

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

# **Industrial Wireless Network Mapping**

A dissertation submitted by

**Glenn Lumsden**

In fulfilment of requirements of

**Bachelor of Computer Systems Engineering**

**October 2015**

## Abstract

Wireless network mapping is the process of displaying quality and performance information related to wireless access points over a relational map. As business systems utilising wireless technologies demand sufficient coverage and performance of their networks under all conditions, it is important to continue to study new methods to monitor wireless networks to detect emerging deficiencies to reduce negative business impacts. The project aims to develop a system that utilises client infrastructure to capture and forward wireless network quality with global positioning system (GPS) coordinates for use in open cut mining activities.

Whilst commercial Wi-Fi mapping software exists, some use information from the access points (AP) to estimate the coverage and other quality factors. Whilst this can be live information it cannot provide suitable information about the network as seen by clients on the network. Others methods employ a technique known as war-driving that can map GPS and Wi-Fi data at a client level but only producing a snap shot in time. This project is developing software allowing clients on a wireless network to report Wi-Fi and GPS information back to a centrally located server for live analysis and reporting.

The project employed researching existing techniques and coding for gathering GPS telemetry and Wi-Fi data including data storage methods. Research provided access to code and methods that have been tested and critiqued. The system has been conceptualised and alternative programming languages and database programs compared. A trial system has been designed and tested to retrieve available Wi-Fi information with GPS telemetry, storing the data in a local database to be installed on existing computing hardware on mining vehicles. A remote/central database has been developed and tested in conjunction with the client application to receive the information buffered on the network clients.

**University of Southern Queensland**

**Faculty of Health, Engineering and Sciences**

**ENG4111/ENG4112 Research Project**

## **Limitations of Use**

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

**University of Southern Queensland**

**Faculty of Health, Engineering and Sciences**

**ENG4111/ENG4112 Research Project**

## **Certification of Dissertation**

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

**Glenn Lumsden**

**Student Number: 0019322995**



---

**Signature of Candidate**

28 October 2015

**Date**

## **Acknowledgments**

The author would like to sincerely thank the following people for their help and support during the project:-

My wife Jenn and my three children, for their support, patience and understanding during the completion of the project. I am looking forward to more quality time after so many busy years studying.

My work and study colleague, Jason Lang, whom assisted me greatly during the project development and final years of study.

Dr Alexander Kist, for his enthusiasm and guidance throughout the completion of the project.

## Table of Contents

ABSTRACT.....	I
LIMITATIONS OF USE .....	II
CERTIFICATION OF DISSERTATION .....	III
ACKNOWLEDGMENTS .....	IV
LIST OF FIGURES .....	IX
LIST OF TABLES .....	X
LIST OF APPENDICES .....	XI
NOMENCLATURE AND ACRONYMS .....	XII
CHAPTER 1 INTRODUCTION .....	1
1.1 Outline of the Study .....	1
1.2 Introduction.....	1
1.3 Problem Statement .....	2
1.4 Research Aim and Objectives .....	2
1.5 Conclusions.....	3
CHAPTER 2 BACKGROUND .....	4
2.1 Introduction.....	4
2.2 Open Cut Coal Mining.....	4
2.3 Fleet Management System.....	5
2.4 High Precision GPS .....	6
2.5 Meandu Mine .....	7
2.6 Conclusion .....	8
CHAPTER 3 LITERATURE REVIEW .....	9
3.1 Introduction.....	9
3.2 Networking for Mining .....	9
3.3 Wireless Network Challenges .....	10

3.4	Wireless Design and Monitoring .....	11
3.4.1	Modelling and Survey .....	11
3.4.2	Sensor Based Monitoring.....	13
3.4.3	Crowdsourcing.....	14
3.5	Global Navigation Satellite System .....	16
3.5.1	Data Protocols .....	17
3.6	Conclusion .....	19
CHAPTER 4 METHODOLOGY .....		20
4.1	Introduction.....	20
4.2	Philosophy.....	20
4.3	Requirements Engineering.....	20
4.3.1	Functional Requirements .....	21
4.3.2	Non-functional Requirements .....	21
4.4	System Modelling .....	22
4.4.1	Waterfall Analysis.....	22
4.4.2	Incremental Development .....	23
4.4.3	Re-use Oriented Development .....	25
4.5	Software Strategy.....	25
4.5.1	Constraints .....	26
4.5.2	IDE.....	26
4.5.3	Simulation .....	26
4.5.4	Database.....	27
4.5.5	Web Server.....	27
4.5.6	Programming Languages .....	27
4.6	Client Design Overview.....	28
4.7	Software Testing .....	32
4.7.1	GPS Testing .....	32
4.7.2	Wi-Fi Testing .....	32
4.7.3	Database Testing .....	33
4.7.4	Transmission Testing .....	33
4.8	Conclusion .....	33
CHAPTER 5 DETAILED DESIGN.....		34
5.1	Introduction.....	34

5.2	Collection User Interface .....	34
5.2.1	Main Menu.....	34
5.2.2	Form Interface.....	36
5.2.3	Systems Health.....	39
5.3	Transmit User Interface .....	41
5.3.1	Main Menu.....	41
5.3.2	Form Interface.....	42
5.3.3	System Health .....	43
5.4	Coding Overview .....	44
5.4.1	Client Overview .....	44
5.4.2	Transmit Overview .....	45
5.5	GPS Interface Coding .....	46
5.6	Wi-Fi Interface Coding .....	46
5.6.1	ManagedWiFi .....	46
5.7	MySQL .....	47
5.8	Graphical User Interface .....	47
5.9	Conclusion .....	47
CHAPTER 6 ANALYSIS AND PERFORMANCE .....		48
6.1	Introduction.....	48
6.2	Analysis Section.....	48
6.2.1	Normal Operation .....	49
6.2.2	Data Buffering .....	50
6.2.3	Performance .....	52
6.3	Conclusion .....	53
CHAPTER 7 CONCLUSION.....		54
7.1	Introduction.....	54
7.2	Achievement of Objectives.....	54
7.3	Further Work.....	55
LIST OF REFERENCES .....		56
BIBLIOGRAPHY.....		59
APPENDIX A: PROJECT SPECIFICATION.....		60



APPENDIX B: WENCOMINE FLEET MANAGEMENT .....62

APPENDIX C: CODE DEVELOPED IN C# .....65

APPENDIX D: CODE DEVELOPED IN PHP .....84

APPENDIX E: MYSQL CONFIGURATION .....89

## List of Figures

Figure 1 - Mining Topology Challenges.....	8
Figure 2 - 2.4 Gigahertz Channel Plan (Gunther, 2015).....	14
Figure 3 - Channel Crossover Interference.....	15
Figure 4 - Waterfall Model.....	23
Figure 5 - Incremental Development.....	24
Figure 6 - Reuse-oriented Development.....	25
Figure 7 - Client Overview Data Flow.....	29
Figure 8 - Client Data Capture.....	30
Figure 9 - Transmit Application.....	31
Figure 10 - Client Interface 1.....	35
Figure 11 -Basic GPS Info.....	36
Figure 12 - NMEA Stream.....	37
Figure 13 - WiFi Data.....	38
Figure 14 - COM Port Error.....	39
Figure 15 - GPS Health Good vs Bad.....	40
Figure 16 - COM Port Disconnect.....	40
Figure 17 - Transmit Application.....	42
Figure 18 - Transfer Failed!.....	44
Figure 19 - VirtualBox.....	48
Figure 20 - GPS Mount.....	49
Figure 21 - Application Testing.....	50
Figure 22 - Initial Display Design.....	51
Figure 23 - Raw data XML.....	52
Figure 24 - Minevision.....	63
Figure 25 - Out of Network Range.....	63
Figure 26 - Fleet Control.....	64

## List of Tables

Table 1 - GLL Data Set.....	18
Table 2 - ZDA Data Set .....	18

## List of Appendices

APPENDIX A:	PROJECT SPECIFICATION.....	60
APPENDIX B:	WENCOMINE FLEET MANAGEMENT .....	62
APPENDIX B1:	MINEVISION .....	63
APPENDIX B2:	FLEET CONTROL .....	64
APPENDIX C:	CODE DEVELOPED IN C#.....	65
APPENDIX C1:	PROGRAM.CS.....	66
APPENDIX C2:	CLIENT.CS.....	67
APPENDIX C3:	TRANSMIT.CS.....	76
APPENDIX C4:	MYSQLDB.CS.....	79
APPENDIX C5:	CONFIGURATION FILE.....	83
APPENDIX D:	CODE DEVELOPED IN PHP .....	84
APPENDIX D1:	GPSDATA1.PHP .....	85
APPENDIX D2:	PHPSQLAJAX_GENXML.PHP.....	87
APPENDIX D3:	PHPSQLAJAX_DBINFO.PHP .....	88
APPENDIX E:	MYSQL CONFIGURATION .....	89
APPENDIX E1:	MYSQL SERVER CONFIGURATION .....	90
APPENDIX E2:	LOCAL DATABASE DESIGN .....	97
APPENDIX E3:	REMOTE DATABASE DESIGN.....	98

## Nomenclature and Acronyms

AP	Access point
BSSID	Basic service set identifier
COM Port	Serial communication port
FMS	Fleet management system
GLONASS	Global Navigation Satellite System - Russian
GNSS	Global Navigation Satellite System
GPS	Global positioning system
IDE	Integrated development environment
IEEE	Institute of Electrical and Electronic Engineers
LAN	Local area network
MAC	Media access control
NAVSTAR-GPS	Navigation system with timing and ranging global positioning system
NMEA	Nation Marine Electronics Association
OBD	On-board diagnostics
PtP	Point to point
RSSI	Received signal strength indication
RTCM	Radio technical commission for marine services
RTK	Real time kinematic
SSID	Service set identifier
TTL	Transistor-transistor logic
USB	Universal serial bus
VHD	Virtual hard disk
Wi-Fi	Wireless fidelity
WLAN	Wireless local area network
WMN	Wireless mesh network

# Chapter 1

## Introduction

### 1.1 Outline of the Study

The need to investigate new methods in monitoring wireless networks was identified from experienced gained whilst working with wireless networking for open cut mining. The processes used to determine the health and effectiveness of the network did not provide information relating directly to the mobile clients operating in the field leading to the investigation to use the mining fleet computing system to assist in reporting network health information.

### 1.2 Introduction

Wireless networks are now common place, domestically, commercially and industrially. For the commercial and industrial implementations the importance for network coverage and performance is becoming a high priority for business systems leveraging the technologies. The introduction of the IEEE 802.11 standard has provided manufactures the ability to produce a standard suite of products and systems now utilised by these businesses; specifically to this study in open cut mining applications.

For many businesses the wireless network is required to perform over a fixed topology but for mining applications the site is continually changing and tends to cover large areas. In this case present form Meandu Mine the site covers approximately twenty five square kilometres. The ability to simulate the coverage requires up-to-date modelling information not always available and survey work takes time, only producing a snap-shot in time.

By utilising the client computing, GPS receivers and wireless network adaptors on mining fleet vehicles to report live information about the wireless network coverage based on GPS location, an alternative method is tested to improve wireless network monitoring.

### **1.3 Problem Statement**

With a continually changing topology, large distances and depths involved with open cut mining, the network infrastructure, required to support the mobile fleet operation, cannot monitor or report the areas affected, as seen at the client interface.

### **1.4 Research Aim and Objectives**

The aim of this project is to develop a system that can utilise mobile fleet, computing, GPS and wireless client infrastructure to report network coverage based on GPS telemetry for early detection of problem areas.

There are two main objectives for this project and each section requires the creation and integration of software systems:

- **Data Collection** - Continuously sample available Wi-Fi data tagged with GPS telemetry, including capturing vehicle location during loss of network connection.
- **Data Storage** - Server/client arrangement to store all the incoming client information in a suitable database at a central location for live results analysis.

## **1.5 Conclusions**

This dissertation aims to determine a suitable process to monitor wireless network performance based on client information. The research resulted in the development of software that is suitable to report wireless network and GPS information, collected at each client on the network.

A review of the literature for this research identified alternative methods to analyse Wi-Fi systems and suitable processes that can be employed to achieve the aim. The outcome of this study produced a design and preliminary development of software that can integrate with existing computing, wireless and GPS hardware for the Meandu Mine mobile fleet infrastructure.



## **Chapter 2 Background**

### **2.1 Introduction**

This chapter discusses the background behind the projects aims. The background information is drawn from personal experience by the author with over 20 years of experience working in the power generation and mining industries. The study has focused on open cut mining operations at Meandu Mine.

### **2.2 Open Cut Coal Mining**

The general requirement of an open cut coal mine is to relocate unwanted material, such as topsoil, rock and other overburden to uncover the seams of coal beneath. The coal is then extracted and processed to meet quality metrics, such as moisture, ash content and the energy per tonne of final product. The excavation process first involves removing and storing the top soil; for later land rehabilitation. The layers between the topsoil and coal seam are then relocated through a cycle of fracturing through drilling and blasting followed by excavation. The excavation is usually performed with combinations of excavators, shovels, rear dump trucks, bulldozers and draglines. As an open cut mine progresses along the seams of coal, the areas now depleted of the useful minerals can be rehabilitated. Rehabilitation requires bulk waste material from the mining activities to fill the mine voids and built up to a profile deemed appropriate for the area. The profile is then capped with the stored top soil and revegetation activities occur. In order to minimise erosion of the new deposited top soil the ground profile, in particular slope, require precise design and construction. It is the use of high precision GPS (refer 2.4) on board machinery such as bulldozer and graders that ensure the profiling meets the plan as designed.

The cost of the final product represented as dollars per tonne or possibly dollars per energy unit. Either way, one primary business objective is to produce the final product at the lowest cost. The costs are measure through wages, fuel and maintenance activities versus the amount of final product delivered or the total energy delivered. In the case for Meandu mine the figures are measured as dollars per gigajoule. The commercial goals for the business include meeting delivery targets, at the required quality, for the lowest cost.

### **2.3 Fleet Management System**

In order to evaluate the cost of the final product, the running cost of the mine and the amount of final delivered product must be sourced. The modern mine has incorporated a fleet management system as one of the tools to capture the production information from the mining fleet (refer Appendix B). The production information recorded varies depending on the vehicle in the fleet. The fleet for Meandu consists of excavators, rear dump trucks, bulldozers, scrapers, graders, water carts and a dragline. The production rates of the excavators ranges from 1000 tonne per hour to 2500 tonne per hour. The variability is dependent on the size of the excavator, the number and size of the dump trucks, the distance from working face to dump location and the density of material being moved.

A fleet management system (FMS) provides information collected from the mining fleet back to a central location to measure fleet performance against the expected metrics, provides the fleet operators with a user interface to track their operational status and operation plan. The primary information collected from the dump trucks is a count of the number of loads of material, the weight of each load and the time taken to complete a complete cycle from dump to dump. This provides the key information to determine the production rates for each dump truck.

