeap value from 24 to 40. The device hardware specification is listed in the following table:

Table 3 - Motorola Milestone Specification

General	2G Network	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 900 / 2100
		UMTS 850 / 1900 – (US only)
Body	Dimensions	115.8 x 60 x 13.7 mm
	Weight	165g

	Keyboard	QWERTY	
Display	Type	TFT capacitive touchscreen	
	Size	480 x 854 pixels, 3.7 inches	
	Multitouch	Supported	
	Protection	Corning Gorilla Glass	
Sound	Alert types	Vibration; MP3, WAV ringtones	
	Loudspeaker	1 x Stereo speakers	
	3.5mm jack	1 x port	
Memory	Card slot	micro SD	
	Internal	133MB storage, 256MB RAM	
Data	GPRS	Class 12 (4+1/3+2/2+3/1+4 slots), 32 - 48 kbps	
	EDGE	Class 12	
	3G	HSDPA, 10.2Mbps; HSUPA, 5.75Mbps	
	WLAN	Wi-Fi 802.11 a/b/g	
	Bluetooth	Supported	
	USB	1 x micro USB v2.0 port	
Camera	Primary	5MP, autofocus, dual-LED flash	
	Features	Geo-tagging	
	Video	D1 (720-480 pixels) @ 24fps	
Features	OS	Android OS, v2.3.7 (CyanogenMod7). Rooted	
	CPU	600 MHz Cortex-A8 (Max at 1000 MHz)	
	GPU	PowerVR SGX530	
	Sensors	Accelerometer, proximity, compass	
	Messaging	SMS, MMS, Email, IM, Push Email	
	Browser	HTML, Adobe Flash	
	GPS	Yes, with A-GPS support, Motonav software	
	Java	Yes, via Java MIDP emulator	
	Battery	Standard battery	Li-Ion 1400 mAh
		Standby/Talk time	~350 hours / ~6 hours 30 minutes

Wireless LAN 802.11 Wi-Fi

The '802' is a general IEEE designation for network standards and the '11' family of standards governs the wireless local area networking. The IEEE 802.11 wireless LAN or Wi-Fi denotes a set of standards developed by working group 11 of the IEEE LAN/MAN Standards Committee (IEEE 802) [15].

Currently Wi-Fi includes six over-the-air modulation techniques that use the same Layer 2 protocol. The most popular techniques are those defined by a, b and g amendments to the original standard. Wireless security was originally included and was further enhanced by the 802.11i amendment. Other standards of the family (c-f, h-j, n) are service enhancement and extensions or corrections to the previous specifications. 802.11b was the first to be accepted wireless networking standard in 1999 and followed by the 802.11a, 802.11g and 802.11n. The following table by [13], shows a summarised standard coverage range for all IEEE 802.11:

Table 4 - Summarise of Current Working Protocol

Protocol	Release Date	Operating Frequency	Data Rate (Typical)	Data Rate (Max)	Range (Indoor)
Legacy	1997	2.4 – 2.5 GHz	1 Mbit/s	2 Mbit/s	Unknown
802.11a	1999	5.15 – 5.35 / 5.47 – 5.725 / 5.725 – 5.875 GHz	25 Mbit/s	54 Mbit/s	~ 30 meters (~ 100 feet)
802.11b	1999	2.4 – 2.5 GHz	6.5 Mbit/s	11 Mbit/s	~ 50 meters (~ 150 feet)
802.11g	2003	2.4 – 2.5 GHz	11 Mbit/s	54 Mbit/s	~ 30 meters (~ 100 feet)
802.11n	2006 (draft)	2.4 GHz / 5 GHz bands	200 Mbit/s	540 Mbit/s	~ 50 meters (~ 160 feet)

Global Positioning System (GPS)

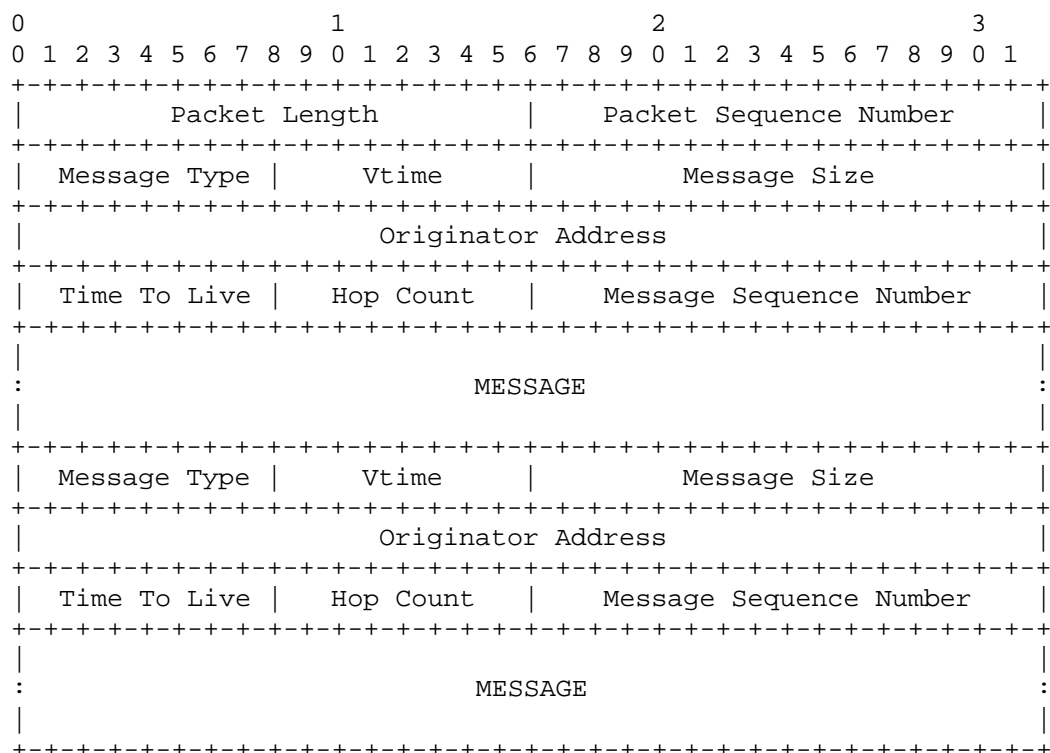
GPS is the first satellite navigation system started by the U.S. Government's Department of Defence whom officially name it as NAVSTAR system [2]. A GPS system consist of a satellite and a ground station; currently there is a total of 24 medium earth orbit satellites that transmit precise microwave signals to-and-fro

between the satellite and its respective ground station to calculate geographical positions. A GPS receiver only receives the precise microwave signals from the satellite to determine the receiver location, speed, direction and time.

Optimised Link State Routing Protocol (OLSR)

OLSR is an IP routing protocol specifically optimized for MANET. It inherits the stability of a link state algorithm and has the advantage of having routes available immediately when needed due to its proactively in sensing its neighbour. OLSR uses only selected nodes, called multipoint relays (MPR), to retransmit control messages as a way to minimize the overhead from flooding control traffic. This technique significantly reduces the number of flood message retransmission to all nodes in the network as it only require partial link state to be flooded in order to learn the shortest path routes.

The designed for OLSR to work is entirely based on distributed system and does not depend on any central entity. It also does not require reliable of transmitting the control message as each node sends a control message periodically, therefore, can sustain a reasonable loss of some control message (especially in radio networks suffer losses due to collisions or other transmission problems). The basic layout of any packet in OLSR is as follows (detail of each message header can be refers in [14]):



: (etc.) :

OpenWrt

OpenWrt is a highly extensible GNU/Linux distribution for embedded devices [16]. It aims not to create a single-static firmware but to provide a fully writable filesystem with package management that is easily to customisable for users. OpenWrt is free and open source; therefore there is no cost of deploying it or a relatively cheaper cost with the embedded wireless network devices to be configured.

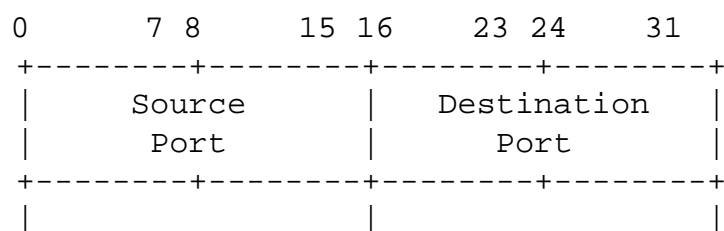
The idea of using this firmware is to create a usable wireless bus network topology with message relaying capabilities or to join up multiple wireless router of small coverage into a big coverage area.

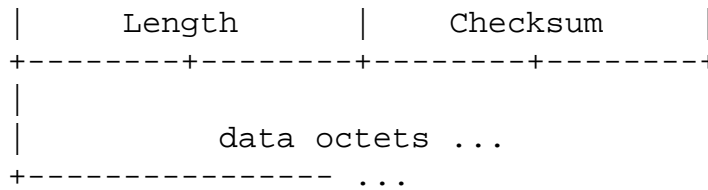
GPS Navigation System (GPSNAV)

The word “Navigation” is defined such that it is a process or activity to accurately determine one’s location with planning of route to reach the destination. And in the current era with the availability of GPS, navigation was never difficult again. A GPSNAV device by default contain a pre-loaded digital map and the internal system will perform some calculation to plan and estimate the time required to reach the destination and the best path is chosen by shortest distance available (commonly uses Dijkstra’s Algorithm but there are many other algorithms available).

User Datagram Protocol (UDP)

UDP is a connectionless transport layer protocol in the OSI model, which provides a simple and unreliable (reliability can be improve in the upper layer implementation) message service for transaction-oriented services, and delivery and duplicate protection are not guaranteed. It is defined by IETF RFC 768 and the following is the header format [17].





User Datagram Header Format

Eclipse

Eclipse is a multi-programming language platform that is open source integrated development environment (IDE) and easily to be extend with custom plugins. Often, programming language used in Eclipse is Java, C/C++, Perl, PHP, Python, Android, and other languages. Although with much of extended support for different programming language, it was initially meant for Java developers only.

3.6 Budget Plan

The budget for this project is shown in the table below. From Table 3, it shows the cost for programming the Android application and the UDP program as initial estimation while from Table 4 shows the reduction of cost from the initial cost and the grand total for this project is RM0. Though, developing this project could be free as in hardware and software requirement, but these figures are just estimation, as the project proceeds, cost might have some slight changes or no changes.

Table 5 - Initial Budget Plan

Item Description	Cost/Item (RM)	Quantity	Total (RM)
CEEDTec Board WW721- MAIN rev4C + 802.11g Wireless Mini PCI module model: WMIA-165G/E	200	4	800
Motorola Milestone	1200	1	1200
Computer	2000	1	2000
Subtotal			4000

Table 6 - Amended Planned Cost

Item Description	Cost/Item (RM)	Quantity	Total (RM)
CEEDTec Board WW721- MAIN rev4C + 802.11g Wireless Mini PCI module model: WMIA-165G/E	200	4	800
Linksys WRT54G	75	2	150
Motorola Milestone	1200	1	1200
Computer	2000	1	2000
Subtotal			4000

Table 7 - Final Cost

Item Description	Cost/Item (RM)	Quantity	Total (RM)
CEEDTec Board WW721- MAIN rev4C + 802.11g Wireless Mini PCI module model: WMIA-165G/E	(200)	4	(800)
Linksys WRT54G	75	2	150
Motorola Milestone	(1200)	1	(1200)
Computer	(2000)	1	(2000)
Subtotal			(4000)
Grand Total			150

3.7 Summary

In a short summary, 2 types of methodology were discussed in this chapter namely Incremental and Iterative Development methodology and Extreme Programming methodology. Incremental and Iterative Development is a type of heavy methodology while Extreme Programming is a type of light methodology and each has their own advantages and disadvantages. Heavy methodology focuses on complete overview of the project for planning, designing, implementation and maintenance, a fixed plan

while light methodology is more towards customer oriented and that it is flexible to change the initial plan. For this project, Extreme Programming was chosen to precede things smoothly and quickly as constraint of time and resources are present.

The requirement specification for this project has only 4 hardware components and 3 software component. The hardware components consist of an Android smartphone, moving vehicle, router bare board and a wireless module for router add-on while the software components consist of Eclipse, Android Development Tool and Windows or Linux base operating system. While defining the system performance, the benchmark will be the speed of the application re-route user and the Wi-Fi coverage area in which also includes the implementation complications introduce by the project.

CHAPTER 4

4.1 Overview of System Design

The general idea here is to get the GPSNAV to reroute user from current navigating route to an alternative available route upon receiving a Wi-Fi message from a traffic monitoring service broadcaster. The GPSNAV will be working on Android platform as GPS and Wi-Fi module is a common functionality of most mobile devices which is the main motivation of creating this GPSNAV. Although the general idea is to respond upon receiving a Wi-Fi message, the Wi-Fi network is of important and crucial for this project to be successful. Hence, a general or simplistic Wi-Fi network is required to be establish where each Wi-Fi node or beacon are able to serve as a bridge between them and also an access point for the GPSNAV to be connected with.

In the next 3 sections of this chapter, it is divided into 4.2 Hardware Design, 4.3 Software Design and lastly 4.4 Summary and Concluding Remark of Chapter 4.

4.2 Hardware Design

In this section, the wireless routers consist of CeedTec MPC8241-M3-EP (with an Atheros based PCI module slotted on it) and Linksys WRT54G (with a variant of WRT54Gv2 and WRT54GL, both share several similarities in design). A simple block diagram detailing the wireless routers internal architecture is design by their respective manufacturer (CeedTec / Linksys). Both type of wireless router share the same platform, OpenWrt (albeit with different revision), this is to simplify the network configuration across any branded routers (assume to have same platform installed).

4.2.1 CeedTec MPC8241-M3-EP

This CeedTec MPC8241-M3-EP router is installed with OpenWrt Kamikaze 8.09 which is an open source router operating system which make possible for configuring a wireless bridge that will connect to other wireless router and at the same time to act as a Wi-Fi access point which will provide Wi-Fi connectivity for the users. A brief hardware and customised software installed in this router is tabulated in Table 3.