In order to capture this information each vehicle in the fleet must have the available sensors to measure the physical information connected to the on-board diagnostic (OBD) processor, usually unique to the manufacturer of the equipment. The information available in the OBD is then connected

to the fleet management hardware; in the case for Meandu mine each vehicle is fitted with a ruggedized, on-board computer. The computer operates on a Windows XP embedded operating system running the FMS software Wencomine, from Wenco. The FMS computer is interconnected to the proprietary OBD systems for each vehicle to extract the raw data, as well as extra technology including wireless networking and GPS receivers.

## **2.4 High Precision GPS**

Another function of the FMS is to support high precision GPS operations. High precision GPS requires a time correction signal to be sent from a nearby base station to the mobile equipment. The corrections made increase the GPS accuracy to a few centimetres. At Meandu Mine the high precision GPS system use the real time kinematic (RTK) signal to provide the necessary correction information to bring the accuracy of the GPS to within a few centimetres.

Equipment and activities now requiring the use of high precision GPS include:

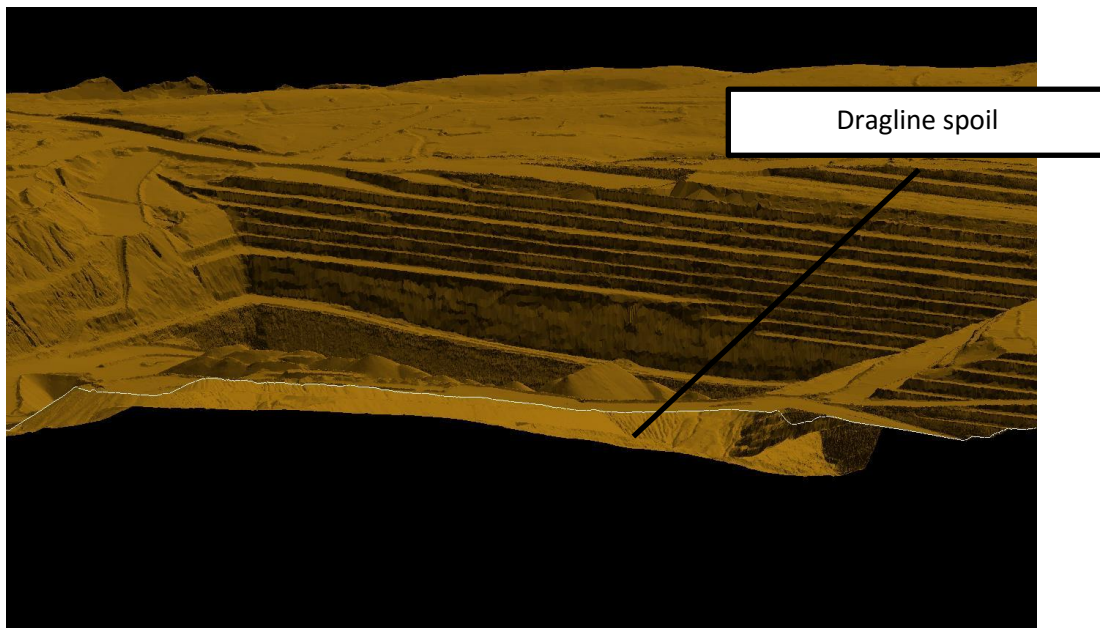
- Rehabilitation by bulldozers and graders for correct sloping to minimise the erosion during early stages of revegetation.
- Excavating equipment removing overburden or coal follow the mine plan set out by the mining engineers.
- Drilling and blasting requires precise location and depths of holes required as per the blast plan. The volume of the explosive products required for blasting the overburden and raw coal over large areas means the use of explosives must be managed effectively to minimise losses.

Rework is a high cost for mines when machinery is required to readdress an area to meet the mine plan. The cost of running the large machines along with wages can be thousands of dollars per hour. So the time and areas during which this equipment is not accessing the GPS correction data the mine can suffer significant losses reworking those areas.

## 2.5 Meandu Mine

The wireless local area network (WLAN) at Meandu utilises the IEEE 802.11 a/b/g/n/s standards for wireless networking operating on the 2.4 gigahertz (IEEE 802.11b/g/n) and 5 gigahertz (IEEE 802.11a) unlicensed frequency bands. The specific design of the network is that of a wireless mesh network (WMN). It is not the intention of this paper to review the operating principles of wireless mesh networking as regardless of the wireless technology implemented, the primary issue related to this study, is maintaining reliable network coverage for all mining activities requiring access to a wireless network. This reliability issue stems from the mining activities dramatically altering the topology of the landscape. These changes come about through excavator and dump truck operations relocating thousands of tonnes of material per day. This creates changes to areas around the working face, where the excavator is operating, also the dump location is receiving this material which may be either filling a void or creating a new dump. The dumps can reach hundreds of meters in height blocking network links and creating shadows with network transmission. A second source to the change in topology is with dragline activity. Draglines remove large quantities of overburden at several thousand tonnes per hour. This material is dumped within the span of the dragline boom thus the topological changes are localised.

An example of topology challenges is where over the course of a few days a dragline has raised a spoil pile by over 50 meters between itself and the nearest access point (refer Figure 1) causing network interruption to the dragline and other vehicles operating at the base of this pit, which is over 200 meters deep.



**Figure 1 - Mining Topology Challenges**

With the implementation of wireless networking in open-cut mining operations the constantly changing terrain, vehicle location and environmental conditions can lead to poor or no coverage in operational areas, causing delays or loss of vital information resulting in costly rework.

## **2.6 Conclusion**

Overviews of mining operations and equipment that utilise wireless network infrastructure have been discussed, as well as the importance of the network supporting these operations. There can be large losses in operational efficiency when the network is not able to provide coverage to areas required, costing the business time and money. Whilst the network may have been deployed and tested to ensure effective coverage initially, the nature of open cut mining causes the topology to change rapidly potentially creating areas affecting network coverage. Investigation into methods capable of supporting the business to analyse the network coverage utilising the client hardware available can provide an additional tool to determine areas affected.

## **Chapter 3**

# **Literature Review**

### **3.1 Introduction**

The literature review supports the background information relating to the use of wireless networking in mining operations. The different techniques available to support wireless network monitoring and development, relating to network coverage, are compared to the problem statement and the proposed solution.

### **3.2 Networking for Mining**

As explained by Vellingiri et al. (2013) one of the primary reasons for the installation of a WLAN in mining is for remote monitoring and diagnosis. A fleet management system allows mining operations to be continuously optimised through real-time monitoring, assignment of fleet resources, monitoring production information, and telemetry and machine health.

Another advance in efficient mining operations has been the implementation of high precision GPS. High precision GPS corrections are sent to equipment such as drill and blast equipment, bulldozers, graders and draglines for critical operation that require accuracy of a few centimetres. This is achieved through the use of a base station situated in a fixed location with a known GPS reference.

Some other uses for a WLAN are real-time video surveillance for monitoring key infrastructure, such as explosive facilities and monitoring fleet operation for feedback in training. Safety system monitoring is also becoming increasingly reliant on WLAN infrastructure, such as man-down systems, geological monitoring, and collision avoidance and tracking systems. Maintenance systems now also integrate with data from boom stress monitoring, fuel tracking, and tire pressure monitoring and machine health. In some cases there is also dust suppression and environmental monitoring

Information system access so personnel in the field can access site specific and corporate information systems.

As more advanced communication systems become integrated into mining activities, the demand for these systems to function with a high reliability also increases. Vellingiri et al. (2013) addressed that business systems for production and maintenance planning as well as monitoring of mining assets can be adversely affected. Monitoring of the assets can lead to increased operational efficiencies by analysing the health of the machines and the operational behaviours of operators. This can then lead to improved workflow implementation through operator training.

Reliable networking is becoming increasingly important for tracking systems for mining. Kloos et al. (2004), Novak et al. (2010) and Jun et al. (2010) all discuss the methods used by tracking systems for mine safety. To determine the location of equipment underground, without the ability for GPS tracking, the distance to APs is calculated using various techniques and so becomes reliant on the quality of the network coverage. Novak et al (2010) recommends communications for emergency systems and post-accident recordings also demand reliable and effective network coverage and that the implementation of wireless mesh technology can meet those needs, due to the systems inherent redundancy and ease of extensibility. They however did highlight these systems are becoming increasingly complex and is still a relatively new technology.

### **3.3 Wireless Network Challenges**

Deploying a wide-scale wireless network is not an exact science. There are many factors to account for in the design phase such as the performance requirements, topology, interferences, security and cost. For fixed infrastructure, such as offices, warehousing or university campuses the design for the given topology tends to remain constant for extended periods. Some of the factors that can change rapidly for these installations are interference due to other networks extending past the boundary of the network, the loading on a given access point due to increased users at a particular location or a general increase in users across the entire network (Sheth, 2007).

In the case of an open cut mining environment a primary difference to fixed infrastructure installations is the continually changing landscape. The initial design and deployment of the access points and backhaul network may have been suitable to meet the performance criteria initially but in some cases, within days, the coverage and thus performance of localised areas can be adversely affected. Operational personnel are generally not trained to fault find or analyse the network performance, thus a problem may only be reported after some time and the detail of the event for further investigation not captured.

Analysing network performance can be a time and resourcing consuming exercise, particularly in an open cut mine where distances can be upwards of 20 kilometres from maintenance workshops. The network infrastructure needs to be reviewed for faults and performance issues; localised inspection at the problem site is surveyed, the machine/s reporting the issue inspected for damage. If the root problem is the location of access points, the network may require additional nodes added to the system or infrastructure relocated. Relocation of network hardware needs to be critically analysed to ensure further network issues are not encountered due to the relocation.

### **3.4 Wireless Design and Monitoring**

There are several phases to a wireless network lifecycle. The system must first be planned based on the requirements specification.

#### **3.4.1 Modelling and Survey**

There are two mainstream methods in designing a WLAN; site survey using empirical measurements and planning software using propagation modelling. Zvenovec. et al. (2003) explore the advantages and disadvantages of these two methods. The testing and modelling were both for indoor office environments and showed that reasonable predictions of the effects of walls, corridors and openings can be simulated with reasonable confidence. Through their study they conclude that planning software has many advantages over site surveys. In the case here of open cut mining, the distances and sometimes complex topology would lend itself to software planning. Manually taking



measurements across even a small mine or quarry would take significant time and only be valid for a short period. Their study was focused on the comparison of the two techniques but did consider the use of both systems where a modelled design could be deployed and a site survey used to test and or improve the models design, perhaps for a sub-set of the area.

In this manner, a report on the statistical validation of WLAN range calculated with propagation models for industrial environments by chipset-level received signal strength measurements was undertaken by Tanghe et al. (2008). The study was again limited to indoor industrial environments such as warehouses and factories. The issue concerning the validation of modelling compared to site survey was addressed with a key finding being that the results between the two methods were statistically in agreement. They did note that the physical environment did cause some deviation between the two methods, notably due to large obstacles causing severe attenuation degradation in local area. Similar to a mining environment, the general topology of the site would remain unchanged for long periods of time but local areas are radically modified with large earth moving equipment removing or depositing material.

Open cut and underground mining both have similar challenges in design, implementation and monitoring of any form of wireless network. Both environments are constantly changing and line-of-sight is regularly affected. A survey of wireless network methods was researched by Forooshani et al. (2013) on underground mining. They found that the mesh network was the most reliable due to the redundant, self-learning and self-healing fundamentals of the technology. Their preference was for empirical measurements via survey as the results are definitive but with the limitations of physically accessing all areas required for survey this approach is becoming less common.

Where a network may be required for emergency communications, more so for underground mining, Moutairou et al.(2006) develop a generic algorithm for a backhaul network design. The link was based on a mesh network operating in the 5 Gigahertz spectrum. Backhaul design is not concerned with coverage but point to point (PtP) links, ensuring each node has adequate throughput available to other nodes in the network and ultimately back to the wired LAN. Should a backhaul system be

designed by modelling then installed, the system is easy to test as all the nodes on the link should be active on the network. Unfortunately their study did not compare their results to a physical system and they do not investigate the potential to simulate the access layer.

These studies highlight that survey and modelling have their advantages and disadvantages. The studies above all prefer the practical advantage of design by simulation and modelling but this does not provide empirical data to ensure an effective and efficient network.

### **3.4.2 Sensor Based Monitoring**

An alternative method to perform live data analysis of wireless network coverage would be to deploy another network of sensors that sense the wireless network. One such system as developed by Zhou et al. (2013), called WizNet, utilises a network of ZigBee-based sensors. ZigBee is a wireless mesh network standard built on the IEEE 802.15.4 standard. The concept is to use the ZigBee modules to listen in to the shared frequency spectrum that 802.11 and 802.15 both operate within. As the ZigBee modules are low cost, low power devices they can be cheaply deployed into areas that require continuous monitoring. Where the studies of Zvenovec et al. (2003), Tanghe et al.(2008) and Moutairou et al.(2006) all see the use of modelling as a suitable approach to determine network penetration, Zhou et al.(2013) believe these methods do not produce the granularity and are location constrained by using the existing network infrastructure. Granularity is becoming essential for wireless networking in mining and so having a system that operates outside the WLAN has its own strengths. Some of the major disadvantages for an open cut mining operation to utilise a system such as WizNet are the distances required for the low power devices to transmit. To be effective, the devices may need to be mounted directly on the mining fleet and so the same problem can occur that the WizNet mesh network does not have suitable penetration for its own mesh system to adequately monitor the WLAN.

### 3.4.3 Crowdsourcing

Crowdsourcing can be defined as the practice of obtaining needed services, ideas, or content by soliciting contributions from a large group of people and especially from the online community rather than from traditional employees or suppliers (merriam-webster.com, 2015).

From my review of others works there is a reasonable degree of interest and development in the use of crowdsourcing for wireless network monitoring. With the use of mobile device capable of collecting network information that can be sent to central location prevalent on most WLANs, as is the case for the mining fleet at Meandu mine, is readily achievable. Frangoudis et al. (2015) presented results showing a significant performance improvement using crowdsourcing on an indoor WLAN. Their study was limited to indoor wireless, as too was the work on Pazl by V.Radu et al.(2013), MCNet by Rosen et al. (2014) and R.K. Ganti et al. (2010).

Besides wireless network coverage another area of interest for P.A. Frangoudis et al. (2015) was cross-channel interference. In Australia the 2.4 gigahertz IEEE 802.11b/g unlicensed spectrum is divided into 14 channels. Each channels bandwidth extends over its neighbour. As shown in Figure 2 the use channels 1, 6 & 11 provide a spread of frequencies that do not interact.