Table 8 - MPC8241-M3-EP Hardware/Software Specification

Model	MPC8241-M3-EP
Processor	Freescale MPC8241 @ 200Mhz
RAM	32MB
Flash	8MB
32-bit 33MHz Mini-PCI Slots	2 slots i. Atheros based Wi-Fi b/g Mini-PCI card ii. None
Mini PCIe Slot	GPRS module card
Standard Interface	i. Ethernet RJ-45 x1 ii. USB 2.0 Host Interface iii. DC 12V Power Set iv. Reset Switch
Operating System	OpenWrt Kamikaze 8.09 (Custom Patch)
Software Installed	i. Olsrd ii. Firewall iii. LuCI iv. LuCI – Freifunk mod v. Freifunk-p2pblock vi. Freifunk-watchdog

Figure 7 is a simplified block diagram just for the understanding of how it works. Basically, just like a normal desktop computer, the operating system act as the middle man for interacting between the user and the hardware. Since this router is Linux based, installing of customised software is not really any big issue (except limited by on-board flash memory). Therefore, the routers are installed with LuCI, a web user interface for configuring the router; Optimized Link State Routing daemon, a type of

layer 3 routing protocol for deploying a mesh network; Freifunk related mod are customised for ease of configuring a mesh network with the use of web user interface. Though, keep in mind that wireless chipset is dependent on which chipset based is slotted into the Mini-PCI slots which will have different wireless configuration sets based on different Wi-Fi chipset as it is supported by different drivers. In this project, the author uses Atheros based Wi-Fi chipset slotted on the Mini-PCI slot 1, hence, the wireless configuration will be based on Madwifi options.

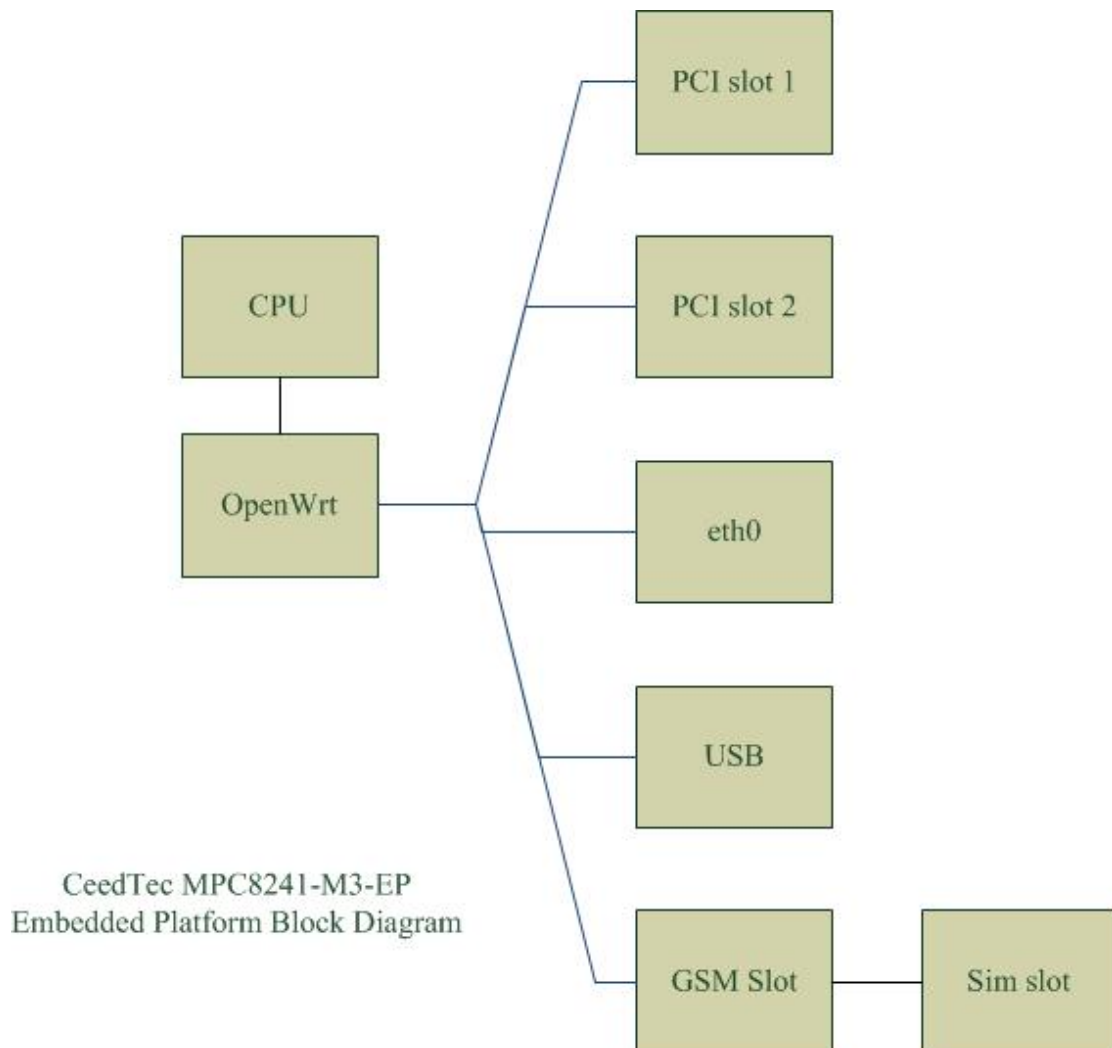


Figure 12 - CeedTec MPC8241-M3-EP Block Diagram

4.2.2 Linksys WRT54G (WRT54Gv2 / WRT54GL)

On the other hand, WRT54G is yet another wireless router with a similar specification as compare to CeedTec MPC8241-M3-EP with both processors running at 200MHz and uses Linux as based operating system with an exception that WRT54G supports 4 port switch but does not support a USB 2.0 Host Interface, 1 Mini-PCIe slot and 2

Mini-PCI slots. Hence, even the software installed is similar to CeedTec MPC8241-M3-EP with the exception of the OpenWrt version is Backfire instead of Kamikaze. The hardware specification and a simple block diagram for Linksys WRT54G is shown in Table 4 and Figure 8. Albeit with the version differences, the configuration for both hardware are identical except for the wireless configuration which in this router, it uses MAC80211 driver to support its Broadcom based Wi-Fi chipset.

Table 9 - Linksys WRT54Gv2 / WRT54GL Hardware/Software Specification

Model	WRT54Gv2 WRT54GL
Processor	Broadcom BCM4712 @ 200MHz Broadcom BCM5352 @ 200MHz
RAM	16MB 16MB
Flash	4MB 4MB
Standard Interface	v. Ethernet RJ-45 x1 vi. USB 2.0 Host Interface vii. DC 12V Power Set viii. Reset Switch
Operating System	OpenWrt Backfire 10.03 (r32751)
Software Installed	vii. Olsrd viii. Firewall ix. LuCI x. LuCI – Freifunk mod xi. Freifunk-p2pblock xii. Freifunk-watchdog

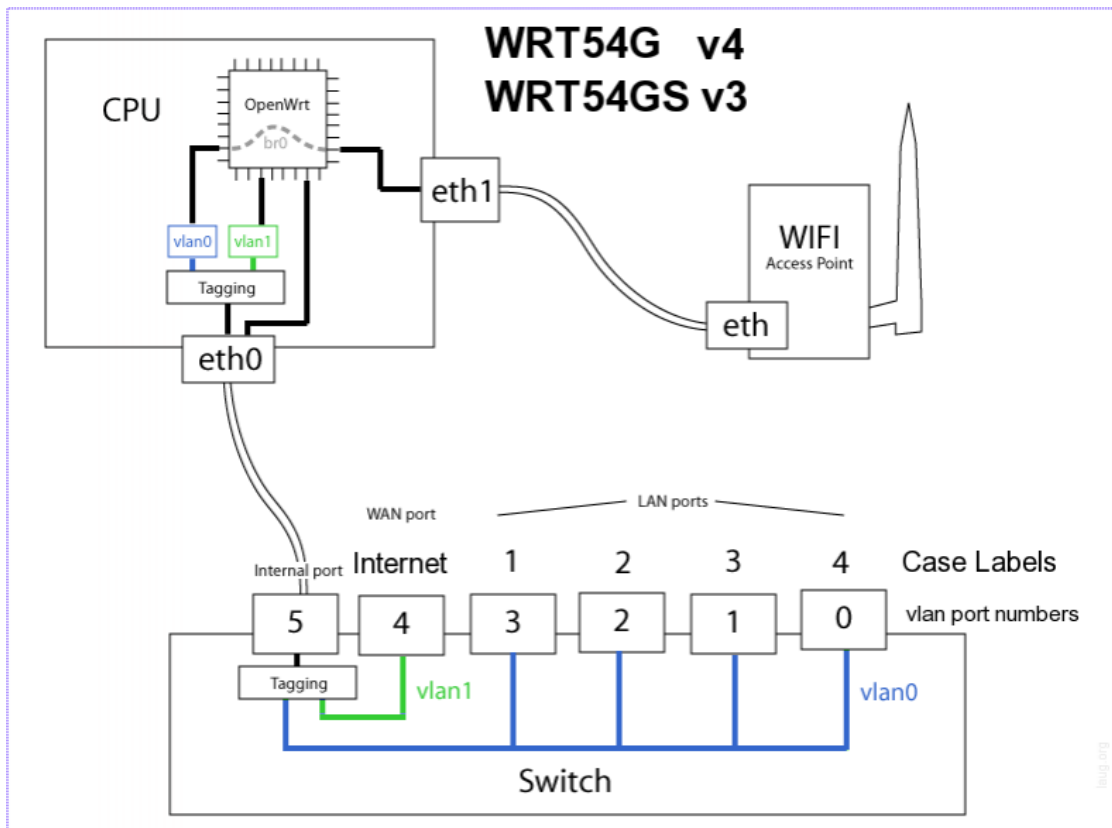


Figure 13 - WRT54G Block Diagram (Generic) extracted from <http://wiki.openwrt.org/toh/linksys/wrt54g>

4.3 Software Design

In this section, flowcharts detailing of how the GPSNAV will monitor the Wi-Fi link for message to be process and how it will respond to the message. And by the respond action to the message is to check the current navigating route list of waypoints, which has the geographic coordinates that is used to compare against the geographic coordinates in the message that was received. This will lead to an alert dialog to be display to prompt user for actions. Let assume the user allowed to be reroute, the GPSNAV will search up the current route list that was loaded by the user for the same final destination waypoint and if there is a route available, the GPSNAV will automatically switch to the next shortest route available base on total distance.

For those that are interested with the Android project folder, one would require permission and access to the source code that should be on a CD which by right, it should be complement with this report. The content of the CD would be sufficient enough for compiling an Android application and guidelines for setting up.

4.3.1 General Concepts

Before anyone would dive further into the flowcharts and trying to understand them, one would require at least the general understanding of how do object-oriented programming works, some basic of markup language and of course the general understanding of how does a Linux kernel works! However, this chapter would not touch anything too in-depth of how does the kernel communicate with the hardware to return the state of each devices or the application framework would handle the message passing. Hence, the basic of programming Android is Java language which is a common and popular object-oriented programming language, and XML markup language for declaring data to be accessible in Java class files.

First of all, in Android development cycle, it is crucial to understand the lifecycle of an activity as depicted in Figure 9 which shows how would a class file that extends activity would look like after execution. For example of an application that has 2 activities, for each time the activity is switch the foreground, the `onResume()` will be invoke. Hence, the important functions that should be continue upon the activity should be run in `onResume()` instead of `onCreate()`. These are just examples, for further understanding, one would be able to access to Android developer center via the internet.

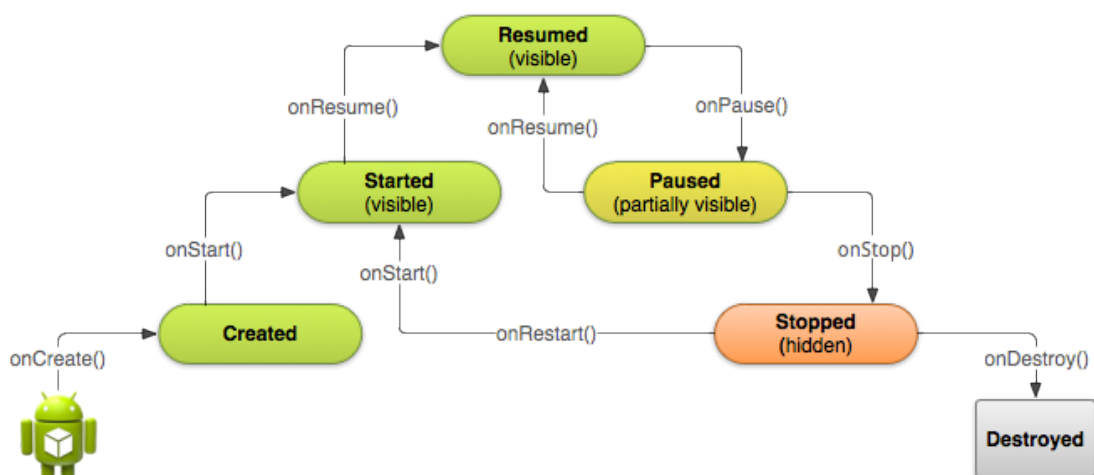


Figure 14- A Simplified Illustration of the Activity Lifecycle (source: <http://developer.android.com/training/basics/activity-lifecycle/starting.html>)

Secondly, in this project, the author uses an open source project obtain from [15] and modified its contents to complement with the objectives of this project. The documentation from the origin website are not sufficient and that understanding an

alien project would take times as the complexity of Androzic is rather high for a beginner in Android like the author of this report. This chapter, however, will include a general flowchart of how does Androzic load the route file and navigate via route.

Thirdly, is that due to Android design best practices, all heavy work and calculations are throw to the background worker thread to work instead of the user interface (UI) thread. This is because if the UI thread is working on heavy stuff and calculation, the front UI will appear to user as not responding and user would most likely to kill the app before it has completed the calculation. Hence, this is the reason why the SmartRoute is a background service.

And lastly, the flowcharts in this chapter are simplified for the sake of the reader of this report to quickly grasp the meaning of each function without the extra attributes of each function.

4.3.2 Androzic

As according to [15], it is an Android navigation client that uses OziExplorer maps (ozf2, ozfx3). Since this Androzic application project is not developed by the report author, a brief flowchart is shown in Figure 10 as the flow is based entirely on the report author understanding of this application and a brief description of how-to usage of Androzic.

In order for Androzic to work, by default, it required an active internet connection (regardless from Wi-Fi network or 2G/3G network) to download a generic world map (world.ozfx3) from the internet and the default online map provider is from OpenStreetMap Mapnik. In order for the user load a route into Androzic, he or she would require to use OziExplorer v3.95.5g (shareware) to plot the waypoints and create the route via it and if one would require to save the map, another tools called GoogleOzi which would be able to grab map image from Google Map with WGS-84 datum embedded within it and save as ozf2 file format (a compatible map for Androzic to load).