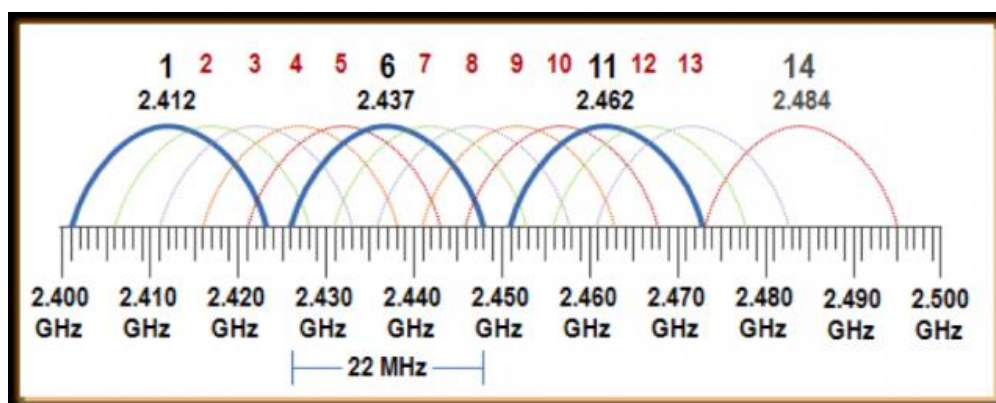
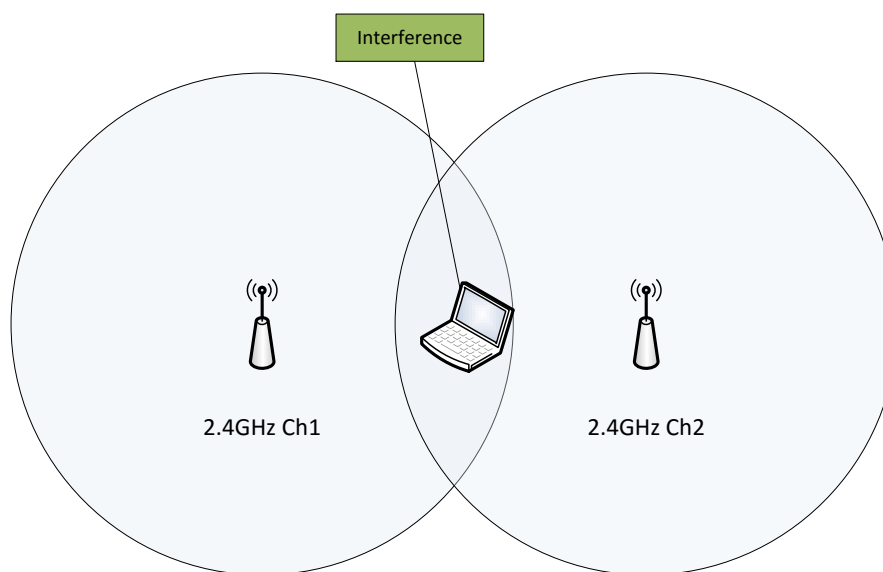


Figure 2 - 2.4 Gigahertz Channel Plan (Gunther, 2015)

A network design should ensure to minimise any channel crossover and usually each AP can observe what is nearby and report any conflicts accordingly. This is fine when the APs are within reach of one another. With the case of cross channel interference (refer Figure 3) where two APs are not within range of one another but their radiated signals do overlap at a location where a user is trying to operate, interference is experienced by the client but is not seen by the monitoring system.



**Figure 3 - Channel Crossover Interference**

A common theme amongst the literature from P.A. Frangoudis et al. (2015), V. Radu et al. (2013), Rosen et al. (2014) and R.K. Ganti et al. (2010) is that of security and trustworthiness. The security of personal information if tracking of a device is being used to map to WLAN coverage was discussed and so methodology was derived to ensure the most basic of information was made available This would be reasonable for a system that is designed to monitor untrusted networks or extended to 3G/4G mapping systems where a user's regular movements can be tracked. In the case for mining network coverage, the user is a known user on a known and trusted network. The location of the user i.e. haul truck etc. is a vital piece of information to assist in network mapping. The other concern listed was that the users could not be considered a trusted source. This was highlighted by both P.A.

Frangoudis et al. (2015) and Rosen et al. (2014) in that the user may present false data to the system and so care must be taken to validate the data. This would not be the case for the mining environment as no uncontrolled user would be connected to the network or the monitoring system.

V. Radu et al. (2013) and Rosen et al. (2014) developed systems called Pazl and MCNet, respectively, for crowdsourcing network information using mobile devices. Both studies were limited to indoor wireless networks utilising basic localisation techniques to determine the user's position. Obviously knowing a client position indoors presents its own challenges and there are many studies relating to this. The authors also compare their work to AP vantage point surveying and manual site survey showing a significant improvement in reporting fidelity. The Pazl system was not live but instead buffered the data locally and waits for an opportunity to download the data from each client. This would be a requirement for the GPS-based system during periods of no signal coverage but GPS tracking can still occur.

A limitation with these systems is the ability to track the user in areas of no network coverage. Obviously if the clients are using estimated distance to APs to determine their rough position and there are no APs, then location tracking is difficult. Both systems consider the additional use of GPS positioning for outdoor tracking but rule this out due to the power consumption required on the mobile devices. This is not a concern for vehicle mounted systems but consideration should be made if a vehicle is out of service for an extended period as battery drain will occur eventually.

### **3.5 Global Navigation Satellite System**

Satellite navigation is used extensively across the world in almost all industries, commercial and domestic applications. Employing a Global Navigation Satellite System (GNSS) provides the information required to accurately determine position and time anywhere on Earth. The fleet management system at Meandu leverages the American GPS system via U-Blox receivers.

Currently there are two operational global operational GNSSs, NAVigation System with Timing And Ranging Global Positioning System (NAVSTAR-GPS) or commonly known as GPS, operated by the United States of America and the Russian GLObal NAVigation Satellite System (GLONASS).

### 3.5.1 Data Protocols

GNSS receivers can be developed either with manufacturer (proprietary) formats and protocols, or they can utilise the two international standards (NMEA or RTCM).

The NMEA-0183 data interface, standardised by the National Marine Electronics Association, is a serial communications interface (TTL or RS232). The data is relayed according to the NMEA-0183 specification as data sets:

- GGA (GPS Fix Data, fixed data for the Global Positioning System)
- GGL (Geographic Position – Latitude/Longitude)
- GSA (GPS DOP and Active Satellites, degradation of accuracy and the number of active satellites in the Global Satellite Navigation System)
- GSV (GNSS Satellites in View, satellites in view in the Global Satellite Navigation System)
- RMC (Recommended Minimum Specific GNSS Data)
- VTG (Course over Ground and Ground Speed, horizontal course and horizontal velocity)
- ZDA (Time & Date)

The data sets of relevance are the:

- GLL data set (geographic position –latitude/longitude) contains information on latitude and longitude, time and health status of the information. Table 1 shows the deconstruction of the NMEA sentence \$GPGLL,2631.78856,S,15151.05565,E,033138.00,A,A\*7E
- ZDA data set (time and date) contains information on UTC time; the date and local time (refer Table 2).

<b>Field</b>	<b>Description</b>
\$	Start of the data set
GP	Information originating from a GNSS appliance
GLL	Data set identifier
2631.78900	Latitude: 26° 31.78900 min
S	Northerly latitude (N=north, S= south)
15151.06000	Longitude: 151° 51.06000min
E	Easterly longitude (E=east, W=west)
33138	UTC positional time: 13h 03min 05.0sec
A	Data set quality: A means valid (V= invalid)
*	Separator for the checksum
7E	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Table 1 - GLL Data Set

<b>Field</b>	<b>Description</b>
\$	Start of the data set
GP	Information originating from a GNSS appliance
ZDA	Data set identifier
035418.00	UTC time: 03h 54min 18.00sec
03	Day (00 -31)
09	Month (01-12)
2015	Year
00	Reserved for data on local time (h), not specified here
00	Reserved for data on local time (min), not specified here
*	Separator for the checksum
61	Checksum for verifying the entire data set
<CR><LF>	End of the data set

Table 2 - ZDA Data Set

### **3.6 Conclusion**

The available literature for wireless network surveying revealed that there is scope for a GPS based crowdsourcing application to be developed. Similar studies and software development has occurred for indoor Wi-Fi monitoring but these studies have not explored the option to integrate GPS telemetry in any detail. As shown in the literature, deployment of a client application for network monitoring is a cheap solution that can provide the fidelity required that modelling cannot guarantee.

The network utilisation for the mining application will have trusted and controlled clients accessing it. Therefore security concerns with tracking users movements is not an issue and is in fact a requirement for the monitoring system. The possibility for corrupt data to be sent from a client due to intentional manipulation of data is extremely low as it is in the interests of the clients to have effective coverage.



# Chapter 4

## Methodology

### 4.1 Introduction

The methodology section discusses the choice and justification of design methodology. This consists of how the system will be modelled and justification of using an incremental development approach mixed with reuse software engineering techniques. What software packages will be utilised and the programming languages chosen.

### 4.2 Philosophy

The use of open source software that is well supported and documented is a criterion during development to provide a cost effective but reliable process.

The programming languages needed to be readily available whilst utilising the strength from others works where permitted. This follows a re-use policy for code development.

Development of programs was best supported using proven modelling techniques for small development software systems.

### 4.3 Requirements Engineering

The requirements engineering process provides a framework to document the user and system requirements in order to produce the software requirements specification.

### **4.3.1 Functional Requirements**

The user requirements are:

- a) The user shall be able to pause and restart data collection
- b) The user shall be able to stop the program anytime
- c) Multiple sampling times for data collection shall be selectable by the user during operation

The system requirements are:

- a) The system requires to operate on a Windows XP Embedded operating system
- b) The system shall have a Windows form interfaces for all user interaction
- c) The system will display the current GPS and Wi-Fi information
- d) The system shall collect GPS data via a serial interface
- e) Information from the GPS data sets shall be the NMEA-0183 specification
- f) The system will record any locations with no Wi-Fi data
- g) All GPS/Wi-Fi related data will be transmitted to a central database

### **4.3.2 Non-functional Requirements**

Product requirements:

- a) The client system shall be able to buffer at least 10 minutes of sequential GPS locations at the lowest sampling rate

Organisational requirements:

- a) None

Development requirements:

- a) The system shall be developed in standard, supported languages
- b) The use of the Integrated Development Environment, Visual Studio

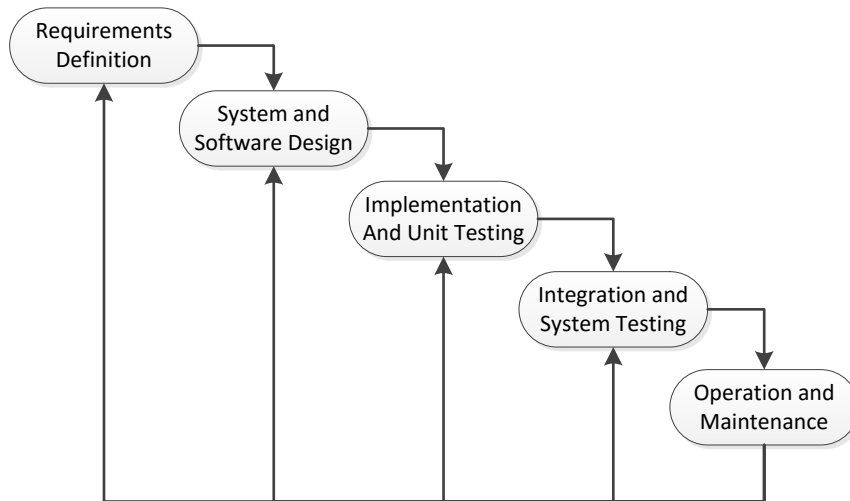
## 4.4 System Modelling

There are three main software process models that could have been used for the development of this software, as described by Sommerville (2011). Whilst not mutually exclusive, any combination can be applied to suit the size and complexity of the project. The three models are:

- Waterfall model
- Incremental development
- Reuse-oriented software engineering

### 4.4.1 Waterfall Analysis

The waterfall model requires a plan-driven process where all the process activities require a plan and schedule before starting. This approach is simple and easy to understand producing a rigid model. Management of each phase of development would provide specific deliverables and a review process for each of the processes. As each phase (refer Figure 4) is developed and completed one at a time with no overlap, the programmer can direct their attention to a single task until completed. Where the requirements are very well known the waterfall model can work well and had the potential to satisfy the size of the project.



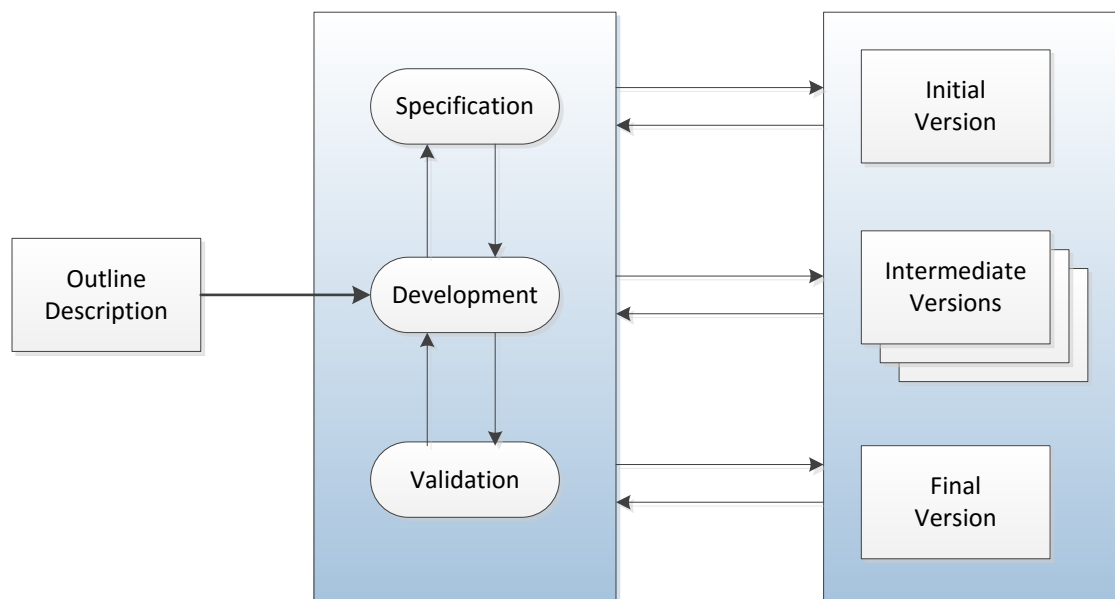
**Figure 4 - Waterfall Model**

The problem with the waterfall model is that once the application is in the testing phase, it is difficult and time consuming to introduce changes that had not been well thought out in the concept phase. Working software is not produced until late in the life cycle of this modelling technique which can present high amounts of risk and uncertainty for software success. As this project is predominately experimental there is not a defined end point in its development and it is expected that the requirements will change throughout development, which conceptually is not supported by the waterfall model. As the software development was expected to be developed through a set of smaller units, each being tested and then later combined for complete system testing the waterfall model was not chosen.