However, in this chapter, the author would only touch on the functions that are used for this project. Hence, any other functionality of that are of no use for this project will not be discuss or explained in details here.

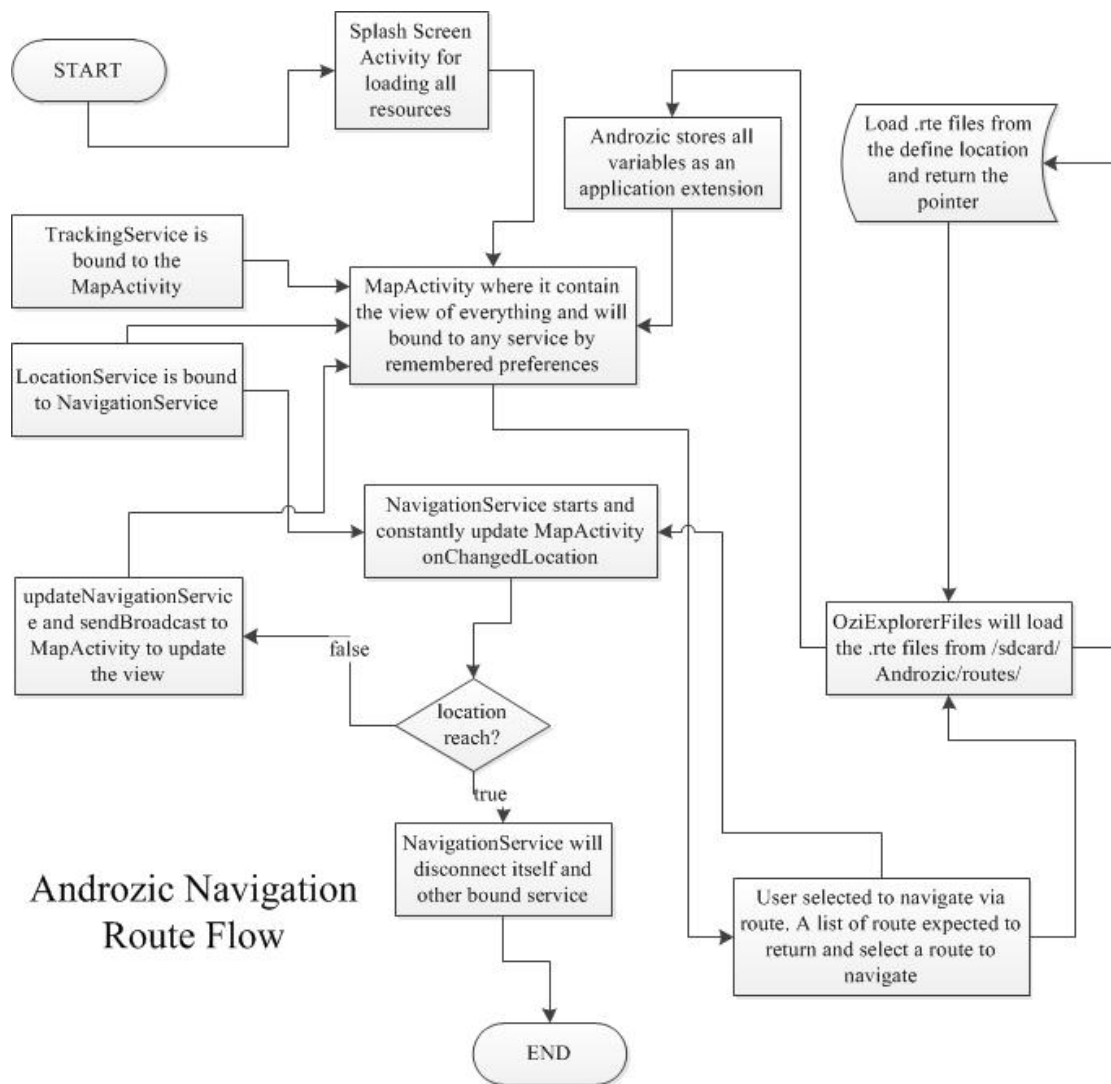


Figure 15 - Androzic Navigation Route Overview

4.3.3 Androzic “NavigationService” Background Service

The Androzic application itself already has a simple navigation built into it. It navigate user either via a predefine route or direct to a predefine waypoint. Navigation via a predefine route, basically, the user loads up a user plotted route (a list of connected waypoints) from Oziexplorer and the NavigationService will navigate user from current location to the second waypoint of the loaded route and continues to navigate between each waypoints until the last waypoint is reach.

While the navigation to a predefine waypoint is basically a direct navigation to the waypoint without considering if the path is even walkable or not walkable. It just calculates the current location and the selected waypoint distance. All these navigation, it will generate an overlay on top of the Androzic MapActivity (an activity

layer to display the map) that connects from the current location to next waypoint to indicate user of current location and distance between it.

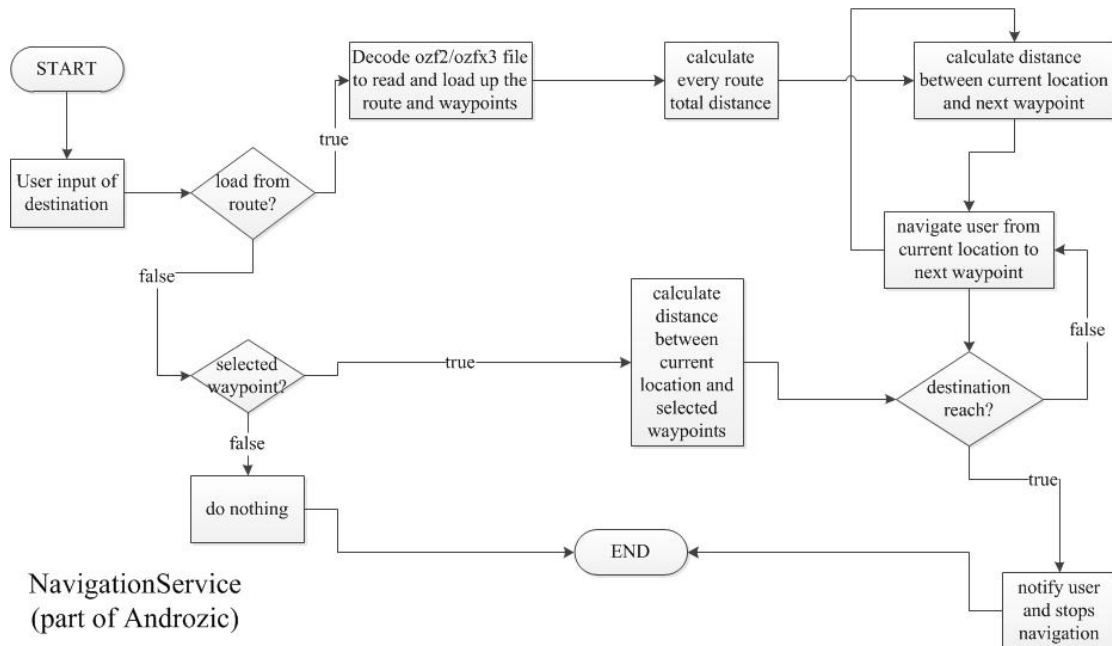


Figure 16 - Androzic NavigationService Flow Chart

4.3.4 Androzic Distance Calculation (Vincenty's Formula)

As further exploration of the Androzic source code, it is found that the developer for Androzic calculates the distance between each waypoint (based on geographic coordinates) uses the Vincenty's formula (as shown in Figure 17) and WGS-84 datum (most accurate and widely used globally-applicable model for the earth ellipsoid) As according to [16], this formula is accurate to within 0.5mm on the ellipsoid being used.

Though, this is a complicated and complex mathematical model which is beyond the author comprehension with the basic geometry and trigonometry knowledge. A rough understanding of this model is that the iteration in within the mathematical model is an attempt to calculate the distance as accurate as possible. However, due to the lack of understanding of this mathematical model, the author uses another simpler mathematical model which will be explained in section "Coverage Area Algorithm".

Vincenty's formula as it is used in the script:

a, b = major & minor semi-axes of the ellipsoid

f = flattening $(a-b)/a$

ϕ_1, ϕ_2 = geodetic latitude

L = difference in longitude

$U_1 = \text{atan}((1-f) \cdot \tan \phi_1)$ (U is 'reduced latitude')

$U_2 = \text{atan}((1-f) \cdot \tan \phi_2)$

$\lambda = L$ (first approximation)

iterate until change in λ is negligible (e.g. $10^{-12} \approx 0.006\text{mm}$) {

$$\sin \sigma = \sqrt{(\cos U_2 \cdot \sin \lambda)^2 + (\cos U_1 \cdot \sin U_2 - \sin U_1 \cdot \cos U_2 \cdot \cos \lambda)^2} \quad (14)$$

$$\cos \sigma = \sin U_1 \cdot \sin U_2 + \cos U_1 \cdot \cos U_2 \cdot \cos \lambda \quad (15)$$

$$\sigma = \text{atan2}(\sin \sigma, \cos \sigma) \quad (16)$$

$$\sin \alpha = \cos U_1 \cdot \cos U_2 \cdot \sin \lambda / \sin \sigma \quad (17)$$

$$\cos^2 \alpha = 1 - \sin^2 \alpha \text{ (trig identity; } \S 6)$$

$$\cos 2\sigma_m = \cos \sigma - 2 \cdot \sin U_1 \cdot \sin U_2 / \cos^2 \alpha \quad (18)$$

$$C = f/16 \cdot \cos^2 \alpha \cdot [4 + f \cdot (4 - 3 \cdot \cos^2 \alpha)] \quad (10)$$

$$\lambda' = L + (1-C) \cdot f \cdot \sin \alpha \cdot \{\sigma + C \cdot \sin \sigma \cdot [\cos 2\sigma_m + C \cdot \cos \sigma \cdot (-1 + 2 \cdot \cos^2 2\sigma_m)]\} \quad (11)$$

}

$$u^2 = \cos^2 \alpha \cdot (a^2 - b^2) / b^2$$

$$A = 1 + u^2/16384 \cdot \{4096 + u^2 \cdot [-768 + u^2 \cdot (320 - 175 \cdot u^2)]\} \quad (3)$$

$$B = u^2/1024 \cdot \{256 + u^2 \cdot [-128 + u^2 \cdot (74 - 47 \cdot u^2)]\} \quad (4)$$

$$\Delta \sigma = B \cdot \sin \sigma \cdot \{\cos 2\sigma_m + B/4 \cdot [\cos \sigma \cdot (-1 + 2 \cdot \cos^2 2\sigma_m) - B/6 \cdot \cos 2\sigma_m \cdot (-3 + 4 \cdot \sin^2 \sigma) \cdot (-3 + 4 \cdot \cos^2 2\sigma_m)]\} \quad (6)$$

$$s = b \cdot A \cdot (\sigma - \Delta \sigma) \quad (19)$$

$$\alpha_1 = \text{atan2}(\cos U_2 \cdot \sin \lambda, \cos U_1 \cdot \sin U_2 - \sin U_1 \cdot \cos U_2 \cdot \cos \lambda) \quad (20)$$

$$\alpha_2 = \text{atan2}(\cos U_1 \cdot \sin \lambda, -\sin U_1 \cdot \cos U_2 + \cos U_1 \cdot \sin U_2 \cdot \cos \lambda) \quad (21)$$

Where:

- ◆ s is the distance (in the same units as a & b)
- ◆ α_1 is the initial bearing, or forward azimuth
- ◆ α_2 is the final bearing (in direction $p_1 \rightarrow p_2$)

Figure 17 - Vincenty's Formula (source: <http://www.movable-type.co.uk/scripts/latlong-vincenty.html>)

4.3.5 Wi-Fi Monitoring Algorithm

This is a simple algorithm that required to be run in either an Activity or Service Android context in order to obtain system service information. Which in this design, it is included with SmartRoute background server (to be explained in the next section).

The algorithm as shown in Figure 18 is basically to check if the current Android device has its Wi-Fi function enable or disable. And if it is enable, then it will check for its current connectivity whether or not it is connected to the access point (assume this is not the first time usage) and if it is connected and it matches the access point SSID, then this will do nothing. But if the Wi-Fi is not connected, then it will scan the

surrounding for any wireless access point and search through if any of the access point SSID matches the required SSID; if matches then it will disable current and connect to it, else it will do nothing.