#### **4.4.2 Incremental Development**

The incremental development model (refer Figure 5), a fundamental part of the agile method, allows for a staged approach to software development. This approach allows for the specification, development and validation to be interleaved providing a fast response to requirement alterations. As the software can be developed incrementally, working code can be generated quickly and early in the software life cycle. Also the scope and requirements were not completely known and so this approach

provided the flexibility to accept the changes with minimal cost to change the scope and requirements as they developed. Testing and debugger smaller iterations of code are also favourable as it is easier to rectify and retest. As each unit of code is developed the end user can provide feedback to any required specification changes early in the software's development and testing.



**Figure 5 - Incremental Development**

The disadvantages to this model are that it requires good planning and design. Before the system can be broken down and created incrementally, there needs to be a clear and complete definition of the whole system. Not having a rigid structure allows the scope to drift potentially risking deadlines to complete the project.

The requirements for the complete system were clearly defined and understood, with details being expected to evolve during time. The time frame to develop the software was short, thus early development of code to be tested and debugged was essential. The incremental model did lend itself to satisfy the project based on the above reasons.

### 4.4.3 Re-use Oriented Development

The concept of retrieving GPS data, Wi-Fi access point information and store and forward data are not new concepts or designs. It was therefore feasible that a large portion of the software project would entail the reuse of software and integrated to suit the specifications. The advantage of code reuse is the development phase builds on previous work thus reducing development time (refer Figure 6). The components intended for reuse can be validated prior to implantation into the developing software, thus the potential errors can be debugged prior to continued development.

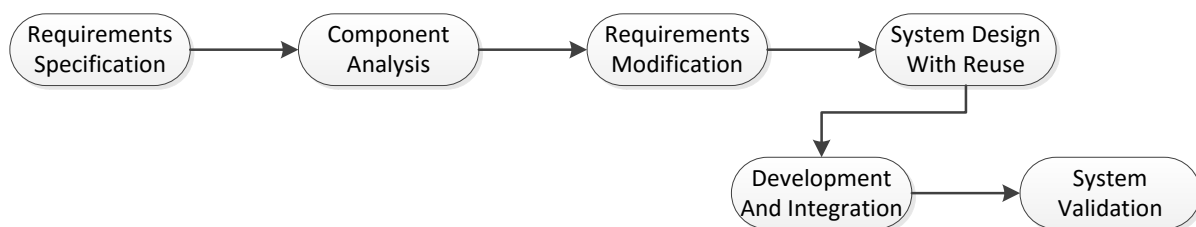


Figure 6 - Reuse-oriented Development

After review of the three basic models the combination of incremental development and reuse-oriented software engineering were chosen for the project. This was due to the expectation that the design will be altered many times during the development and that there would be suitable reuse of software.

## 4.5 Software Strategy

The methodology reviewed software applications capable to support the development and operation of the system. The software strategy includes the use of programming languages suited to developing the data collection application, data storage methods and the systems graphical user interfaces for the client and remote application.

### **4.5.1 Constraints**

The Fleet Management System (FMS) at Meandu utilises a ruggedized computer mounted onboard the fleet vehicles. These computers run a Windows XP Embedded operating system supporting the .NET framework 4.0 on an X86 processor. Being a Windows operating system all the development work was also completed in the Windows environment capable of supporting .NET 4.0.

### **4.5.2 IDE**

To develop software in any language the use of a source code editor is required. The source code also requires a compatible compiler that supports the libraries and operating systems. A widely used and support method for software development is Microsoft Visual Studio. Visual Studio is an integrated development environment (IDE). The advantages of using Visual Studio, is a developer can work with C#, PHP, JavaScript and Visual C++ all supporting the .NET framework for language interoperability. The IDE selected was Microsoft Visual Studio 2013 IDE Community Edition.

### **4.5.3 Simulation**

As a full production based FMS computer operating in the production environment was not available for developmental system testing, a virtualised testing platform was utilised. VirtualBox is a free, open source application that is capable of supporting the Windows operating system both as a host and client. Using the disk2vhd application, freely available, a virtual hard drive (VHD) of a running FMS computer was created for use with VirtualBox. This provided access to a simulated working system, with some limitations regarding wireless network adaptor access on the host, but still allowed some testing during development; importantly that the project application was compatible with the OS, producing no errors or degradation in the FMS system.

#### **4.5.4 Database**

The type of database for the system to operate was not a key requirement. However the criteria based on the overall philosophy during development was to utilise free, open source software were possible. MySQL met these criteria, is commonly used and well supported. A common suite of software packages for web serving data is known as WAMP; Windows, Apache, MySQL, PHP. These packages working together is a pseudo standard amongst developers. The selection based on the research suggested MySQL as a suitable package to utilise both as the local and remote database tools. Appendix E contains the database configuration information. The datatypes were selected based on the information being gathered by the application and the ability to filter and display information on a Google Map; in particular the format for latitude and longitude. Both the client computer and the remote database required each instance to have the MySQL server configured all ensuring the correct database server and schemas are aligned with all installations.

#### **4.5.5 Web Server**

Leading on from the WAMP concept, some display interfacing was desired with a logical interface to the information being via a web server. Apache web server is the most commonly used web server in the world and is well supported and documented (W3Techs, 2015). It also complies with the free and open source philosophy with the development for displaying information on via a web page being written in the PHP language. The Visual Studio IDE also supported PHP coding.

#### **4.5.6 Programming Languages**

There are numerous software programming languages that can be utilised for such a project. The perception as to the best language is subjective to the programmer based on personal and professional preferences. The approach was to survey available code developed by others that provided a potential guide towards the required solutions for the areas of serial port connectivity, GPS data set reading and



wireless network scanning. Other factors included the ability to use a simple display for testing and validating all iterations of the process. The choice was dependent on familiarity to the C family and the review of available examples of similar coding to the aim of the project. Some reuse of code, such as the use of the ManagedWiFi.dll and the ease of use in developing Windows forms for use GUI, the C# language is predominately used for the project. For development on web page design the review led to the Apache development being created with the use of PHP.

## **4.6 Client Design Overview**

The core process for the entire system is for the software to be installed on the client machines; being the FMS computing system. The remote database is capable of storing all the information transmitted from multiple clients. The data being transmitted from each client must align with the central server such that all data types match and each entry is a unique entry.

The design of the system to be installed on the clients was split into two distinct applications (refer Figure 7). The first was the collection of the information and storage to a local database. The second application was to retrieve unsent information and transmit it to a central database. The separation of these activities allows each to operate at distinct, adjustable, time intervals to collect and transmit the data, without affecting the performance of the other. The performance of this approach is discussed in Chapter 6.

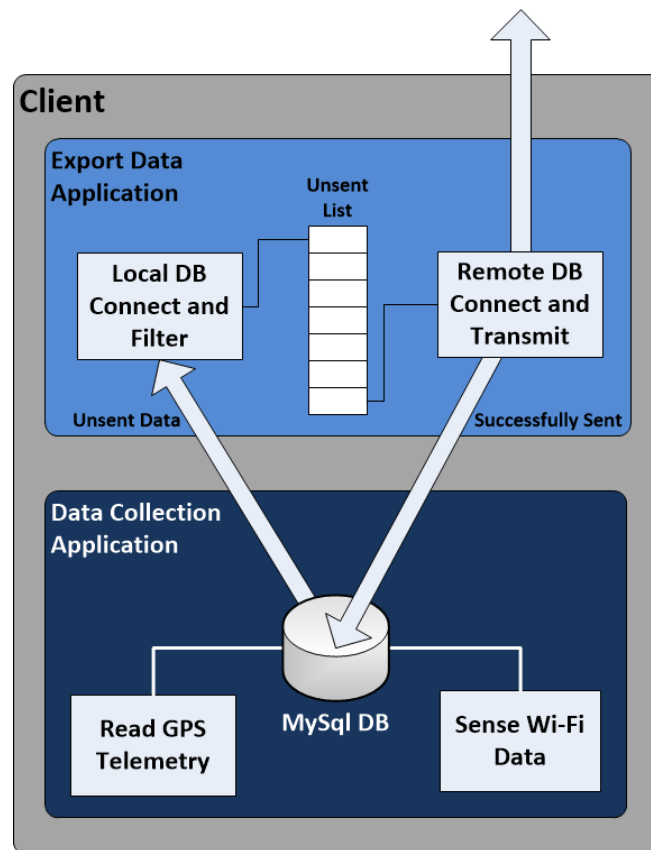


Figure 7 - Client Overview Data Flow

As shown in Figure 8, the data collection application retrieves the GPS location and all the available SSID information for each available access point. This information is then stored into a local database. This process is repeated continuously at a selected timer interval.

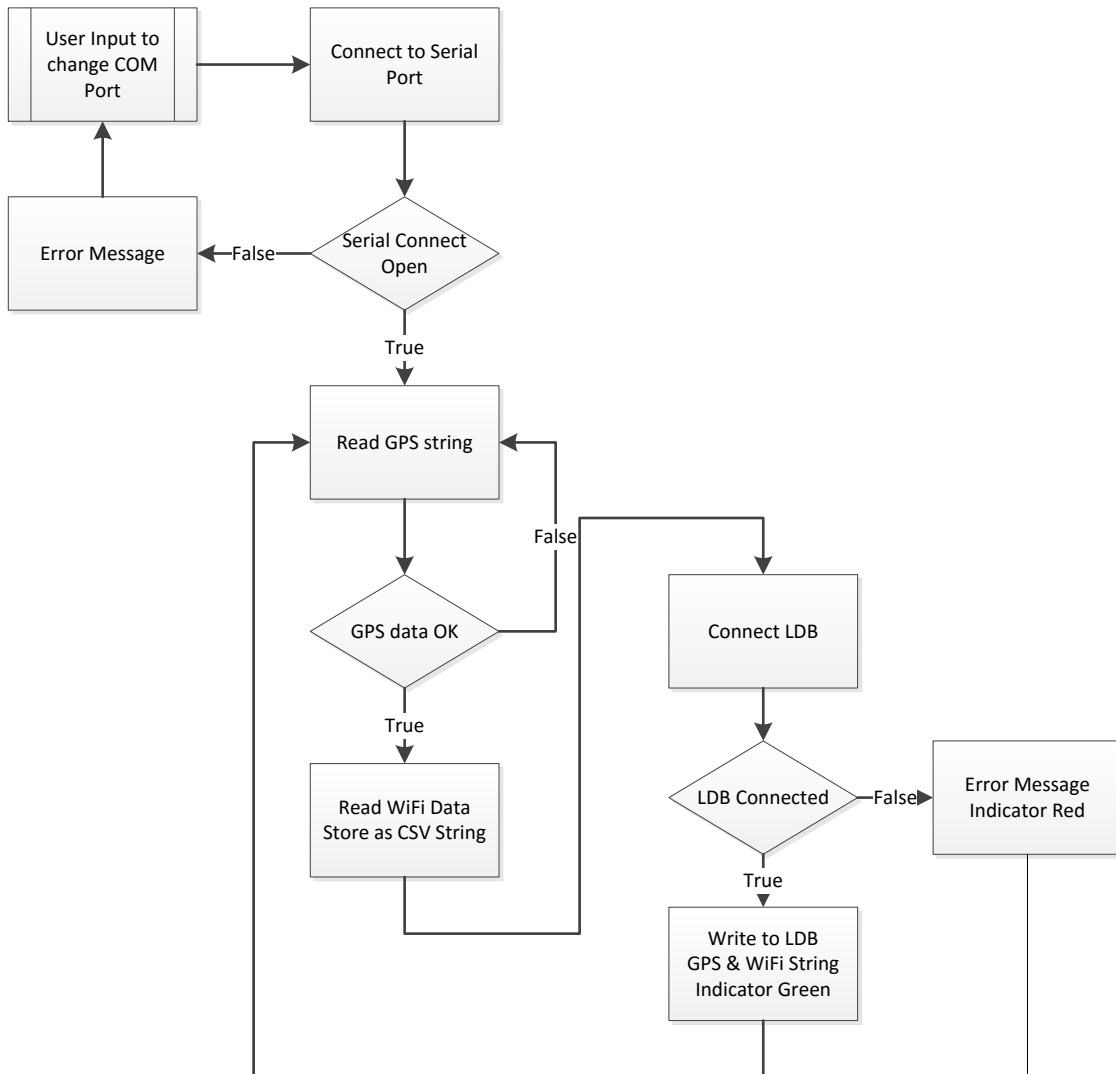
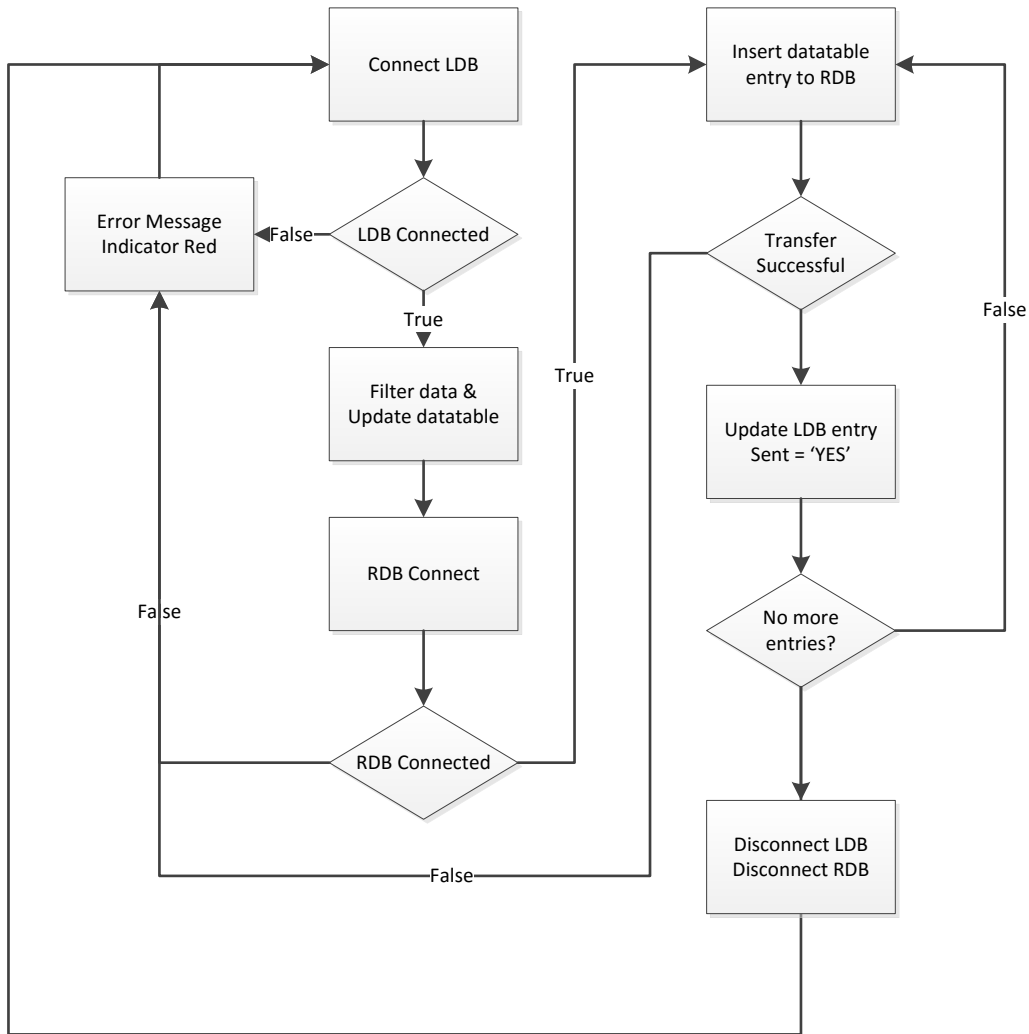


Figure 8 - Client Data Capture

Whilst the client data collection application is running, the second application is exporting the local information to a remote database (refer Figure 9). For each iteration of the timer in the export data application, the local database is filtered for all unsent data into a list; the database contains a table entry for sent data as a string for ‘Yes’ or ‘No’. As each list item is sent to the remote database the local database entry is updated to indicate that entry has now been sent.



**Figure 9 - Transmit Application**

The concept of this approach allows the local database to buffer the GPS locations during periods of ineffective wireless network coverage associated with the network the client should be connected to. Once network connection is re-established, the buffered GPS locations are sent to the remote database providing vital location data to allow accurate analysis of the wireless network coverage. Also if the data collection application has stopped, for any reason, the export functionality can continue until all buffered data has been sent; assuming network connection to the remote database is possible.

## **4.7 Software Testing**

The system was segregated into several modules: GPS, Wi-Fi, Local Database and Transmission. Each unit tested to ensure there was no inadvertent failure of the program as well as the system providing an accurate and desired response during normal and abnormal operation.

### **4.7.1 GPS Testing**

Testing of the GPS unit established reliable reading of the NMEA sentence streams. More importantly all the possible failure mechanisms were tested. The importance of the location information is central to the systems functionality. Testing of possible failure mechanisms included:

- GPS receiver disconnected at system start.
- GPS receiver disconnection during operation
- No satellite information available
- NMEA sentence unavailable

For a successful GPS read, the latitude, longitude, time and date information must all be successfully read from the NMEA sentence.

### **4.7.2 Wi-Fi Testing**

The desired response of the functioning system is to report GPS locations with whatever wireless information is or isn't available. Any failure of the local wireless adaptor on the client provides the same result as no wireless network information. The unit was tested for the scenario where the local wireless adaptor was not available, when no wireless networks were available and also when the systems wireless network was not present but other networks were.

### **4.7.3 Database Testing**

If the client application has no access to the local database or there is a failure to store the information the client application will not buffer the required information for transmission back to the remote storage. The system requires specific datatypes for each table entry which was tested against simulated data and data types, including incorrect formats to ensure the system did not fail unexpectedly. The remote database was tested in conjunction with the transmission unit described below.

### **4.7.4 Transmission Testing**

This application relies on the successful connection and manipulation of the local database and remote database. The transmission application was tested against failure to connect to the local database, failure to connect to the remote database before and during transmission. The unit's ability to collect and successfully update the local database was confirmed.

## **4.8 Conclusion**

This section has resulted in the development of a system overview capable of meeting the project specifications; to collect data at a client level and transmit this information to a remote server. The methodology has covered the tools and methods required to achieve these goals, with all the development relating to the ability to produce programs through coding. The methods to test the modules and system have been described with now allowing a detailed design to emerge for testing.

# Chapter 5

## Detailed Design

### 5.1 Introduction

With the development of the initial design overview this chapter provides information on the detailed design as was in place during development. As the method is that of an incremental approach to the engineered software, this detailed design was developed over several steps. Section 4.2 to 4.3 describes the user interfaces for data collection and transfer for the client machines. The general functionality of the code is explained in Section 4.5 with the detail explained from Section 4.6 onwards.

### 5.2 Collection User Interface

The client application collecting the Wi-Fi and GPS data has been designed with a simple user interface via a Windows Form. This provides the user with the ability to monitor the health of the application and provides the ability to control the application during runtime.

#### 5.2.1 Main Menu

The main menu is a standard Windows form interface providing a series of drop down menu item listings. The first menu tab provides the user to exit the application. Exiting the WiFi\_GPS\_Client application will stop all GPA and Wi-Fi data collection and the Windows form interface will close down for both the collection and transmit applications. A secondary method to close the application has been included as a pushbutton via the form's front end.

The second menu tab titled 'COM Port' provides a user with ability to select the serial communication port that the relevant GPS receiver will be connected to. By default this has been set to COM Port 4 during the testing.

The final menu item tab sets the scan time for the system to gather and store GPS and Wi-Fi data in the localised client based data base. By default this has been set to 1 second scan times. A selection of sample times has been included as 1, 2, 5, 10, 30 and 60 seconds. The amount of data collected will be directed affected by the sample time selected. This is discussed further in Chapter 6 for system performance.

Feedback indication has been included in each menu item selection (other than Exit) to provide a visual queue as to the current selections for Com Port and Sample Time.

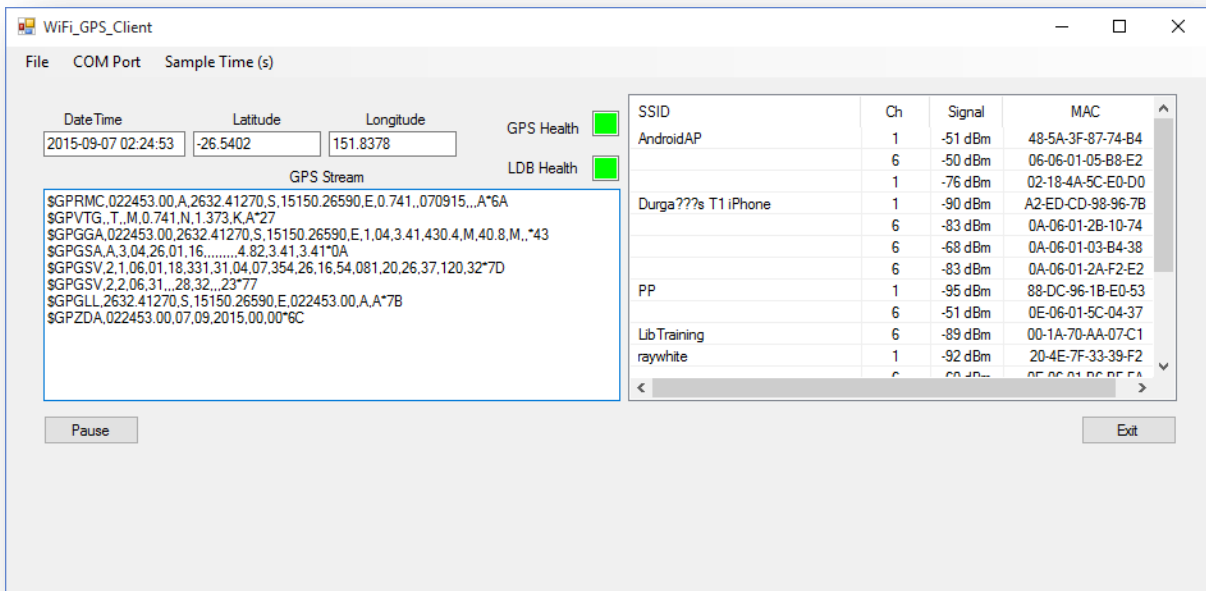


Figure 10 - Client Interface 1



### 5.2.2 Form Interface

The main display presents the current state of the system during normal operation and the ability to start, pause and exit the application. There is a single button that allows the user to pause or recommence the data collection process. By default the system will automatically commence in the running state if the COM port is available and working.

The information displayed to the user includes the states of the systems components:

- a) Serial communications port health
- b) GPS data health
- c) Local data base health
- d) Wi-Fi interface health

For each of the systems that are functioning correctly the form will display the current data sets as collected by the GPS receiver and the Wi-Fi adapter:

- a) GPS UTC date and time
- b) Last received latitude and longitude
- c) All NMEA strings received via the serial com port
- d) A list of all access points within range of the client wireless adapter

The last known GPS coordinates and UTC time are filtered from the NMEA sentences and presented to the GUI, providing a clear view of this vital piece of information (refer Figure 11).



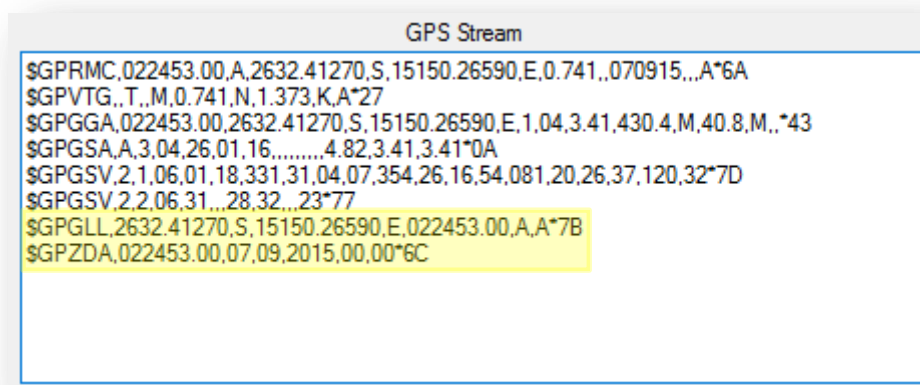
Date Time	Latitude	Longitude
2015-09-07 02:24:53	-26.5402	151.8378

**Figure 11 -Basic GPS Info**

The Date Time format has been configured as YYYY-MM-DD hh:mm:ss. This format provides a string format suitable for addition of markers to a Google Maps image . The data has been extracted

from the NMEA stream received via the serial communications port. The UTC date and time is extracted from the ZDA string (refer 3.5.1). This information is not converted in any way and whilst there may be value in displaying the time stamp as localised time, this has little value from the client perspective. The final goal of the system is to analyse the collective data received from multiple clients. It is the central display that could be developed to manipulate the time stamps to local time if required.

The GPS stream of NMEA sentences as displayed in Figure 12 gives all the available NMEA strings as activated in the GPS receiver.



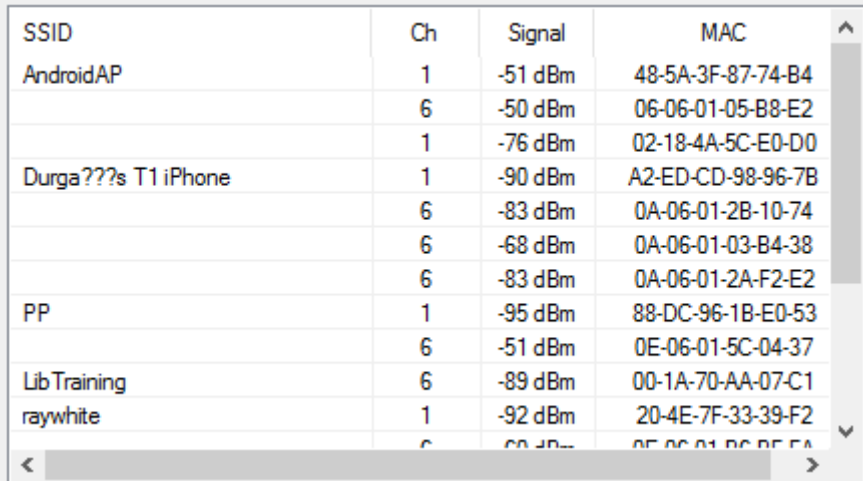
```
GPS Stream
$GPRMC,022453.00,A,2632.41270,S,15150.26590,E,0.741,,070915,..,A*6A
$GPVTG,.T.,M,0.741,N,1.373,K,A*27
$GPGGA,022453.00,2632.41270,S,15150.26590,E,1.04,3.41,430.4,M,40.8,M,..*43
$GPGSA,A,3,04,26,01,16,.....,4.82,3.41,3.41*0A
$GPGSV,2,1,06,01,18,331,31,04,07,354,26,16,54,081,20,26,37,120,32*7D
$GPGSV,2,2,06,31,..,28,32,..,23*77
$GPGLL,2632.41270,S,15150.26590,E,022453.00,A,A*7B
$GPZDA,022453.00,07,09,2015,00,00*6C
```

Figure 12 - NMEA Stream

The primary data required for this initial development of the system is the GLL and ZDA sentences for the Latitude and Longitude as well as the Time and Date, respectively. The GLL string does contain the UTC time. This was initially used but during early testing the requirement for the date became obvious when attempting to analyse data received over different days.

Iteration of the sample time has the system rescan the wireless networks available to the client wireless adapter. For each available network there is a basic set of data that is displayed to the user. The data is displayed to the user and also built into a single string for storage in the local and remote

databases (refer Figure 13).The service set identifier (SSID) is the public name of the wireless network. Entries that are blank; the SSID has been set not to broadcast.



SSID	Ch	Signal	MAC
AndroidAP	1	-51 dBm	48-5A-3F-87-74-B4
	6	-50 dBm	06-06-01-05-B8-E2
	1	-76 dBm	02-18-4A-5C-E0-D0
Durga???'s T1 iPhone	1	-90 dBm	A2-ED-CD-98-96-7B
	6	-83 dBm	0A-06-01-2B-10-74
	6	-68 dBm	0A-06-01-03-B4-38
	6	-83 dBm	0A-06-01-2A-F2-E2
PP	1	-95 dBm	88-DC-96-1B-E0-53
	6	-51 dBm	0E-06-01-5C-04-37
Lib Training	6	-89 dBm	00-1A-70-AA-07-C1
raywhite	1	-92 dBm	20-4E-7F-33-39-F2
	6	-80 dBm	0E-06-01-0C-0F-FA

**Figure 13 - WiFi Data**

Each network is operating on a standard channel as per the 802.11 standard. The channel is listed in the CH column and as discussed in the introduction range from 1 to 13. If there is a localised problem with interference a trained user can view the live data and interpret any nearby APs that may be conflicting on the same channel or partially crossed channels.

The column labelled Signal contains the received signal strength indication (RSSI). As discuss in section 2.5 the RSSI the signal strength is a key piece of information when discerning any network problems. The requirements for the client tool do not require any further manipulation of the received Wi-Fi data as all detailed analysis should occur from the collated data at the central database. The information does however provide some initial diagnosis.

The physical address of each network is contained within the basic service set identifier (BSSID) or media access control (MAC) address. Each MAC address is unique for all network adapters produced

globally. As a wireless network may contain more than one access point with the same SSID or network name, it is important to capture all the listed APs associated with that network. The mining equipment at Meandu all connect to a single network across the Mesh network, so being able to distinguish each of the APs within the Mesh is important when analysing the performance and coverage by each node.