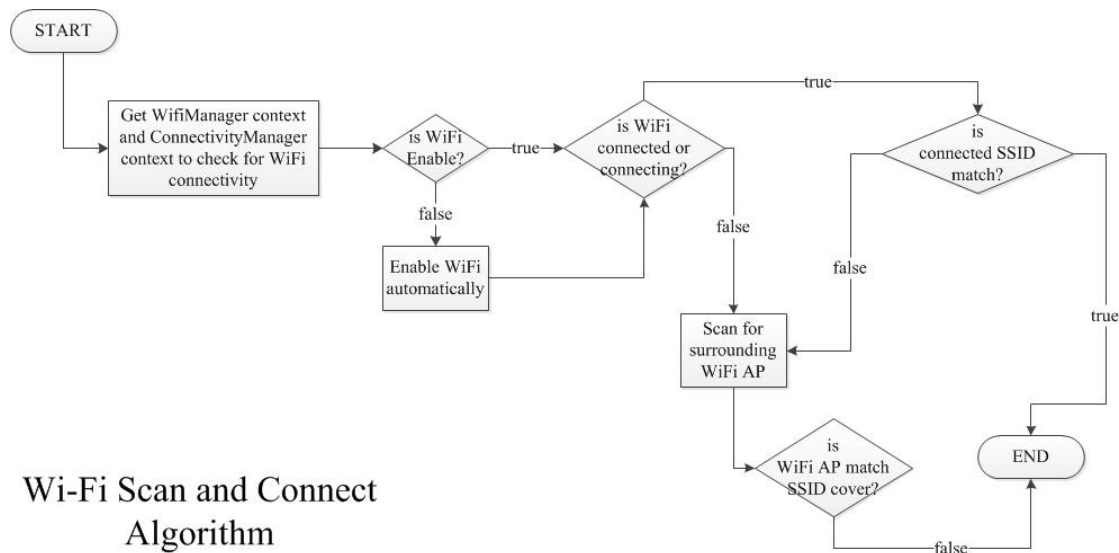
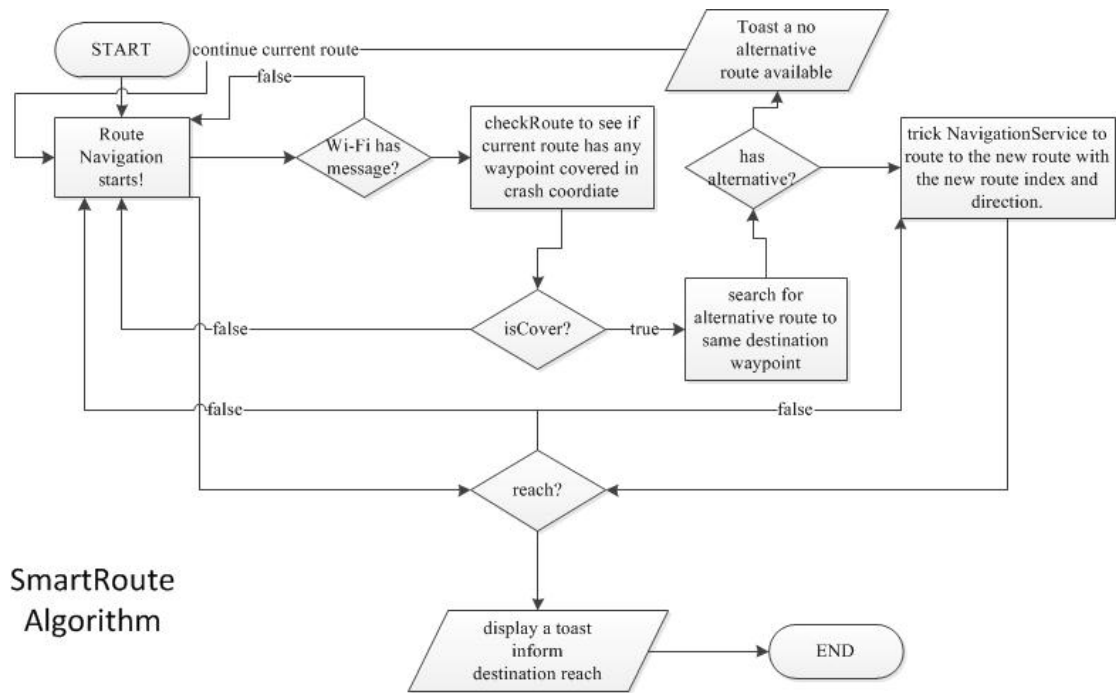


Figure 18 - Wi-Fi Scan and Connect Algorithm Flow Chart

4.3.6 “SmartRoute” Background Service

To begin, this service is bound to the NavigationService which is believe to be necessary as the SmartRoute service only runs when the user is Navigating but not when the user is browsing or searching for locations. In Figure 11 depicts the flow of the service as the flow of the message is passed and received within the application as it is pretty much of self-explanatory.

A Wi-Fi monitoring thread will be spawn to constantly monitor the network for any incoming UDP message with destination port 8900. Once receive, the action given is to acquire the current navigating route information, this is to check against the current route waypoint geographic coordinates against the crash coordinates. If there is any current route waypoints that is within the crash point coverage, then it will post an alert dialog box to user to inform that there is an accident up ahead and require an action from user, given that the user click yes, then the SmartRoute service will perform a scan through the list of available routes which has the same final destination waypoint as the current route final destination waypoint. It will then pass the route information to the NavigationService to switch to the new route.

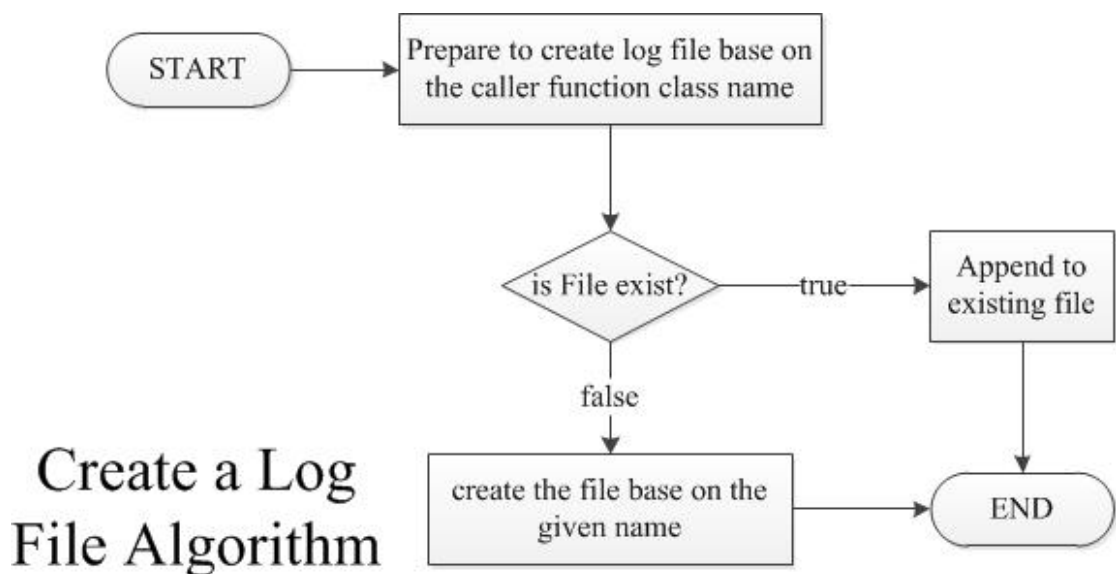


SmartRoute Algorithm

Figure 19 - SmartRoute Algorithm

4.3.7 Create Log File Algorithm

This create log file algorithm is responsible for creating the log file for data analysis during system testing, debugging and implementation phase. Though, this is something simple, the idea is to either create a new log file if there isn't any existed in the SD Card or if there is an existed file, then it will just append the debugging information into it.



Create a Log File Algorithm

Figure 20 - Create Log File Algorithm Flow Chart

4.3.8 Coverage Area Algorithm

This particularly subsection involve some geometry calculation as this is dealing with geographical coordinates. In order to get the coverage area, first of all, is to get any waypoint to mark as the centre of a circle with a fix coverage radius to scan for. In this case, the coverage area is approximately 100m (about 2 to 3 wireless routers apart, assuming the coverage is perfect). Hence, this can detect the distance between the crash point and the waypoint around it. The geographical coordinates are than converted into Cartesian coordinates to enable easy understood of where on earth that the user is standing right now. Hence, the formulae of converting latitude, longitude and spheroidal height into X, Y, and Z Cartesian coordinates are as follow:

Equation 1

$$X = (N + h) \times \cos \varphi \cos \lambda$$

Equation 2

$$Y = (N + h) \times \cos \varphi \sin \lambda$$

Equation 3

$$Z = ((b^2 \div a^2)N + h) \sin \varphi$$

Equation 4

$$\therefore N = \frac{a^2}{\sqrt{a^2(\cos \varphi)^2 + b^2(\sin \varphi)^2}}$$

Where,

X, Y, Z are the cartesian coordinates of the point

φ, λ are the latitude, longitude of the point

h is the height of the point above the spheroid along the ellipsoidal normal

N is the radius of curvature in the prime vertical at the point

This mathematical model data is based on WGS-84 datum to obtain the value for earth semi-major axis (6,378,137) and earth semi-minor axis (6,356,752.314245). This mathematical model calculation is based on [17] and converted into programming code by the report author.

4.4 Network Design

Of course, with all the algorithm and design mention above, without a reliable Wi-Fi network, none of the above will be working as the SmartRoute can't receive any message containing the geographical coordinate of the accident point.

Hence, in order to create a reliable Wi-Fi network, the selection requirement as mention earlier is that the Wi-Fi access point can serve multiple client and at the same time, bridging between 2 or more Wi-Fi access point together to form a bigger network.

4.4.1 Wireless Mesh Network Topology

With the requirement stated previously, a wireless mesh network (WMN) would fit into it. Since the wireless router acquire in this project is running on OpenWrt, which itself has a readily installed wireless mesh network routing protocol, OLSR (as explained previously in Chapter 3.5 Technology Involve). The wireless mesh network topology is as depicted in Figure 12, as according to this report conditions, shows each wireless access point coverage area overlap with each other in order to bridge together and provide client (Android Phone) wireless access. The different color represents each wireless access point coverage area respectively.

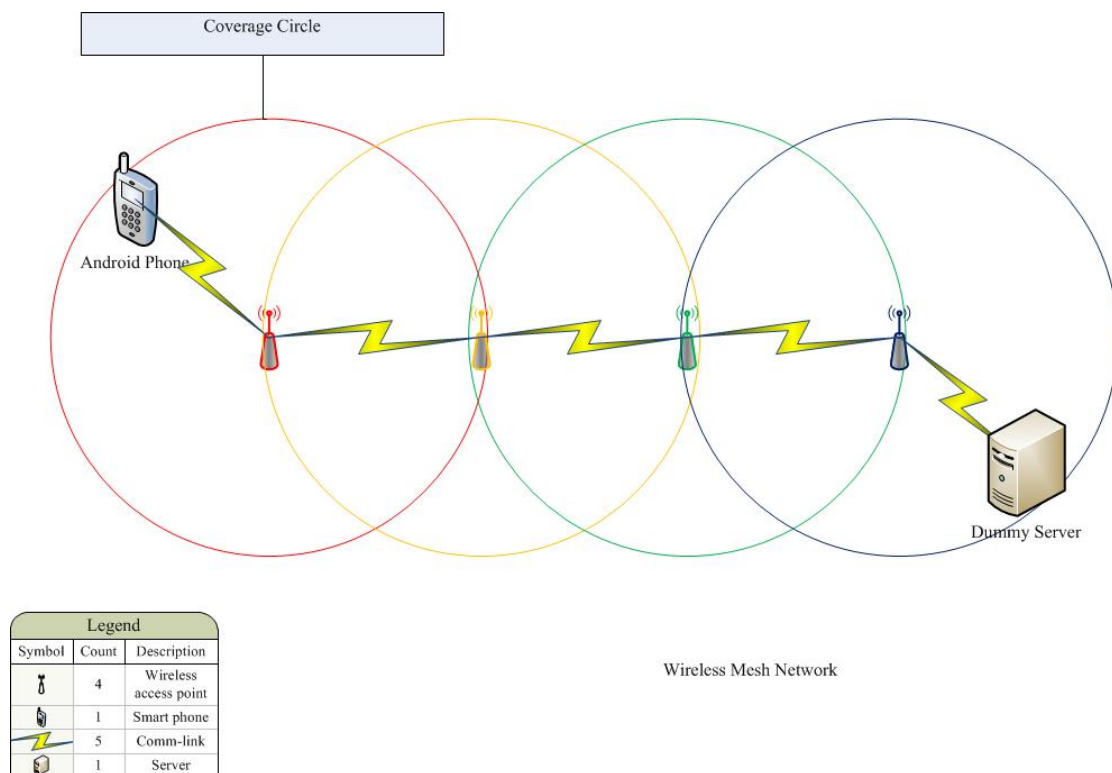


Figure 21 - Wireless Mesh Network

4.5 Summary and Concluding Remarks about Chapter 4

To summarise the whole chapter, this chapter explains mainly on the system design for the hardware, software and network aspects of the project. Under the hardware design aspect, it covers only the wireless router that is going to be used in this project as this project main objective is more towards the software design.

In the software design aspect, the used of an open source project, Androzic, is to implement author own algorithm, SmartRoute background service into it, which has a flow chart to display how it would work in theory. The algorithm is to be automated with little or no interaction with user (probably the only interaction is the alert dialog box). Other functions that are used in this project also have their basic algorithm to display in the flow charts.

In the network design aspect, this project utilize the wireless routers as mention in hardware aspect, is to form a working mesh network topology. It purposes is to enlarge the wireless router coverage and serve as access point for the user to connect to.



CHAPTER 5

5.1 Software Design Implementation

As explained in the previous chapter, chapter 4. This project covers 3 points of design, hardware design, software design and network design and that the flowcharts as depicted above are simplified version of the actual implementation. However, in hardware design, it covers only the wireless router that is going to be used in the network design as the hardware is not technically design by the report author. As for the software design, this chapter will give a full explanation of the SmartRoute algorithm, Wi-Fi scan and connect algorithm and the used of the coverage area algorithm as well as the network design for this project.

5.1.1 “SmartRoute” System

To begin with, writing an Android application share several similarities of writing a Java program since it also uses some of Java API (though, one would need at least an Eclipse IDE installed); hence, the reason for SmartRoute algorithm to be written in a Class file. SmartRoute algorithm is design to run as a background service which technically there is not a need to have any interaction between this and the user. And in this section, it’s a sum up of the entire algorithm (SmartRoute, NavigationServer and Coverage Area) as mention previously in Chapter 4 since they work concurrently.

This background service cannot be started by user but it is started automatically whenever user uses “Navigate via route” as depicted in Figure 15 and in Figure 16 shows the navigation route (yellow line heading North-West) from current location (when the NavigationService started). Do notice that the notification bar has two small icons ( and ) which represents NavigationService and SmartRoute service have started respectively.

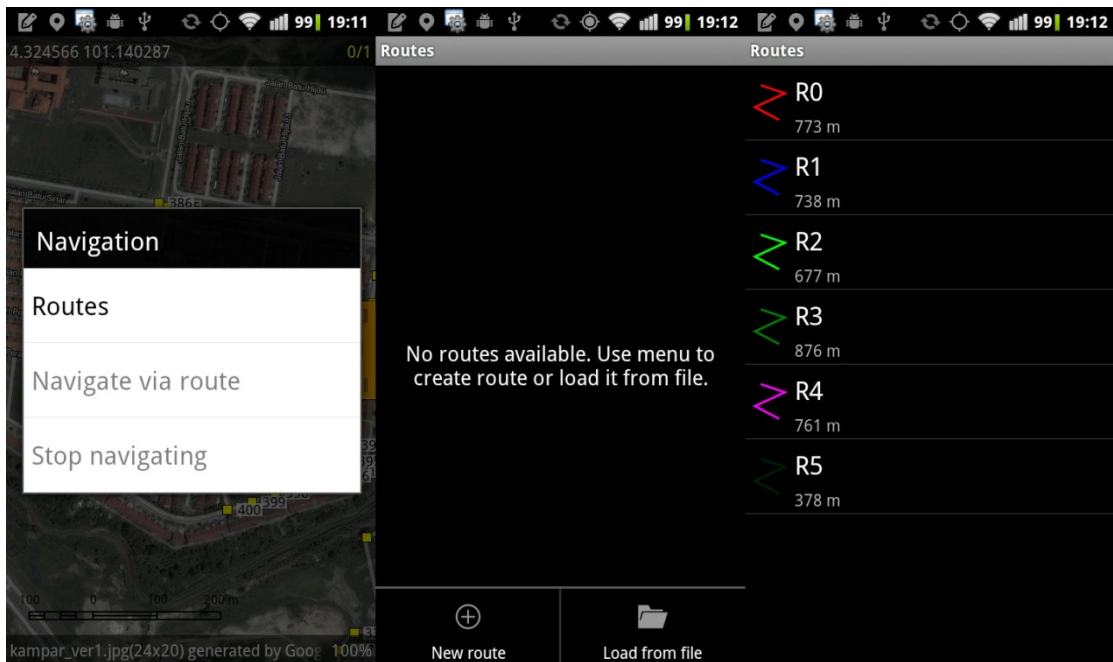


Figure 22 - Screen Cap For Navigation



Figure 23 - SmartRoute and Navigation Started

Since SmartRoute has a method which runs a `TimerTask()` to perform scheduling with an interval of 5 minutes (30000 milliseconds) to run a separate thread for monitoring of receiving any message from port 8900 and UDP stream of data. The method of how does `TimerTask()` runs will not be touch here as it is provided by Java API. Hence, when the dummy server sends a broadcast message via UDP stream at port 8900, SmartRoute will be able to detect it and check if the given geographic

coordinate covers with an area of 100 meter radius. If it there is any waypoint geographic coordinate is within the radius, SmartRoute will pop an alert dialog box to prompt user for action (shown in Figure 17).