### 5.2.3 Systems Health

The health of each system is presented to the user with clear indication. Currently there is no notification or record taken of the failures, only when no Wi-Fi data is available. A fundamental design of the system is that the program will not record any data if the GPS data is not healthy. This approach has been taken, as per the specification, that the primary role of the system is to determine network issues at precise GPS positions, thus if the GPS data is not available any further information regarding wireless network quality is of no value to the central database.

On commencement of the application should the COM port selected by default not be available, such as the GPS receiver is disconnected or failed, the user will receive a notification (refer Figure 14)

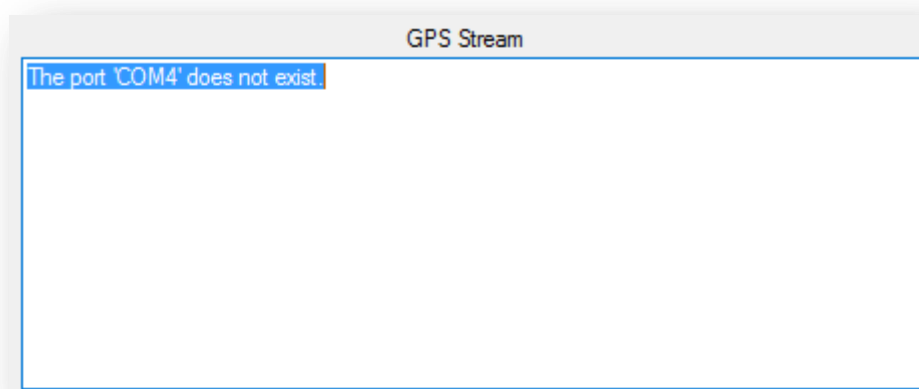
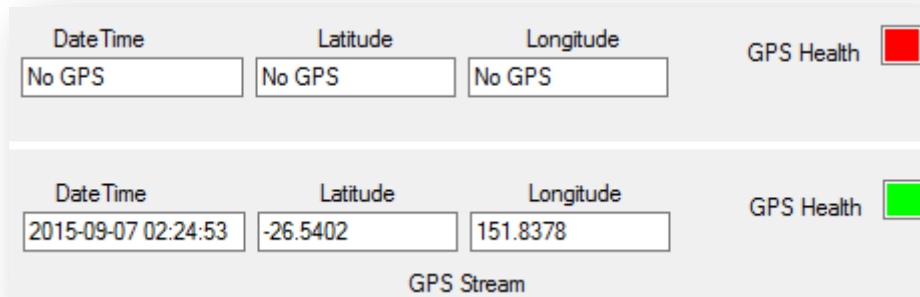


Figure 14 - COM Port Error

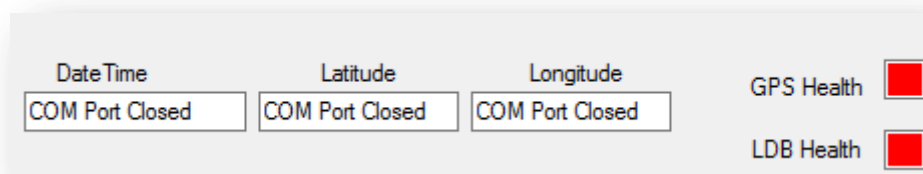
The system will continue to wait for the existing COM port to become available or the user can attempt to select another port from the main menu.



**Figure 15 - GPS Health Good vs Bad**

When the GPS data is healthy the user will see the current date, time, latitude and longitude. The text boxes for the simplified GPS information will also display 'No GPS' as an indication that suitable information to the GPS receiver is not available (refer Figure 15).

For a serial comports failure to connect to the GPS receiver the display is similar, with the GPS Health indicator as red and text information related to the failure will be present as shown below.



**Figure 16 - COM Port Disconnect**

The overall design of the user interface is for a technical person to interpret and analyse the raw information as collected by the client. Generally the system would run as a background Windows service collecting and transmitting the data. The vehicle operators would have no interaction with the software.

### **5.3 Transmit User Interface**

As the GPS and wireless network information is collected at each known client, a requirement of the system is such that each client must be able to transmit data back to a central database for analysing and monitoring of the overall network. A second interface has been developed to provide a user interface for the transfer of the buffered data, collected by the client application, to a single central database.

#### **5.3.1 Main Menu**

For consistency in design and functionality the collection user interface application (refer 5.2) and the transmit user interface application described here provide almost identical interfacing. The main menu is a standard Windows form interface providing a series of drop down menu item listings. The first menu tab provides the user to exit the application. Exiting the Transmit application will stop attempting connection to the local (client) and remote database and the Windows form interface will close down. A secondary method to close the application has been included as a pushbutton labelled 'Exit' via the form's front end.

The second menu tab titled 'Timer' provides a user with ability to select the period between transmission attempts to the remote database. By default this has been set to 5 second intervals during the testing. A selection of transmit attempt times has been included as 5, 10, 15, 20, 30 and 60 seconds. The amount of data filtered and sent will be directed affected by the sample time selected. This is discussed further in Chapter 5 for system performance.

Feedback indication has been included in each menu item selection (other than Exit) to provide a visual queue as to the current selections for transmit timing intervals.

### 5.3.2 Form Interface

The Transmit application form interface provides the user with a clear indication of the application during normal operation. The user has the ability to pause and recommence the operation of transmitting data via the Pause/Run pushbutton, toggling between the two states (refer Figure 17). The Exit pushbutton stops all functions for accessing local and remote databases and closes the application.

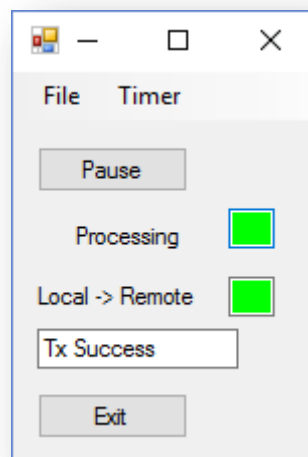


Figure 17 - Transmit Application

The text box provides a simplified update on the systems operation during each attempt to transmit data. There are three states the system can be in during operation:

- a) Transmitting...
- b) Tx Success

c) Tx Failed!

Each operation of the timer the system will attempt to transmit any available data to be sent from the local database. If the system successfully connects and transmits to the remote database 'Tx Success' is displayed. Any failure in transmitting the information for that cycle will result in a 'Tx Failed'.

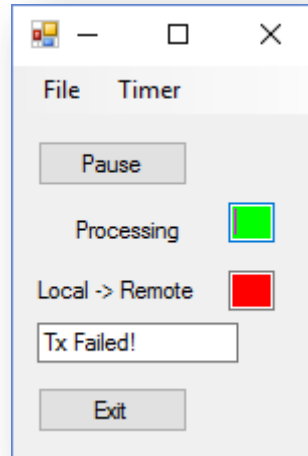
### 5.3.3 System Health

Besides the text indication there are two indicators for the user to interpret the current state of the Transit application.

During running operation, that is the timer is active and operating, the 'Processing' indicator is green. When the system has been paused, the timer is no longer active and the 'Processing' indicator will display red.

The second indicator (refer Figure 18) labelled 'Local -> Remote' indicates a successful transfer from the local database to the remote database. A successful transfer will result in a green indicator. Any failure to transmit will result in a red indicator.





**Figure 18 - Transfer Failed!**

The interface for the transmission of local data to a remote database provides the user with a simple tool to operate and analyse the operation of this portion of the system.

## 5.4 Coding Overview

The system has been developed with two separate applications; collection and transmission. Both the applications are run from the main program (refer Appendix C1: ).

### 5.4.1 Client Overview

The client application on commencement establishes if the default COM port has been opened successfully. Should this fail the client application presents the COM port error (refer 5.2.3), and continues to wait for a successful COM port selection from the user. On successful connection to the serial port the system then:

1. reads in the GPS receiver ASCII stream,
2. separates each NMEA sentence via '\$' character leading each sentence
3. Extracts the UTC Date and Time from the ZDA sentence
4. Extracts the Longitude and Latitude from the GLL sentence
5. Retrieves all Wi-Fi information (if any) into a single string

6. Stores the data in a local MySQL database
7. Repeats continuously - with fault conditioning and user interaction

Should the GPS stream or connection to the receiver fail the system will not store data for that iteration of the timer. The concept for the system is to ensure reliable information is captured relating to the position of the client. Should the GPS information not be available for any reason the Wi-Fi information that may be available at that location is not relevant as it cannot be linked to a location.

Alternatively during any failure to collect Wi-Fi data (regardless of the reason) the GPS data is still stored in the local database with no, as this will present to the remote database as problem experience by that particular client presenting no Wi-Fi data for those locations.

#### **5.4.2 Transmit Overview**

The transmit application has been developed separately and runs separately from the client application. The system commences after successful commencement of the client application. During normal operation the transmit application:

1. Connects to the local database
2. filters for any unsent data in a list
3. connects to the remote database
4. transmits each data row to the remote database
5. updates the local database items as sent
6. close both database connections

This cycle is repeated continuously based on the timer value set.

## 5.5 GPS Interface Coding

The NMEA sentences are read into a string on each iteration. The sentences are parsed out and compared to the relevant sentence type; the GLL sentence containing the relevant latitude and longitude information and the ZDA sentence with the UTC time associated with that location.

If there is healthy GPS data the system will continue to retrieve Wi-Fi data and store the information. Should the GPS data be inadequate no information is recorded and the program waits for the next timer tick to attempt the next GPS read.

## 5.6 Wi-Fi Interface Coding

The system requires access to the wireless adaptor on the fleet vehicles FMS computer. These machines as previously discussed in the background chapter, are a ruggedized computer operating on a Windows XP embedded platform. This provides the use of the standard Windows NativeWiFi API available through the .NET framework. For the conceptual development of this application the use of the ManagedWiFi.dll was incorporated into the program.

### 5.6.1 ManagedWiFi

ManagedWiFi.dll is an API developed at Monfort Software Engineering. It utilises the Windows Native Wifi API library to provide a .NET class library for controlling Windows wireless adaptors. This allowed all the available wireless adaptor information to be captured in a single scan with all the relevant information to be extracted via the class members and functions.

The library, whilst professionally developed, is not a complete and comprehensive coding solution. For the research into this project, only the essential information on SSID, RSSI and channel data was sourced which this library satisfied.

## 5.7 MySQL

A class was created to handle all MySQL actions. This class provides the interface to open and close the connection to a database. The system uses the class functions to insert the required data into a database, retrieve a list of unspent entries from the local database, update the list and transfer entries from one database to another. The code detailing the member functions as shown in the mysqlDB code (refer Appendix C4: ).

## 5.8 Graphical User Interface

There are many methods available to display information through a graphical user interface (GUI). As the research suggested using a WAMP model (Windows, Apache, MySQL, PHP) with the use of a web interface based on an Apache server can provide a suitable interface.

An initial interface developed with PHP (refer Appendix D). This provides a single snapshot of the data at each launch of the web page. Whilst beyond the scope of this project, the concept of reproducing the data visually via a Google map, including some indication of areas where network was compromised was successful.

## 5.9 Conclusion

The requirements in section 4.3 directed the detailed design and the results required. This chapter has developed the detailed design for the client application to operate in conjunction with the hardware available on the FMS computing system. This includes accessing the wireless adapter to scan for all wireless access points within range of the client. Also the design connects to and retrieves the available GPS coordinates from the GPS receiver. The user has been given a basic interface to ensure adequate control over the functions performed during data collection and buffering as well as the transfer of data to the remote database. The system as a whole can be tested and analysed to test for performance issues and results analysis.

# Chapter 6

## Analysis and Performance

### 6.1 Introduction

In this chapter the program is tested under varies failure conditions as well as an expected normal operation. The results are analysed to determine if the system operates in a manner that satisfies the aim of the project.

### 6.2 Analysis Section

The system was run in the simulated FMS on VirtualBox (refer Figure 19). A virtual GPS signal was also run on the simulation to test the client applications ability to connect and receive information from a GPS receiver. The initial findings showed the application performed without compromising the FMS systems.

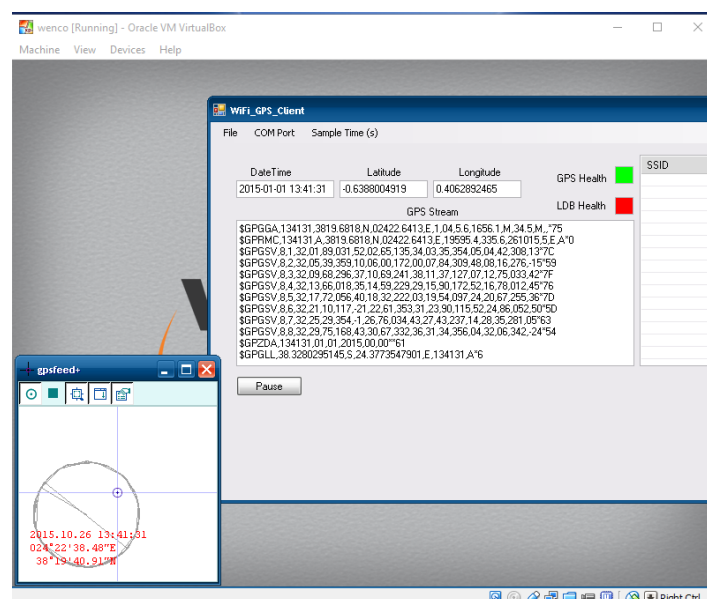


Figure 19 - VirtualBox

### 6.2.1 Normal Operation

The system, as an initial concept, was tested at Meandu mine. The network infrastructure was fully operational with no faults present. The software was loaded and run from a laptop mounted in a light vehicle (Hilux) with its wireless adapter connected to the local wireless network and a USB GPS receiver mounted externally on the vehicle (refer Figure 20). A second laptop containing the remote database had been configured and connected to the network backhaul via a LAN connection.



Figure 20 - GPS Mount

The client application was started and the vehicle was driven around several kilometres of light vehicle roads and haul roads. During this testing period the performance of both the data collection application and data transmission application were monitored (refer Figure 21).



**Figure 21 - Application Testing**

The client performed as expected during normal conditions and throughout a series of faults. The fault conditions tested included:

- Disconnecting and reconnecting the USB GPS receiver
- Stop and restart of local database
- Wireless adaptor stopped and restarted

The system produced all the correct fault information as per the detailed design with the data either not captured or buffered depending on valid GPS coordinates.