Figure 24 - Crash Alert Dialog Box

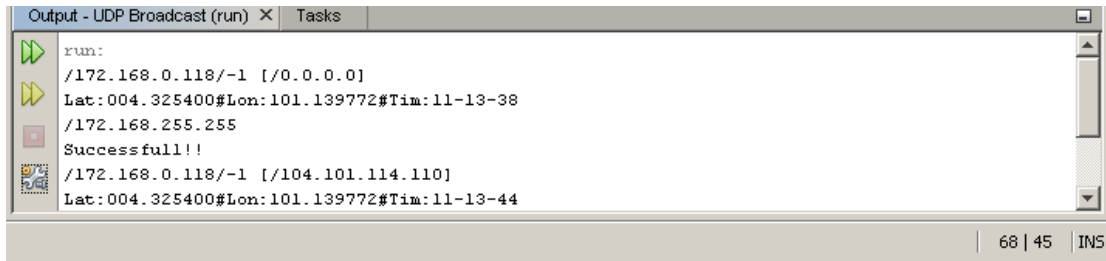
Once the user click on “Yes”, then the SmartRoute will run a check of the existing route table available (Figure 15) and search if there is any available alternative route to the same last waypoint geographic coordinate and compile a list of route index (labelling for each route) base on the search result. Once the search is complete, the list of route index will be compared with each other to search for the shortest distance and it will ask NavigationServer to switch to the new given route.

5.1.2 Creating Log File

As explained previously in Chapter 4, Create Log File Algorithm; this section has already a detail explanation of how it would flow with it flow chart as shown in Figure 20. The log file will store information with the following time format “dd/MM/yyyy HH:mm:ss:SSS” where the formatting explanation could be found in Java API and the message that is parsed from the caller function.

5.1.3 Dummy Server Broadcaster

This dummy server broadcaster is written in Java language. Its main role is just to send the crash point dummy message with a format of (Lat:xxx.xxxxxx#Lon:xxx:xxxxxx#Tim:xx-xx-xx). A screen capture of it running in NetBeans IDE as depicted in Figure 18. However, do note that the broadcast address is manually entered as a bug reported in [18] shows that Java API has a limitation of obtaining the netmask for wireless devices (regardless it's a USB dongle or on-board Wi-Fi module)



```
Output - UDP Broadcast (run) x Tasks
run:
/172.168.0.118/-1 [/0.0.0.0]
Lat:004.325400#Lon:101.139772#Tim:11-13-38
/172.168.255.255
Successfull!!
/172.168.0.118/-1 [/104.101.114.110]
Lat:004.325400#Lon:101.139772#Tim:11-13-44
68 | 45 | INS
```

Figure 25 - Dummy Server

5.2 Network Design Explanations

The network topology as mentioned in the previous chapter, Chapter 4, is a mesh network and the configuration would be eased up by the OpenWrt LuCI web user interface. In this section, it will cover on OpenWrt CLI configuration for network and wireless while the OLSRD is configured via LuCI webUI. For names convenient, the naming of the wireless routers from Linksys and CeedTec is given as Node1, Node2, Node3, and Node4 which represents Linksys WRT54Gv2, Linksys WRT54GL, and two CeedTec MPC8241-EP-M3 respectively.

A side note regarding the DHCP server, it is disabled in Node2 and Node3. Node1 serves as the main DHCP server while Node4 serves as a backup DHCP server to serve as a redundant in case of main DHCP server is not reachable. However, the configuration of DHCP server is not covered here as it is left at default values by OpenWrt.

5.2.1 OpenWrt Configuration

Configuring OpenWrt is easy and doesn't really take much of a time to complete the configuration unless faulty hardware is at fault. Node1 and Node2 share the same configuration as both use the same wireless chipset but it is slightly different to

Node3 and Node4 where both of these have different wireless chipset. But since both are using OpenWrt, configuration wise would be nearly identical; hence, screen captured only show Node1 instead of showing all other nodes (Node2, Node3 and Node4).

Figure 19 and Figure 20 shows a PuTTY client (an SSH/Telnet client) that is connected to Node1. The configuration shown is the current configuration in used as of this report is written. Figure 21 shows Google Chrome (a web browser) is browsing the Node1 OLSRd configuration page, all settings are left at default value except that the listening interface is set to wlan. Now, the beauty of OLSRd is that the configuration is automated in condition that the wireless configuration has to be set with the same SSID and share a common BSSID.



```
172.168.1.1 - PuTTY
root@WRT54Gv2:~# cat /etc/config/wireless

config 'wifi-device' 'radio0'
    option 'type' 'mac80211'
    option 'macaddr' '00:13:10:2f:64:d5'
    option 'channel' '6'
    option 'txpower' '20'
    option 'country' 'MY'

config 'wifi-iface'
    option 'device' 'radio0'
    option 'encryption' 'none'
    option 'network' 'wlan'
    option 'mode' 'adhoc'
    option 'bssid' '00:13:10:2F:64:D5'
    option 'ssid' 'utarfyp'

root@WRT54Gv2:~#
```

Figure 26 - Node1 Wireless Configuration

```

172.168.1.1 - PuTTY
config 'interface' 'loopback'
  option 'ifname' 'lo'
  option 'proto' 'static'
  option 'ipaddr' '127.0.0.1'
  option 'netmask' '255.0.0.0'

config 'interface' 'lan'
  option 'proto' 'static'
  option '_orig_ifname' 'eth0.0'
  option '_orig_bridge' 'false'
  option 'ipaddr' '192.168.1.1'
  option 'netmask' '255.255.255.0'
  option 'ifname' 'eth0.0'

config 'interface' 'wan'
  option 'ifname' 'eth0.1'
  option 'proto' 'dhcp'

config 'interface' 'wlan'
  option 'proto' 'static'
  option 'ipaddr' '172.168.1.1'
  option 'netmask' '255.255.0.0'

root@WRT54Gv2:~#

```

Figure 27 - Node1 Network Configuration

The OLSR daemon is an implementation of the Optimized Link State Routing protocol. As such it allows mesh routing for any network equipment. It runs on any wifi card that supports ad-hoc mode and of course on any ethernet device. Visit olsrd.org for help and documentation.

General settings

General Settings | Link Quality Settings | SmartGW | Advanced Settings

Internet protocol: IPv4
 IP-version to use. If 6and4 is selected then one olsrd instance is started for each protocol.

FIB metric: flat
 FIBMetric controls the metric value of the host-routes OLSRd sets. "flat" means that the metric value is always 2. This is the preferred value because it helps the linux kernel routing to clean up older routes. "correct" uses the hopcount as the metric value. "approx" use the hopcount as the metric value too, but does only update the hopcount if the nexthop changes too. Default is "flat".

Port: 698
 The port OLSR uses. This should usually stay at the IANA assigned port 698. It can have a value between 1 and 65535.

Main IP: 0.0.0.0
 Sets the main IP (originator ip) of the router. This IP will NEVER change during the uptime of olsrd. Default is 0.0.0.0, which triggers usage of the IP of the first interface.

Interfaces Defaults

General Settings | IP Addresses | Timing and Validity

Mode: mesh
 Interface Mode is used to prevent unnecessary packet forwarding on switched ethernet interfaces. valid Modes are "mesh" and "ether". Default is "mesh".

Weight: 0
 When multiple links exist between hosts the weight of interface is used to determine the link to use. Normally the weight is automatically calculated by olsrd based on the characteristics of the interface, but here you can specify a fixed value. Olsrd will choose links with the lowest value. Note: Interface weight is used only when LinkQualityLevel is set to 0. For any other value of LinkQualityLevel, the interface ETX value is used instead.

LinkQuality Multiplier: default 1.0
 Multiply routes with the factor given here. Allowed values are between 0.01 and 1. It is only used when LQ-Level is greater than 0. Examples:
 reduce LQ to 192.168.0.1 by half: 192.168.0.1 0.5
 reduce LQ to all nodes on this interface by 20%: default 0.8

Interfaces

Enable	Network	Mode	Hello	TC	MID	HNA
<input checked="" type="checkbox"/>	wlan:	mesh	5.0s / 40.0s	2.0s / 256.0s	18.0s / 324.0s	18.0s / 108.0s

[Add](#)

Figure 28- Node1 OLSRd Configuration Page

5.3 Software Used In System Implementation

In this section, the Eclipse IDE that is setup and run to implement the algorithms as mention in the previous chapter, Chapter 4. This would also include the installation of Android SDK, Android NDK and Cygwin Terminal.

5.3.1 Android Development Installation

First of all, Android SDK is a must to be installed (obtainable from <http://developer.android.com/sdk/index.html>); else this would not be called Android Development! The installation of Android SDK, assuming the reader is also uses Windows platform for development, then installing it is the same as installing any application one would install in their computer as it just require to follow the on-screen instruction to install it properly as there isn't any special instructions for this (so, there is no screen capture for this part). After finish installing, it is a must to update the Android SDK tools (via Android SDK Manager) to the latest version and also download Android 2.3.3 (API 10) SDK platform and Google APIs as shown in Figure 22.

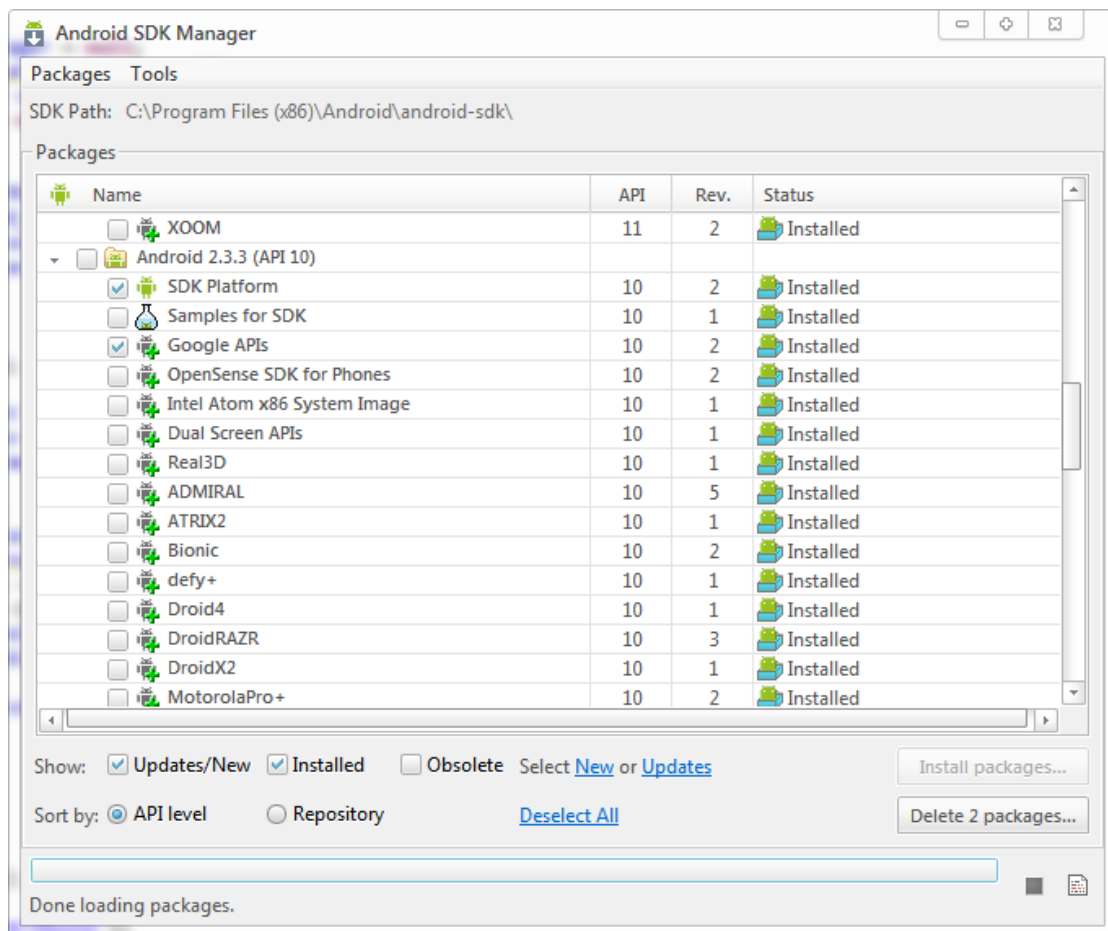


Figure 29 - Android SDK Manager

Secondly, the installation of Eclipse is as easy as ABC as there is no setup is required. Just download from Eclipse official site (<http://eclipse.org/downloads/>) and unpack the zip to C:\Eclipse. Once it is completely unpacked, run Eclipse.exe and go to install new software (Help > Install New Software...) to install Android Development Tools (ADT) which require to be obtain from <https://dl-ssl.google.com/android/eclipse/>.

Thirdly, due to the compiling of Androzcic would require the use of Android NDK. Android NDK installation is fairly simple and straight forward as it just require to download the zip file from Android website (<http://developer.android.com/tools/sdk/ndk/index.html>) and unpack it to the user desire location (advisable to be at root as there are issues associated with whitespace between folder names when associating from Eclipse and Cygwin). In conjunction with the installing of Android NDK, the requirement for it to work for Windows is Cygwin 1.7 or higher. Hence, for Cygwin installation, it is required that all package

that is related to GNU Make 3.81 and above to be selected and install in order for Android NDK to work properly.