### **6.2.2 Data Buffering**

The prime functionality of the system is to buffer GPS location when operating in areas with no wireless connectivity to the remote database. It is this capability that provides the unique point of indifference to other wireless network monitoring tools available. This was achieved during testing with an area resulting in no available wireless access points. The result however shows no networks whilst travelling in one direction then some networks available on the return trip on the same stretch of road. The initial web page design (refer Figure 22) has been configured to produce green markers where network connectivity exists and red markers for locations containing no connectivity. As the





```

<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:36" lat="-26.812401" lon="151.919601" wifi="STA-CHARLIE,00-0D-97-19-75-83,-84,6.STA-MESH,00-0D-97-09-75-83,-85,6."/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:38" lat="-26.812599" lon="151.919266" wifi="STA-CHARLIE,00-0D-97-19-75-83,-84,6.STA-MESH,00-0D-97-09-75-83,-85,6."/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:40" lat="-26.812700" lon="151.919083" wifi="STA-CHARLIE,00-0D-97-19-75-83,-84,6.STA-MESH,00-0D-97-09-75-83,-85,6."/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:44" lat="-26.812981" lon="151.918732" wifi="STA-CHARLIE,00-0D-97-19-75-83,-84,6.STA-MESH,00-0D-97-09-75-83,-85,6."/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:48" lat="-26.813110" lon="151.918365" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:50" lat="-26.813210" lon="151.918182" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:54" lat="-26.813379" lon="151.917816" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:38:58" lat="-26.813520" lon="151.917496" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:00" lat="-26.813570" lon="151.917343" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:04" lat="-26.813601" lon="151.917007" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:08" lat="-26.813490" lon="151.916656" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:10" lat="-26.813391" lon="151.916489" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:14" lat="-26.813141" lon="151.916214" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:18" lat="-26.812860" lon="151.916000" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:20" lat="-26.812710" lon="151.915894" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:24" lat="-26.812389" lon="151.915695" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:28" lat="-26.812031" lon="151.915543" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:30" lat="-26.811840" lon="151.915497" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:34" lat="-26.811440" lon="151.915466" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:38" lat="-26.811001" lon="151.915512" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:40" lat="-26.810850" lon="151.915543" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:44" lat="-26.810471" lon="151.915680" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:48" lat="-26.810101" lon="151.915848" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:50" lat="-26.809919" lon="151.915955" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:54" lat="-26.809589" lon="151.916183" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:39:58" lat="-26.809259" lon="151.916443" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:00" lat="-26.809099" lon="151.916595" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:04" lat="-26.808800" lon="151.916885" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:08" lat="-26.808531" lon="151.917160" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:10" lat="-26.808430" lon="151.917206" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:14" lat="-26.808310" lon="151.917450" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:18" lat="-26.808331" lon="151.917755" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:20" lat="-26.808399" lon="151.917938" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:24" lat="-26.808510" lon="151.918350" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:28" lat="-26.808670" lon="151.918747" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:30" lat="-26.808760" lon="151.918900" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:34" lat="-26.808880" lon="151.919205" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:38" lat="-26.808870" lon="151.919464" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:40" lat="-26.808809" lon="151.919617" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:44" lat="-26.808590" lon="151.919907" wifi=""/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:48" lat="-26.808350" lon="151.920227" wifi="STA-CHARLIE,00-0D-97-19-75-83,-81,6.STA-MESH,00-0D-97-09-75-83,-83,6."/>
<marker id="GLENN_LAPTOP" ts="2015-09-10 01:40:50" lat="-26.808220" lon="151.920595" wifi="STA-CHARLIE,00-0D-97-19-75-83,-81,6.STA-MESH,00-0D-97-09-75-83,-83,6."/>

```

Figure 23 - Raw data XML

## 6.2.3 Performance

The system performs as expected during normal operation as well as during fault conditions, such as GPS receiver failure (USB unplugged) and return to normal. The system does have an issue during periods of extended buffering during poor wireless coverage. As the local database compiles an ever increasing list of unsent waypoints upon return to network connectivity the transmission application attempts to retrieve the entire list for transmission to the remote database. The testing of this fault as with the collection rate at one second intervals and the transmission application set to five second intervals. The transmission application was paused, so as to allow buffering of data more substantial than occurs during the five second transmit times. Degradation of the system became noticeable with the buffered entries exceeding thirty; this is thirty seconds of collection time. Whilst the display indicated some fault of the GPS data on the GUI, the resultant data sent to remote database was not corrupted, the collection application however did miss up to two entries during collection. Whilst this issue does require further investigation, the resultant data at the remote database is not corrupted, in that if no viable GPS info is available at the client, then no Wi-Fi data is sent. The fidelity of the information is affected but the system still report the areas affected and not affected by wireless coverage.

### **6.3 Conclusion**

The system was tested for many operating conditions and fault conditions. The resultant applications performed as designed and expected meeting the requirements as set on in section 4.3. More work with the simulation of the FMS computer will provide a structured testing regime to mitigate any future unwanted interactions causing system instability or failure. The system can also be tested on spare hardware if made available. Whilst the application appears successful there is more testing required.

# Chapter 7

## Conclusion

### 7.1 Introduction

A wireless network is design to provide suitable network coverage for client systems to receive and transmit data. The design of this monitoring software is to add to the collection of tools available to monitor and analyse the performance of a wireless network, where the network is designed for outdoor systems supporting GPS tracking. The proposed system provides network details as recorded by the client infrastructure, for coverage and no coverage that can be live. Crucially, this information is not estimated, simulated or modelled but actual network information seen by the client.

### 7.2 Achievement of Objectives

The project results have shown the possibility of extending the use of the mobile fleet computing hardware to support a client based wireless network monitoring tool. The overall system has been conceptualised, with alternatives evaluated within the literature review. A methodology to the approach taken has been detailed and followed to produce a detailed design.

The system has been coded and tested in both simulation and real world application; noting the restriction to perform the test on live production equipment. The testing showed the system has the ability to record Wi-Fi information related to GPS locations and during gaps in the network coverage the system successfully buffered the locations. The transmission to a remote central database was successful, including the transmission of buffered data from a client out of network range for short periods. This data was presented in a basic visual display as an extension to the primary objectives.

The results show that the use of client computing infrastructure operating on an outdoor industrial network can be used to assist in accurately reporting network quality based on GPS locations. The

ability for the system to use a central database, receiving the network information from all the clients allows a user to analyse the data live. The ability to analyse the live data will assist in rectifying the coverage issues much sooner and more efficiently as further manual survey work is not required.

### **7.3 Further Work**

The expectation of the designed system is to present the data collected as a live window. The vision would be to have the data presented overlaid on a map of the networked area. Filtering of the information would also be highly desirable to select time windows, individual client results, areas of no signal and automatic reporting and notifications. As different computing systems, such as tablets, laptops and mobile phones with GPS and wireless capability become part of the outdoor network, the application can be modified to support the different mobile systems to further add to the network monitoring.

Applying the data to modelling software may also lead to optimisation of network infrastructure location and help predict problems before they occur.

## List of References

Crowdsourcing, Definition of crowdsourcing by Merriam-Webster, 2015, viewed 22 May 2015, <<http://www.merriam-webster.com/dictionary/crowdsourcing>>

Einicke, G, Duff, E, Reid, D, Ralston, J, 2002, *The application of wireless LANs in mine automation*, viewed 12 May 2015, <[https://scholar.google.com.au/scholar?q=open+cut+mining+wireless+lan&btnG=&hl=en&as\\_sdt=0%2C5](https://scholar.google.com.au/scholar?q=open+cut+mining+wireless+lan&btnG=&hl=en&as_sdt=0%2C5)>

Forooshani, A, Bashir, S, Michelson, D, Noghianian, S, 2013, *A Survey of Wireless Communications and Propagation Modelling in Underground Mines, Communications Surveys & Tutorials*, IEEE, viewed 24 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=6489881>>

Frangoudis, P, & Polyzos, G, 2015, *Reputation-based crowdsourced Wi-Fi topology discovery. Computer Networks*, Volume 79, Pages 1–16, viewed 14 March 2015, <<http://www.sciencedirect.com.ezproxy.usq.edu.au/science/article/pii/S1389128614004666>>

Ganti, R, Fan, Y, Hui, L, 2011, *Mobile crowdsensing: current state and future challenges, Communications Magazine*, IEEE, viewed 5 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=6069707>>

Gunther, A, 2015, *Getting Familiar with Wi-Fi Channels? WLAN Back to Basics*, viewed 15 October 2015, <<http://boundless.aerohive.com/technology/WLAN-Channels-Explained.html>>

Holt, A, 2014, *Wireless Mesh Networks*, viewed 20 May 2015, <<http://www.utilitynetworks.co.uk/site/content/wireless-mesh-networks-alan-holt>>

IEEE 802.11s: *The WLAN Mesh Standard*, viewed 25 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=5416357>>

Jun, W, Geng, L, 2010, *Research of mine wireless communication system based on wireless Mesh technology, Image and Signal Processing (CISP)*, 2010 3rd International Congress on, IEEE, viewed 27 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=5647907>>

Kloos, G, Guivant, J, Worrall, S, Maclean, A, & Nebot, E, 2004, *Wireless network for mining applications, In Australasian Conference on Robotics and Automation (ACRA)*, viewed 31 May 2015, <[https://scholar.google.com.au/scholar?q=wireless+network+for+mining+applications&btnG=&hl=en&as\\_sdt=0%2C5](https://scholar.google.com.au/scholar?q=wireless+network+for+mining+applications&btnG=&hl=en&as_sdt=0%2C5)>

Moutairou, M, Aniss, H, Delisle, G, 2006, *Wireless mesh access point routing for efficient communication in underground mine, Antennas and Propagation Society International Symposium 2006*, IEEE, viewed 27 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=1710589>>

NMEA 0183 Standard, National marine electronics association, 7 Riggs Avenue, Severna Park, MD; 2006

Novak, T, Snyder, D, Kohler, J, 2010, *Postaccident Mine Communications and Tracking Systems , Industry Applications, IEEE Transactions on*, viewed 12 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=5382595>>

Radu, V, Kriara, L, Marina, M, 2013, *Pazl: A mobile crowdsensing based indoor WiFi monitoring system, Network and Service Management (CNSM)*, 2013 9th International Conference, viewed 27 April 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=6727812&tag=1#references>>

Rosen, S, Sung-Ju, L, Jeongkeun, L, Congdon, P, Mao, Z, Burden, K, 2014, *MCNet: Crowdsourcing wireless performance measurements through the eyes of mobile devices, Communications Magazine, IEEE*, viewed 7 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=6917407>>

Sheth, A, 2007, *Improving the Performance of Wireless Systems Through Distributed Fault Diagnosis*, viewed 29 May 2015, <[https://books.google.com.au/books?id=C-pDi8VsMGsC&dq=causes+of+wireless+network+degradation&source=gbs\\_navlinks\\_s](https://books.google.com.au/books?id=C-pDi8VsMGsC&dq=causes+of+wireless+network+degradation&source=gbs_navlinks_s)>

Shi, J, Guan, Z, Qiao, C, Melodia, T, Koutsonikolas, D, Challen, G, 2014, *Crowdsourcing Access Network Spectrum Allocation Using Smartphones*, viewed 5 May 2014, <<http://dl.acm.org.ezproxy.usq.edu.au/citation.cfm?doid=2670518.2673866>>

Sommerville, I 2011, *Software Engineering* (9<sup>th</sup> International Edition), Pearson, Boston.

Tanghe, E, Joseph, W, Verloock, L, Martens, L, Capoen, H, Van Herwegen, K, Buysschaert, T, 2009, *Statistical validation of WLAN range calculated with propagation models for industrial environments by chipset-level received signal strength measurements, Science, Measurement & Technology, IET* (Volume:3 , Issue: 3 ), viewed 15 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpl/articleDetails.jsp?tp=&arnumber=4907013&queryText%3Dwlan+modelling+or+site+survey>>

*Usage statistics and market share of Apache for websites*, W3Techs, viewed 28 May 2015, <<http://w3techs.com/technologies/details/ws-apache/all/all>>

Vellingiri, S, Tandur, D, Kande, M, 2013, *Communication architecture for Remote Monitoring and Diagnostics in Open Pit Mine, Emerging Technologies & Factory Automation (ETFAs)*, 2013 IEEE 18th Conference on, viewed 21 May 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=6648073>>

Zhou, R, Xing, G, Xu, X, Wang, J, Gu, L, 2013, *WizNet: A ZigBee-based sensor system for distributed wireless LAN performance monitoring, Pervasive Computing and Communications (PerCom)*, 2013 IEEE International Conference on, viewed 27 April 2015, <<http://ieeexplore.ieee.org.ezproxy.usq.edu.au/xpls/icp.jsp?arnumber=6526722>>

Zvanovec, S., Pechac, P., & Klepal, M. (2003). *Wireless LAN networks design: site survey or propagation modeling?*. *Radioengineering*, 12(4), p42-49, viewed 9 May 2015, <[https://scholar.google.com.au/scholar?q=wlan+site+survey+outdoor&btnG=&hl=en&as\\_sdt=0%2C5](https://scholar.google.com.au/scholar?q=wlan+site+survey+outdoor&btnG=&hl=en&as_sdt=0%2C5)>

## Bibliography

ikonst (2010). ManagedWiFiAPI (Version 11) [Computer Software]. Retrieved from codeplex.com

Project Directories: <http://managedwifi.codeplex.com/>

salysl (2008). *Mapping with a GPS and C#*. Retrieved from

[http://www.codeproject.com/Articles/23135/ Mapping-with-a-GPS-and C -](http://www.codeproject.com/Articles/23135/Mapping-with-a-GPS-and-C-)



## **Appendix A: Project Specification**

The project specification as agreed to by the author and the project supervisor.

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

**ENG4111 Research Project**  
PROJECT SPECIFICATION

FOR: Glenn LUMSDEN

TOPIC: INDUSTRIAL WIRELESS NETWORK MAPPING

SUPERIVOR: Dr Alexander Kist

ENROLEMENT: ENG4111 Semester 1, Ext, 2015

PROJECT AIM: To map the coverage of outdoor wireless distribution systems or mesh network coverage through client reporting of GPS location versus access point signal strengths to assist in maximising network coverage.

PROGRAMME: Issue C, 2<sup>st</sup> April 2015

1. Undertake a comprehensive requirements analysis, consult with stakeholders and note any commercial constraints that might impact on the system design.
2. Detailed literature review on Wi-Fi and GPS technology.
3. Background research on similar existing technology and systems.
4. Conceptualise the overall system and evaluate alternative approaches and implementations.
5. Design and test a system to retrieve Wi-Fi signal strength and GPS data from mining fleet.
6. Design and test a central server to capture and store the network data
7. Document all steps of the project and write an academic dissertation on the research.

As time permits:

8. Develop and test a method to display information in a graphical format
9. Develop and test a method to analyse the data to determine coverage issues

AGREED:

\_\_\_\_\_ (Student)      \_\_\_\_\_ (Supervisor)

\_\_\_\_/\_\_\_\_/\_\_\_\_

\_\_\_\_/\_\_\_\_/\_\_\_\_

Examiner/Co-examiner: \_\_\_\_\_

## **Appendix B: Wencomine Fleet Management**

## Appendix B1: Minevision

The fleet management system overview shows the location of all mining fleet vehicles (Figure 24). The status of any machine selected is displayed on the right hand side. Should a vehicle no longer be in network range the system shows the status as 'Out of Network Range' (Figure 25). The detail of where the vehicle has been is not recorded or available.