And finally, for everything mention above to work together, first is to import Androzic into workspace and edit the launch configuration properties to the following parameters (this is based on author parameters):

Location: *c:\cygwin\bin\bash.exe*

Working Directory: *c:\cygwin\bin*

Arguments: *--login -c "cd /cygdrive/c/Users/khord/workspace/Androzic/jni && /cygdrive/c/Eclipse/android-ndk-r8/ndk-build APP_ABI=all"*

Now with this complete, Eclipse should be able to compile Androzic successfully.

5.4 Summary and Concluding Remarks For Chapter 5

To summarise everything in short, this chapter touches about the software design implementation, network design explanation and software in used installation. In software design implementation, the SmartRoute is a background service that runs a separate thread to monitor for message and its own thread is use for heavy calculation.

CHAPTER 6

6.1 System Verification

In this section, system verification will cover the final stage of accessing the Androzic with SmartRoute algorithm integrated. Just a recap of the project scope, is to reroute user upon receiving a traffic congested message.

However, as describe in Chapter 3, 3.2.2 System Performance Definition and 3.2.3 System Verification Plan, all the testing are done in a real physical Android device (Motorola Milestone) instead of AVD due to the limitation of AVD have no access to the physical wireless mesh network. The wireless mesh network is all done in the laboratory since the CeedTec routers are only available in the laboratory compound.

Hence, for this system verification purpose, a log writing function is also integrated with Androzic as all output is directed to “/sdcard/Androzic/log/<function_called>.log”. The source code is available in the disc as well.

For performance evaluation, the target test is on Androzic and the wireless mesh network. The characteristic of the test done on Androzic is:

- a) Speed of receiving the message
- b) Speed of checking the current route
- c) Alert the user and the speed of rerouting.

Therefore, the result for the test is done with an average of 11 seconds for Androzic to receive the message, while for the speed of checking the current route and alert the user and the speed of rerouting takes less than 2 seconds. For field test result, kindly refer to the objective achievement section.

As for the wireless mesh network, the naming convention of the wireless routers are the same as previous chapter has mention, Node1 (10.0.0.1), Node2 (10.0.1.1) and Node3 (10.0.2.1), and Node4 (10.0.3.1). The test characteristic of the wireless mesh network is as follow:

- a) The throughput of UDP stream
- b) Ping test
- c) OLSR expected retransmission count (ETX)

The result for the network test for the throughput of UDP streams between the author computer and the dummy server has an average of 25.375 KBps where the author is connected to Node2 and the dummy server is connected to Node3.

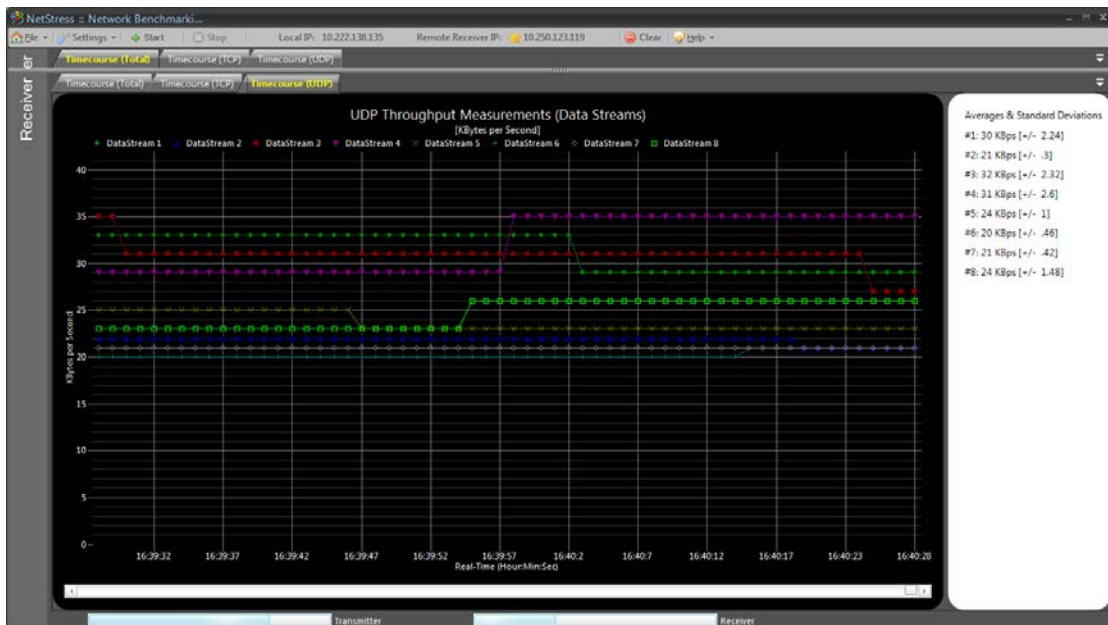


Figure 30 - NetStress Network Benchmarking Tool Result

The result for the ping test, Node3 and Node4 has a 100% packet drop when they ping to each other while the average packet drop for Node 1 and Node2 is around 10%. However for Node3, ping to Node1 and Node2 has an average of 65% of packet drop rate!

The result for the OLSR ETX (lower is better), based on Node2 topology, shows an unsatisfactory result between Node3 and Node4 while Node1, Node2 and Node4 show acceptable level of ETX rate.

Active OLSR nodes				
Overview of currently known OLSR nodes				
OLSR node	Last hop	LQ	NLQ	ETX
10.0.0.1	10.0.1.1	0.940	0.748	1.418
10.0.0.1	10.0.2.1	1.000	0.164	6.071
10.0.0.1	10.0.3.1	1.000	0.282	4.541
10.0.1.1	10.0.0.1	0.831	0.940	1.257
10.0.1.1	10.0.2.1	1.000	0.129	7.727
10.0.1.1	10.0.3.1	1.000	0.360	2.771
10.0.2.1	10.0.3.1	0.211	0.230	20.409
10.0.2.1	10.0.0.1	0.164	1.000	6.071
10.0.2.1	10.0.1.1	0.129	1.000	7.727
10.0.3.1	10.0.0.1	0.329	1.000	3.035
10.0.3.1	10.0.2.1	0.255	0.227	17.247
10.0.3.1	10.0.1.1	0.317	1.000	3.147

Figure 31 - Node 2 Active OLSR Nodes | OLSR Topology (Lower Number is Better)

6.2 Case Study

This section will explain the test conducted based on the system verification and their results. However, since this is a mobile navigation application, it is best to have a live demonstration of what it does.

6.2.1 Environment Description

The general environment where the entire test conducted for both the AndroziC and wireless mesh network are mostly indoors where the Android device would need to be place near a window to get GPS signal. The environment is cold as the laboratories at top floor are always cold and as this would have a possibility of affecting the wireless signal strength.

As for the on the field testing, Figure 32 (from OziExplorer) shows the actual map location for the real wireless access point which is situated at waypoint 67, while there is a black arrow pointing at the crash site (faking an accident) and with an assumption that throughout the road has the wireless access point which could pass the message to the wireless access point at waypoint 67. The red spot is the author test start point with motorcycle moving at an average of 20 KM/H.

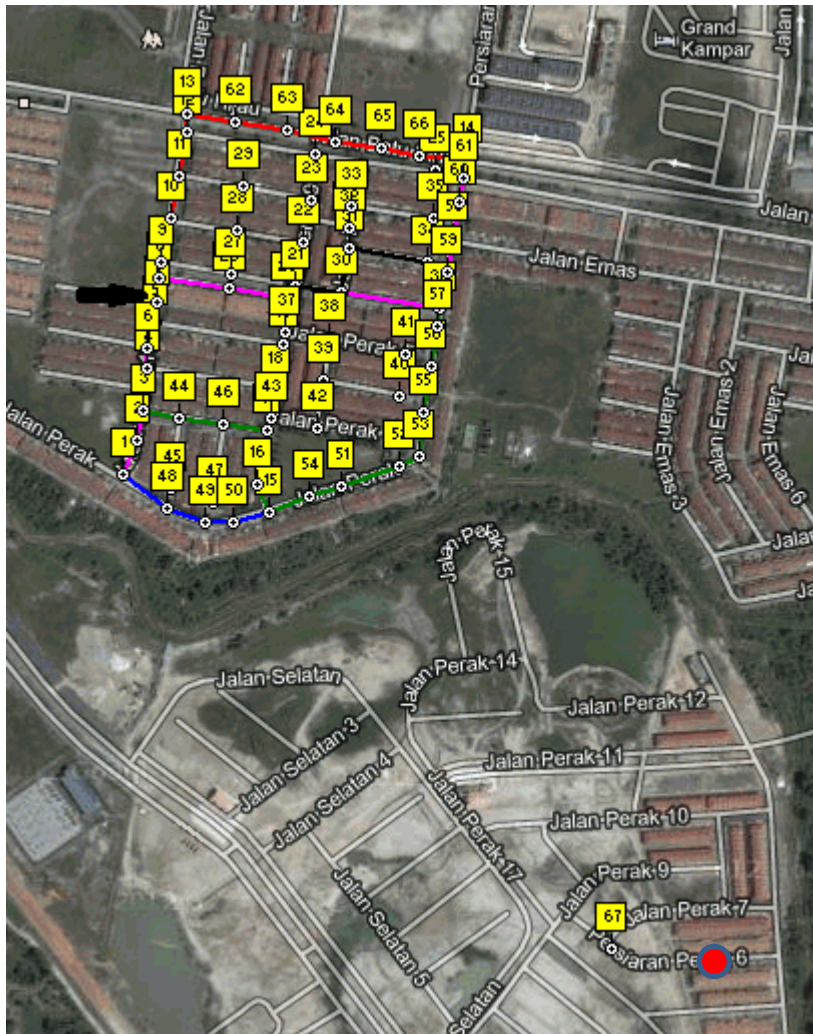


Figure 32 - Field Test Map

6.3 Objective Achievements

This section is just a checklist for verifying the achievement for each objective as stated in Chapter 1.

The following are the objectives,

- a) Search for an open source Android GPSNAV application to test the routing system.

This is done at early stage with the finding of Waze. However, the source codes given are not completed or not workable as Eclipse was able to compile correctly but whenever it is tested in AVD, it crash. After another few days of search and testing, found Androziq which has a complete source code uploaded in Google Code. The

understanding of the routing system by Androzic is explained in Chapter 4 under 4.3.2 Androzic and 4.3.3 Androzic “NavigationService”.

- b) Code a program capable of sending a UDP packet message containing GPS coordinate and timestamp.

This is done at around week 3 or week 4 of the semester. The use of Java to send a UDP broadcast at port 8900 along with the message. This is explained in Chapter 5 under 5.1.2 Dummy Server.

- c) Simulate a congested packet receive via Wi-Fi to let the system reroute.

This is done along with the system verification stage. With the use of the dummy server to broadcast the message, it contains the latitude, longitude and timestamp to be sent out. The format of the message is (Lat:xxx.xxxxxx#Lon:xxx.xxxxxx#Tim:xx-xx-xx).

- d) On-the-road experiment for successful routing and re-routing

This is partially done if considering the limitation and fault at CeedTec routers that the experiment is done with limited wireless coverage. Hence, the test is brought indoor. The Androzic is able to reroute the author upon receiving the message with the screen capture show in Figure 17 (in Chapter 5).

The field test result, as based on the system verification information and map details from Figure 32. It takes around 2 seconds for the Androzic to pick-up the wireless access point signal and connects to it. The reroute is done in less than a seconds.

6.4 Issues

In this section, some of the problems that were encountered are ranted here as it contributed to the project delay in producing the product. Though, this may not be a comprehensive list of complaints that the author would want to voice out.

6.4.1 Android Application Lifecycle And Big Project

Not to be expected that the development of an Android application would be a nightmare. Though, going through the tutorial are easy as it seems but when trying to understand the application flow of Androzic, a big project, it took months of

frustration and trial-error and not even understand how Androzic works in general. Attempt to contact the Androzic developer with approval given by the supervisor, the developer doesn't reply to any email being sent.

With things like this, the author uses a reversed engineer UML generator to generate an Entity Relationship Diagram (ERD) in order to understand it but because of the complexity, even the generated ERD has too many relationships and that it is too big to print out (an ERD for all com.androzic related package would need 12 A4 pages to resemble it).

Another thing about Android application is that it is a heavy multi-threaded where the author has limited knowledge in this field. For an instance, pass a message or arguments from a background thread would require sending Intent via broadcast intent and that the receiver would need to register the broadcast receiver and have an Intent Filter to monitor the Intent. Of course, after all the sweats and swearing, the author just understand a gist of it only.

6.4.2 Wireless Connectivity

There are some minor issues with installing the OLSRd, LuCI and Freifunk-mod in the CeedTec routers as it has many dependents are not installed and that the OpenWrt branch has no individual package for CeedTec routers since all the package were compile by author himself and with the help of Mr. Liew patch file.

Though, soon to found out that the dependencies are not required by OLSRd, hence it was not further to troubleshoot. Although, OLSRd are installed but there are constant issue with LuCI for not able to connect to OLSRd for every configuration is amended. Currently the settings for OLSRd in both CeedTec routers are at their default values.

Another issue arises after settling the package installation when the CeedTec routers are apart from each other, the connectivity drop dramatically with the ping result hitting the highest 100% packet drop rate and lowest at 45%. Has consulted with Mr. Goh and tested the received signal strength indication (RSSI) as it shows an unsatisfied value of -53db as according to Mr.Goh. Hence, the idea of using the CeedTec routers is scrap with the only reliant of the Linksys routers.

6.4.3 Maps and Waypoints

This is extremely labour intensive as Androzic has nothing about waypoints and routes for Malaysia. A raw map obtain from Google and loaded it up in OziExplorer would require to calibrate and this is not 100% accuracy as mainly it is manual plotting of 4 points for North, East, West and South. However, did manage to download a map from Google Map with WGS-84 datum embedded via OziGoogle application has save the problem from the map coordinates issue. However, the OziExplorer will only plot the waypoints and that each waypoint have no extra information except for the spot geographic coordinate.

6.5 Summary and Concluding Remarks For Chapter 6

In a summary, the system verification performance test as mentioned in this chapter sets a few characteristic to evaluate the processing speed of Androzic with SmartRoute integrated and the throughput of the wireless mesh network with 4 wireless routers. This gives the test an indicator of the result produce from the test being conducted. Next would be the case study where the test environment is always cold. It is followed by the objective achievements which mark it as a checklist for what the author have done and completed. And lastly are the issues that rise throughout the project timeline.

CHAPTER 7

7.1 Project Conclusion

In this project, as previously mention in Chapter 1, the issue with the dynamic changes towards the road condition contributes to the road user frustration as the GPSNAV is routing the road user through a congested road as road changes dynamically. Though, there is already current solution as provide by [5] and [6] for rerouting user through advance detection of road condition via satellite by [5] or internet community by [6]. However, in Malaysia, solution provided by [5] is not available and that only [6] is supported provided user as active internet connectivity.

This report also reviews other works of related interest in [7-10]. A quick summary of the methods that was used in these works is traffic information gathered and store in a database for later review by the system. Traffic information is gathered via GPS probes and GSM probes. And all these methods also have a type of artificial intelligent for self-learning and exploring routes.

To sum up everything mention in this report, the hardware design (wireless routers), software design (SmartRoute algorithm) and network design (wireless mesh network) is that for the SmartRoute algorithm to work, a working and reliable wireless mesh network is needed and in order for a reliable wireless mesh network, a good wireless routers which in this report is Linksys WRT54G (OpenWrt) is needed for stability and reliability. And with the system is well design and planned, a series of performance test were carried out with Androzic is performing well but not the wireless network. Of course, nothing is perfect and nothing can sail through a sea peacefully without some random storm ahead. Issues that occur in this project that contributed to the major delay is the author lack the knowledge of Android programming during the initial stage where understanding of the Androzic application took the toll of it.

7.2 Recommendation

This section composes of future works involving Android application development in general and GPS navigation system. As mention in Chapter 6, the maps and waypoint information would need to have better implementation. Although knowing that the commercial GPS navigation devices all has dynamic route, but their sources are all closed and only internal staff would know how it works. So, getting more information

into the waypoints, such as junction, traffic light, bridge, and other common road items that could differentiate between a walkable path or not walkable path (or drive).

As for the Android application, the background service is unable to disable in turn where the Androzic would crash or it has ended properly but with the NavigationService still active. And it should be able to route more dynamically without the need of a static route table.

REFERENCE

- [1] Kidcyber. (2011, December) Technology & Inventions. [Online].
<http://www.kidcyber.com.au/topics/inventor.htm>
- [2] Juerg Dedual. Virtual Archive of WILD HEERBRUGG. [Online].
http://www.wild-heerbrugg.com/gps_timeline.htm
- [3] Harald74. (2010, October) Ask MetaFilter. [Online].
<http://ask.metafilter.com/167974/How-does-a-GPS-calculate-which-route-to-display-Shortest-Fastest-Most-ridiculous>
- [4] Reed Stevenson. (2008, August) REUTERS. [Online].
<http://www.reuters.com/article/2008/08/14/us-column-pluggedin-idUSN1448237220080814>
- [5] TomTom International BV. (2012) TOMTOM. [Online].
<http://www.tomtom.com/>
- [6] Waze. (2012) waze. [Online]. <http://www.waze.com/>
- [7] TomTom International BV. (2009, December) TomTom. [Online].
www.tomtom.com/lib/doc/download/HDT_White_Paper.pdf
- [8] Yao Hua Ho, Yao Chuan Wu, Meng Chang Chen, Tsun-Jui Wen, and Y.S. Sun, "GPS Data Based Urban Guidance," in *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on* , , 2011, pp. 703 - 708. [Online].
<http://ieeexplore.ieee.org.libezp.utar.edu.my/stamp/stamp.jsp?tp=&arnumber=5992685&isnumber=5992565>
- [9] J.L. Duffany, "Artificial Intelligence in GPS Navigation Systems," *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on*, vol. 1, pp. 382-387, October 2010.
- [10] J.-W. Ding, C.-F. Wang, F.-H. Meng, and T.-Y. Wu, "Real-time vehicle route guidance using vehicle-to-vehicle communication," *Communications, IET*, vol.

4, no. 7, pp. 870 - 883, April 2010.

- [11] Sarah Kessler. (2011, July) Mashable Tech. [Online].
<http://mashable.com/2011/07/26/android-history-infographic/>
- [12] Android Inc. (2012, March) Android Developers. [Online].
<http://developer.android.com/guide/topics/fundamentals.html>
- [13] RFID Centre Ltd. (2012, August) RFiD Centre. [Online].
http://www.rfidc.com/docs/introductiontowireless_standards.htm
- [14] P. Jacquet T. Clausen. (2003, October) The Internet Engineering Task Force (IETF). [Online]. <https://tools.ietf.org/rfc/rfc3626.txt>
- [15] Andrey Novikov. (2012, May) androziC. [Online].
<http://code.google.com/p/androziC/>
- [16] Chris Veness. (2012, August) Movable Type Scripts. [Online].
<http://www.movable-type.co.uk/scripts/latlong-vincenty.html>
- [17] Luis Machado. (2012, August) Instituto Politécnico de Beja. [Online].
www.estig.ipbeja.pt/~legvm/cartmat/coisas/Conversion.doc
- [18] Oracle Corporation. (2012, April) Oracle. [Online].
http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=7158636
- [19] Hamid Faridani. (2011, March) A Guide to Selecting Software Development Methodologies. PowerPoint Slides. [Online].
http://www.gtislig.org/HamidFaridani_GuideToSelectingSWMethodologies_SOC_PDD_20110305.pdf
- [20] IEEE. (2012, March) IEEE Global History Network. [Online].
http://www.ieeeahn.org/wiki/index.php/Wireless_LAN_802.11_Wi-Fi
- [21] jow. (2011, November) OpenWrt Wiki. [Online]. <http://wiki.openwrt.org/start>
- [22] J. Postel. (1980, August) The Internet Engineering Task Force (IETF). [Online].

<http://datatracker.ietf.org/doc/rfc768/>

APPENDICE A – GANTT CHART

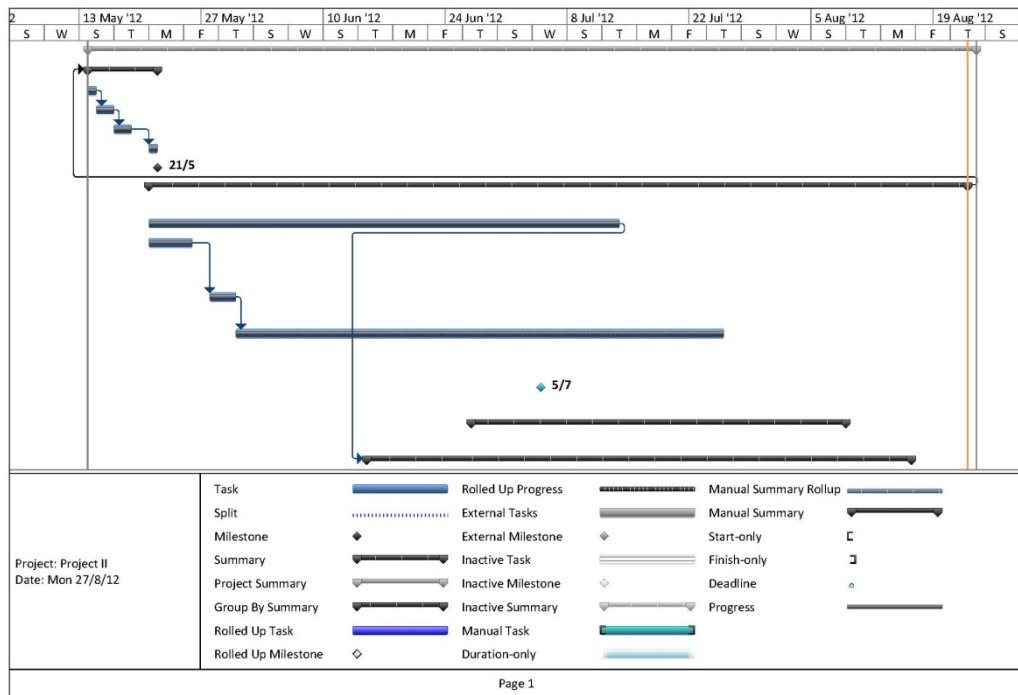


Figure 33 - Project II Gantt Chart Part 1

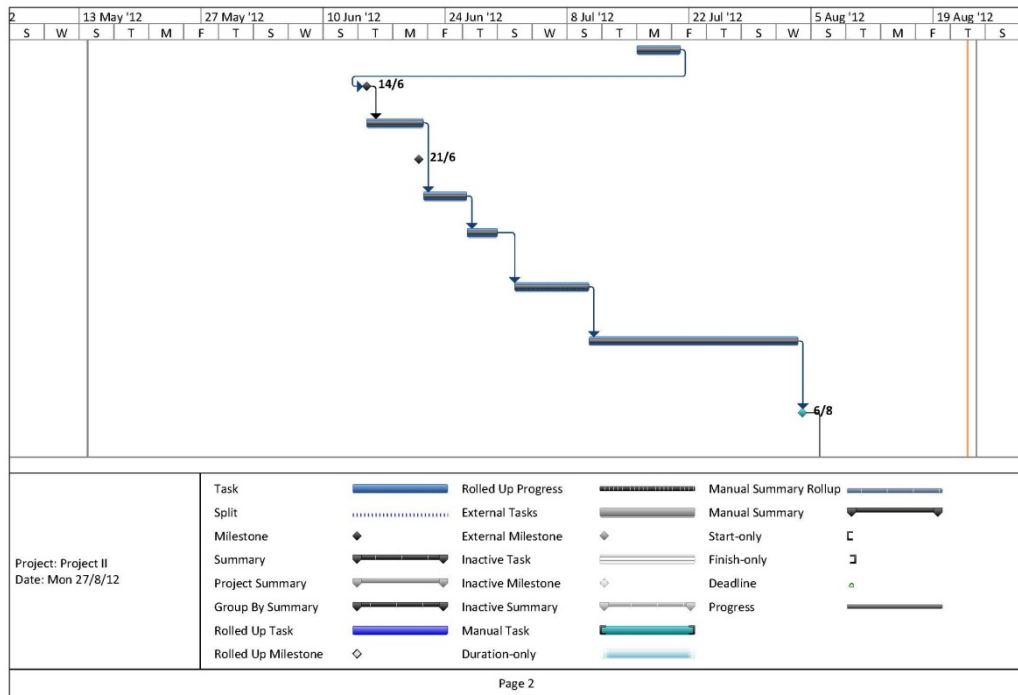


Figure 34 - Project II Gantt Chart Part 2

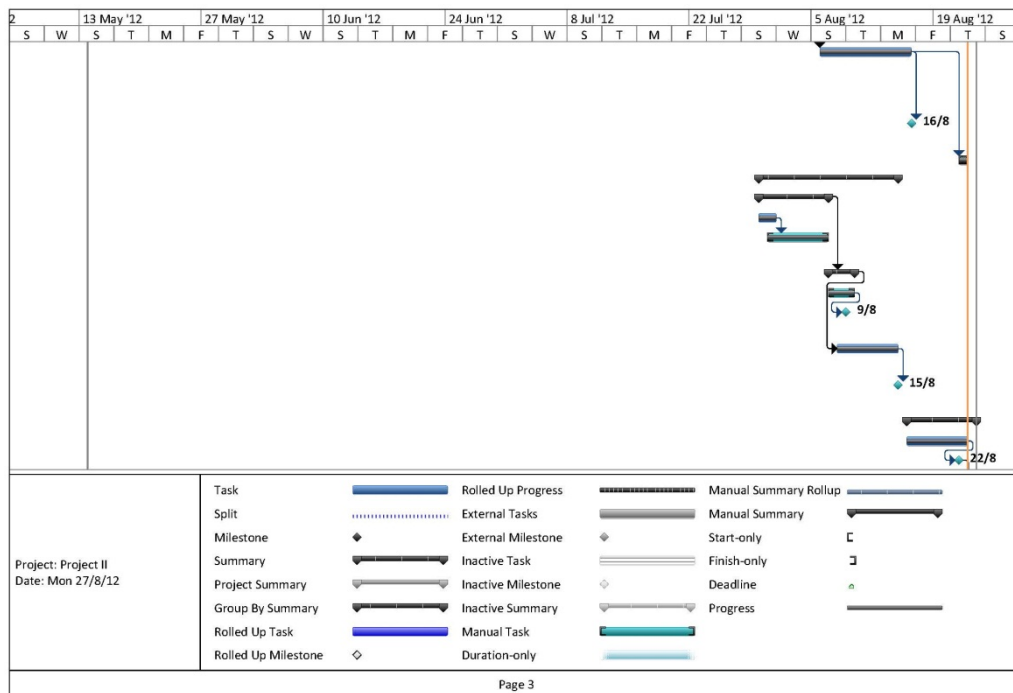


Figure 35 - Project II Gantt Chart Part 3

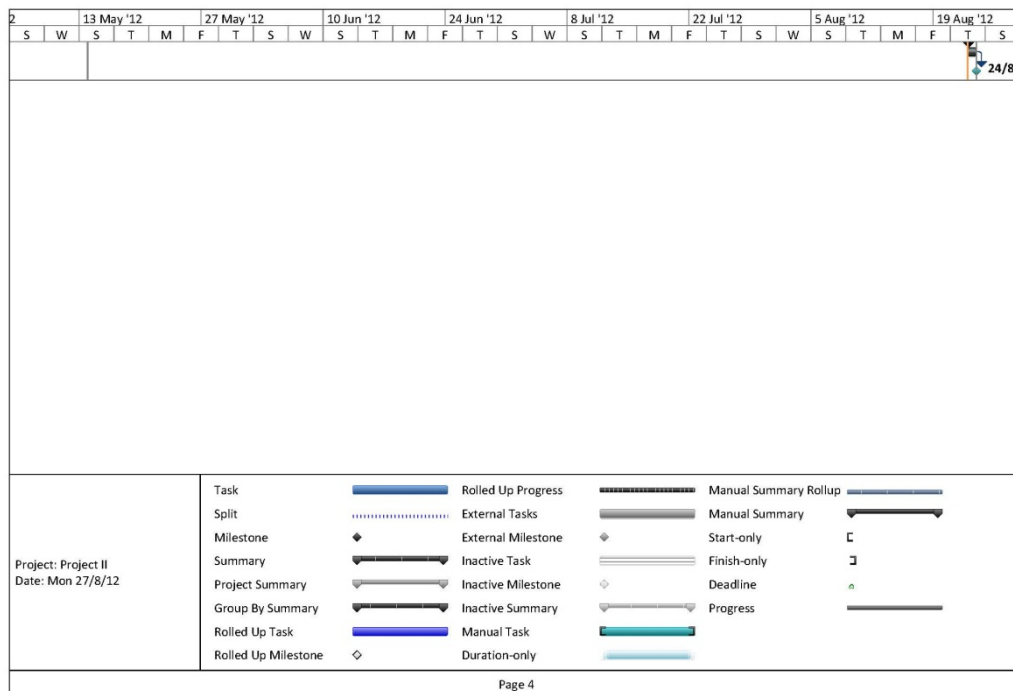
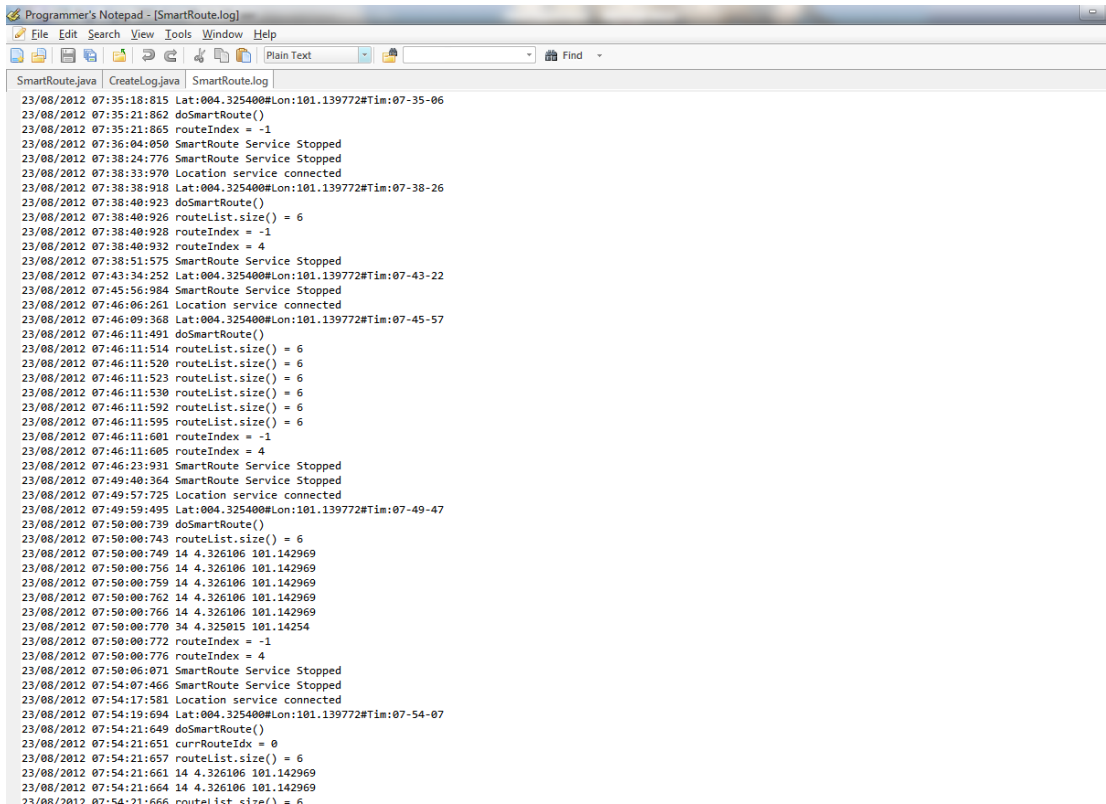


Figure 36 - Project II Gantt Chart Part 3

APPENDICE B – SAMPLE LOG FILES



```
Programmer's Notepad - [SmartRoute.log]
File Edit Search View Tools Window Help
Plain Text Find
SmartRoute.java CreateLog.java SmartRoute.log
23/08/2012 07:35:18:815 Lat:004.325400#Lon:101.139772#Tim:07-35-06
23/08/2012 07:35:21:862 doSmartRoute()
23/08/2012 07:35:21:865 routeIndex = -1
23/08/2012 07:36:04:050 SmartRoute Service Stopped
23/08/2012 07:38:24:776 SmartRoute Service Stopped
23/08/2012 07:38:33:970 Location service connected
23/08/2012 07:38:38:918 Lat:004.325400#Lon:101.139772#Tim:07-38-26
23/08/2012 07:38:40:923 doSmartRoute()
23/08/2012 07:38:40:926 routeList.size() = 6
23/08/2012 07:38:40:928 routeIndex = -1
23/08/2012 07:38:40:932 routeIndex = 4
23/08/2012 07:38:51:575 SmartRoute Service Stopped
23/08/2012 07:43:34:252 Lat:004.325400#Lon:101.139772#Tim:07-43-22
23/08/2012 07:45:56:984 SmartRoute Service Stopped
23/08/2012 07:46:06:261 Location service connected
23/08/2012 07:46:09:368 Lat:004.325400#Lon:101.139772#Tim:07-45-57
23/08/2012 07:46:11:491 doSmartRoute()
23/08/2012 07:46:11:514 routeList.size() = 6
23/08/2012 07:46:11:520 routeList.size() = 6
23/08/2012 07:46:11:523 routeList.size() = 6
23/08/2012 07:46:11:530 routeList.size() = 6
23/08/2012 07:46:11:592 routeList.size() = 6
23/08/2012 07:46:11:595 routeList.size() = 6
23/08/2012 07:46:11:601 routeIndex = -1
23/08/2012 07:46:11:605 routeIndex = 4
23/08/2012 07:46:23:931 SmartRoute Service Stopped
23/08/2012 07:49:40:364 SmartRoute Service Stopped
23/08/2012 07:49:57:725 Location service connected
23/08/2012 07:49:59:495 Lat:004.325400#Lon:101.139772#Tim:07-49-47
23/08/2012 07:50:00:739 doSmartRoute()
23/08/2012 07:50:00:743 routeList.size() = 6
23/08/2012 07:50:00:749 14 4.326106 101.142969
23/08/2012 07:50:00:756 14 4.326106 101.142969
23/08/2012 07:50:00:759 14 4.326106 101.142969
23/08/2012 07:50:00:762 14 4.326106 101.142969
23/08/2012 07:50:00:766 14 4.326106 101.142969
23/08/2012 07:50:00:770 34 4.325015 101.14254
23/08/2012 07:50:00:772 routeIndex = -1
23/08/2012 07:50:00:776 routeIndex = 4
23/08/2012 07:50:06:071 SmartRoute Service Stopped
23/08/2012 07:54:07:466 SmartRoute Service Stopped
23/08/2012 07:54:17:581 Location service connected
23/08/2012 07:54:19:694 Lat:004.325400#Lon:101.139772#Tim:07-54-07
23/08/2012 07:54:21:649 doSmartRoute()
23/08/2012 07:54:21:651 currRouteIdx = 0
23/08/2012 07:54:21:657 routeList.size() = 6
23/08/2012 07:54:21:661 14 4.326106 101.142969
23/08/2012 07:54:21:664 14 4.326106 101.142969
23/08/2012 07:54:21:666 routeList.size() = 6
```

Figure 37 - Sample SmartRoute.log

Full log file is available in the source code disc.

APPENDICE C – BI-WEEKLY LOG
FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 **Study week** 1
no.:

Student Name & ID: Daniel Khor Say Hou & 09ACB03739

Supervisor: Mr.Goh Hock Guan

Project Title: GPS Traffic Navigation System

1. WORK DONE

- Experimenting with Android SDK and Waze sourcecode
- Exploring multicast in C and Java for the dummy server

2. WORK TO BE DONE

- Search for different open source project
- Write a multicast program for the dummy server

3. PROBLEMS ENCOUNTERED

- Waze successfully compile but cannot execute
- Still figuring out how to write a multicast program

4. SELF EVALUATION OF THE PROGRESS

- Not much of a progress with the Waze cannot run

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 **Study week** 2
no.:

Student Name & ID: Daniel Khor Say Hou & 09ACB03739

Supervisor: Mr.Goh Hock Guan

Project Title: GPS Traffic Navigation System

1. WORK DONE

- Written a Java dummy server by using broadcast instead of multicast
- Found another open source project, Androzic.
- Learning the basic of Android, the first “Hello World” app installed in AVD
- Successfully compile and run Androzic.

2. WORK TO BE DONE

- Further understand Android to enhance my knowledge in Android in order to debug
- Understands the open source project source code

3. PROBLEMS ENCOUNTERED

- Windows 7 unable to perform broadcast, only Windows XP. Not tested in Linux environment yet.
- Androzic always crash whenever tried to load file, however, nothing crash when uses online map.

4. SELF EVALUATION OF THE PROGRESS

- Progressing slowly

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 **Study week no.:** 4
Student Name & ID: Daniel Khor Say Hou & 09ACB03739
Supervisor: Mr.Goh Hock Guan
Project Title: GPS Traffic Navigation System

1. WORK DONE

- Further understand Android
- Done few prototype in Wi-Fi based application
- Done 1 prototype in SQLite based application
- Installed Cygwin and NDK, established the additional builder in Androzic
- Manage to get Kampar map from Google Map.
- Installed OziExplorer as suggested by Androzic FAQ.

2. WORK TO BE DONE

- Understanding Androzic
- Write few more prototype for testing the functions

3. PROBLEMS ENCOUNTERED

- Android is hard to program as unlike Java/C, debugging it require the application to be installed and running.
- Always crash whenever tried to display information on the GUI
- Very hard to understand the flow for Androzic.
- Took some time to realize this Androzic require NDK to compile but it can still compile despite having such error.

4. SELF EVALUATION OF THE PROGRESS

- Progressing relatively slow due to lack of understanding in Android

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 **Study week no.:** 6
Student Name & ID: Daniel Khor Say Hou & 09ACB03739
Supervisor: Mr.Goh Hock Guan
Project Title: GPS Traffic Navigation System

1. WORK DONE

- Android Wi-Fi and UDP function prototypes tested to be successful
- Uses simple wireless network connection
- Plotted simple waypoints around Kampar.

2. WORK TO BE DONE

- Understanding Androzic!!!!

3. PROBLEMS ENCOUNTERED

- Still don't get the Androzic how it works. Apparently, the map view for this app is self-written by the original developer. And found that there are interface file with the services for Androzic but no idea how it works or functions.
- No idea how to add route in OziExplorer

4. SELF EVALUATION OF THE PROGRESS

- Stuck at Androzic.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 **Study week no.:** 8
Student Name & ID: Daniel Khor Say Hou & 09ACB03739
Supervisor: Mr.Goh Hock Guan
Project Title: GPS Traffic Navigation System

1. WORK DONE

- Email to the developer of Androzic. In hope to get some help from the developer about the Androzic flow and each of the Activity and Service files.

2. WORK TO BE DONE

- Understanding it

3. PROBLEMS ENCOUNTERED

- Can't understand Androzic much.
- No idea how to use Android debugging tools in eclipse.
- Spent time on current semester 2 subject assignment.

4. SELF EVALUATION OF THE PROGRESS

- Stagnant at Androzic, unable to progress.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 Study week no.: 10

Student Name & ID: Daniel Khor Say Hou & 09ACB03739

Supervisor: Mr.Goh Hock Guan

Project Title: GPS Traffic Navigation System

1. WORK DONE

- Reverse engineer Androzic into UML flowchart
- Trial and error and somehow manage add some testing message to pop up
- Manage to add a "UTAR logo" button at the side pop-up of Androzic
- Start working on OpenWrt with the Linksys routers and CeedTec routers

2. WORK TO BE DONE

- Integrate the prototype tested few weeks ago into it
- Need to get the routers into WDS

3. PROBLEMS ENCOUNTERED

- UTAR logo can't change color even when tried using the Androzic original images. It should change upon press to activate or deactivate the function bind to it.
- The integrated function is not working properly. Always crash.
- Having problems to configure OpenWrt in WDS mode. Doesn't seem to function.
- Androzic developer have no reply.

4. SELF EVALUATION OF THE PROGRESS

- Forcing it to move a bit but still far from completion. Just started to work on the wireless backbone.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 **Study week no.:** 12
Student Name & ID: Daniel Khor Say Hou & 09ACB03739
Supervisor: Mr.Goh Hock Guan
Project Title: GPS Traffic Navigation System

1. WORK DONE

- Prototype a background service with the UDP listener
- Integrate the background service prototype into Androzic, SmartRoute.class
- SmartRoute is configure to auto connect to the Wi-Fi AP, with the SSID : utarfyp
- SmartRoute is able to receive message broadcasted and display an alert dialog box. The message is broadcasted in the Wi-Fi network.
- SmartRoute is able to pass message to NavigationService to switch to another route.
- Scrap WDS idea, uses Mesh network idea instead.
- Manage to deploy mesh network and all routers are able to communicate with each other.

2. WORK TO BE DONE

- Writing a report, designing poster and prepare presentation

3. PROBLEMS ENCOUTERED

- Passing message between services in Android is hard by using handler. Instead of using that, I uses sendBroadcast and broadcastReceiver with Intent-filter to do the work.
- The packet receive by SmartRoute are all rubbish for many times until found out somehow it was unable to convert bytes array into string implicitly as tested in prototype stage. It requires explicit conversion of byte arrays into string.

- WDS wasn't able to deploy with unknown configuration as not much of documentation on it. Cause of Linksys routers uses Broadcom chipset while CeedTec uses Atheros chipset
- The wireless mesh network, CeedTec routers having a lot of link quality issue with average less than 50%. High packet drop rate once the CeedTec wireless routers are 1 table apart!

4. SELF EVALUATION OF THE PROGRESS

- Somehow, manage to finish all the objectives.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I)

Trimester, Year: 1, 3 **Study week no.:** 14
Student Name & ID: Daniel Khor Say Hou & 09ACB03739
Supervisor: Mr.Goh Hock Guan
Project Title: GPS Traffic Navigation System

1. WORK DONE

- Hand in the report
- Done presenting and live demo.
- Hand in revised report

2. WORK TO BE DONE

- Prepare for this semester final exams
- Prepare for competition if successful entered by supervisor

3. PROBLEMS ENCOUNTERED

- On the day of presentation, the Androzic cant receive message like usual in debugging mode. Found out, it was the debugging out message priority taken over the UDP listener as the update constantly on the GPS location updated.

4. SELF EVALUATION OF THE PROGRESS

- Done everything I could for achieving the objectives.
- This is just stage 1 as discussed. Stage 2 would need additional time to work on it. Probably for the next take over or I do it myself over my free time.

Supervisor's signature

Student's signature