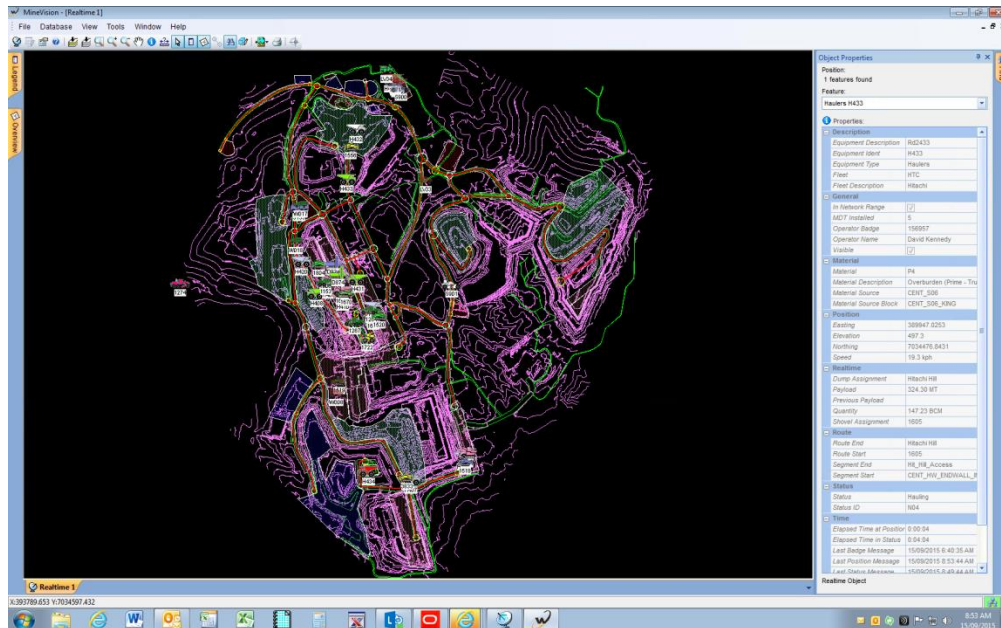


Figure 24 - Minevision

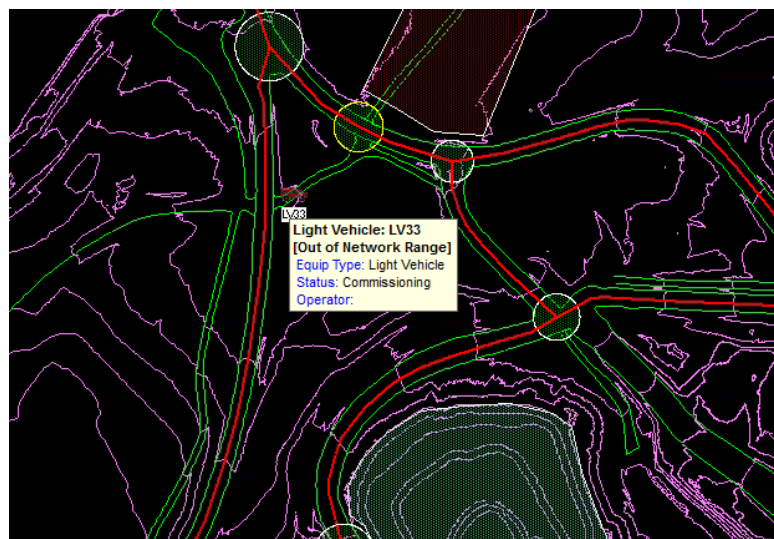


Figure 25 - Out of Network Range

## Appendix B2: Fleet Control

Another application for the FMS is the ability to monitor the performance of a fleet and the supporting vehicles, such as bulldozers and scrapers. The Fleet Control interface provides the business with the ability to remote access the machines to update the plan as well and monitor performance.

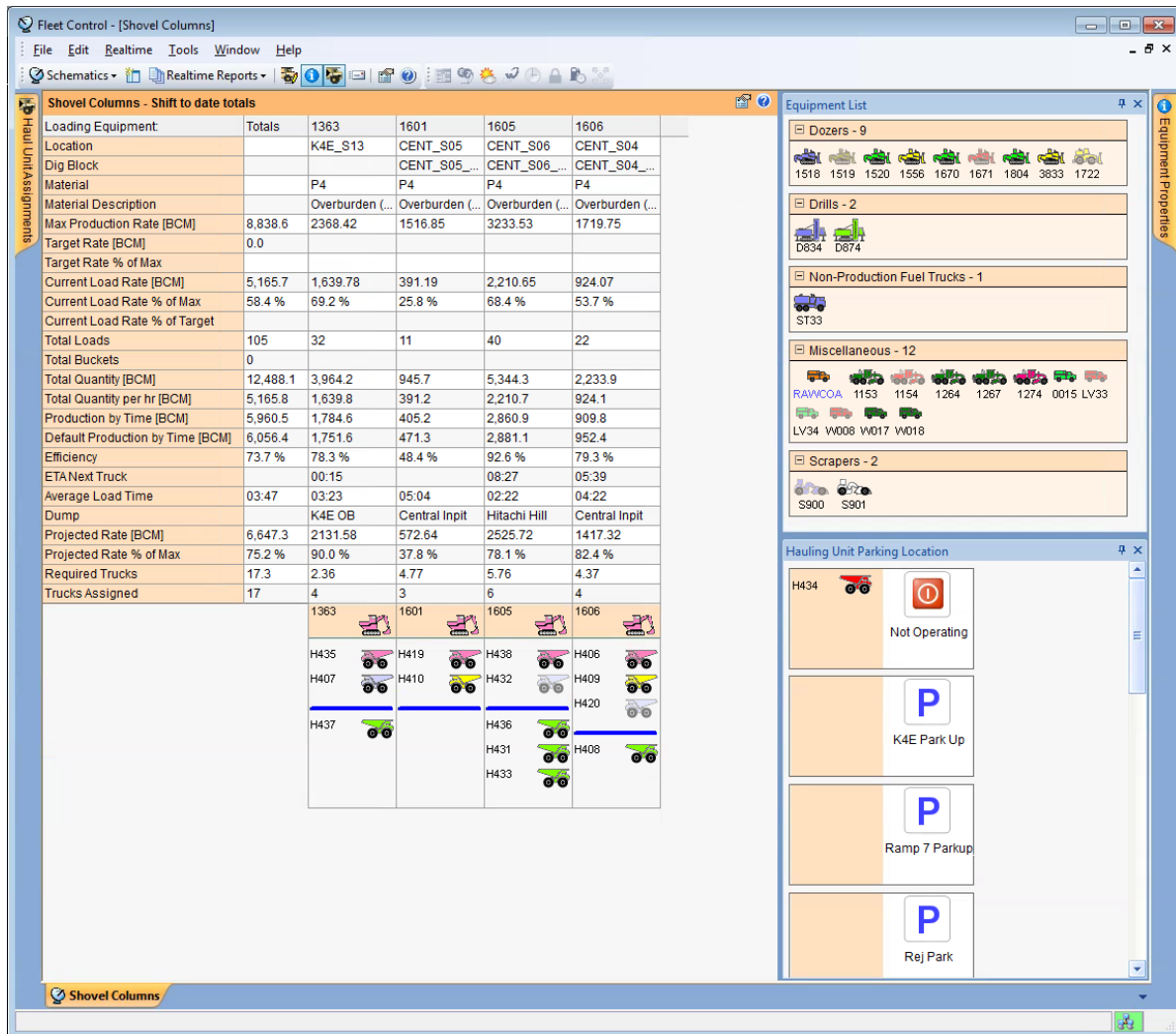


Figure 26 - Fleet Control

## **Appendix C: Code developed in C#**

## Appendix C1: program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace APpertain
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            new transmit().Show();
            new client().Show();
            Application.Run();
        }
    }
}
```

**Appendix C2: client.cs**

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using NativeWifi;
using MySql.Data.MySqlClient;

namespace APpertain
{
    public partial class client : Form
    {

#region Member Variables

        // Local gps variables
        Double Latitude;
        Double Longitude;
        String lDateTime;
        String data;
        String wifi_list;
        String sent = "no";
        String LocalId = System.Environment.MachineName;
        bool gpshealthy = true;

        // Local database config data
        String lserver = ConfigurationManager.AppSettings["lserver"].ToString();
        String luid = ConfigurationManager.AppSettings["luid"].ToString();
        String lpassword = ConfigurationManager.AppSettings["lpassword"].ToString();
        String ldatabase = ConfigurationManager.AppSettings["ldatabase"].ToString();
        DBConnect ldbConnect;

#endregion

#region Constructor

        public client()
        {
            InitializeComponent();

            try
            {
                serialPort1.Open();
            }
            catch (Exception ex)
            {
                //MessageBox.Show(ex.Message);
                GPSstream.Text = ex.Message;
                //timer1.Enabled = false;
                //button1.Text = "Run";
                return;
            }

            ldbConnect = new DBConnect(lserver,ldatabase,luid,lpassword);
        }
    }
}

```



```

#endregion

#region GPSData
// Update GPS location and UTC
// Successful Update then scan WIFI
// Available WIFI list created
// MySQL DB updated with current GPS location with all WIFI data
private void timer1_Tick(object sender, EventArgs e)
{
    if (serialPort1.IsOpen)
    {
        // Any failure of the TRY block will result in a GPS unhealthy result with error response as per CATCH block

        try
        {
            // Read in serial stream from GPS receiver
            data = serialPort1.ReadExisting();
            GPSstream.Text = data;
            // Separate GPS data into NMEA sentences
            String[] strArr = data.Split('$');

            // Cycle through each sentence seperating each value
            for (int i = 0; i < strArr.Length; i++)
            {
                String strTemp = strArr[i];
                String[] lineArr = strTemp.Split(',');

                #region GLL
                // Latitude Longitude data in GLL sentence
                if (lineArr[0] == "GPGLL")
                {
                    // Retrieve Latitude
                    Double dLat = Convert.ToDouble(lineArr[1]);

                    // North vs South
                    if (lineArr[2].ToString() == "S")
                        dLat = dLat / -100; //South -'ve
                    else
                        dLat = dLat / 100; //North +'ve

                    // Convert from Deg.Mins to Deg
                    String[] lat = dLat.ToString().Split('.');
                    String sLat = lat[0].ToString() + "." + ((Convert.ToDouble(lat[1]) / 60)).ToString("#####");
                    Latitude = Convert.ToDouble(sLat);

                    // Retrieve Longitude
                    Double dLon = Convert.ToDouble(lineArr[3]);

                    //East vs West
                    if (lineArr[4].ToString() == "W")
                        dLon = dLon / -100; //West -'ve
                    else
                        dLon = dLon / 100; //East +'ve

                    // Convert from Deg.Mins to Deg
                    String[] lon = dLon.ToString().Split('.');
                    String sLon = lon[0].ToString() + "." + ((Convert.ToDouble(lon[1]) / 60)).ToString("#####");
                    Longitude = Convert.ToDouble(sLon);

                    //update display
                    txtLat.Text = sLat;
                    txtLong.Text = sLon;
                }
            }
        }
    }
}

```

```

    }
    #endregion

    #region ZDA
    else if (lineArr[0] == "GPZDA")
    {
        //get datetime format 'YYYY-MM-DD HH:MM:SS' for MySql
        //GPZDA 'HHMMSS.ss,DD,MM,YYYY,00,00...
        String[] lTime = (lineArr[1]).Split('.');
        String hms = lTime[0];
        String hh = Convert.ToString(hms[0]) + Convert.ToString(hms[1]);
        String mm = Convert.ToString(hms[2]) + Convert.ToString(hms[3]);
        String ss = Convert.ToString(hms[4]) + Convert.ToString(hms[5]);
        IDateTime = lineArr[4] + "-" + lineArr[3] + "-" + lineArr[2] + " " + hh + ":" + mm + ":" + ss;
        //update display
        txtDateTime.Text = IDateTime;

        } //end ZDA sentence
    #endregion
}
// If the GPS stream is empty, then GPS not healthy
if (String.IsNullOrEmpty(data))
{
    gpshealthy = false;
    Ind1.BackColor = Color.Red;
    txtLat.Text = "No GPS";
    txtLong.Text = "No GPS";
    txtDateTime.Text = "No GPS";
    GPSstream.Text = "Receiver Disconnected";
}
else
{
    gpshealthy = true;
    Ind1.BackColor = Color.Lime;
}
}
catch
{
    gpshealthy = false;
    Ind1.BackColor = Color.Red;

    //Can't Read GPS values
    txtLat.Text = "No GPS";
    txtLong.Text = "No GPS";
    txtDateTime.Text = "No GPS";
}

}

// scan wifi and store results as string
wifi_list = getWiFiList();

// If GPS data is healthy add results to local database
if (gpshealthy)
{
    try
    {
        ldbConnect.Insert(LocalId, IDateTime, Latitude, Longitude, wifi_list, sent);
        // Local database success in storing data
        Ind2.BackColor = Color.Lime;
    }
}
catch

```

```

        {
            //Local database failure to store data
            Ind2.BackColor = Color.Red;
        }
    }
}
else
{
    // If COM port not connected
    txtDateTime.Text = "COM Port Closed";
    txtLat.Text = "COM Port Closed";
    txtLong.Text = "COM Port Closed";
    Ind1.BackColor = Color.Red;
    try
    {
        serialPort1.Open();
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
        //timer1.Enabled = false;
        //button1.Text = "Run";
        //return;
    }
} // end serial port open

// Clear out data block
data = "";

} // end timer
#endregion

#region WiFi
// Wireless network interface scanner
// Retrieve SSID,RSSI,BSS(mac) and Channel for each available network
// Returns all AP data in a string.
// Parsing available between network with '|' eg SSID1.SSID2.SSID3 etc.
// Parsing available between network details with ',' eg ssid,mac,rssi,channel.
String getWiFiList()
{
    lstNetworks.Items.Clear();
    String wifi;
    try
    {
        using (var client = new WlanClient())
        {
            //clear wifi list
            wifi = "";
            foreach (WlanClient.WlanInterface wlanIface in client.Interfaces)
            {
                Wlan.WlanBssEntry[] APs = wlanIface.GetNetworkBssList();
                foreach (Wlan.WlanBssEntry AP in APs)
                {
                    Wlan.Dot11Ssid ssid = AP.dot11Ssid;
                    byte[] bss = AP.dot11Bssid;
                    String MAC = BitConverter.ToString(bss);
                    String networkName = Encoding.ASCII.GetString(ssid.SSID, 0, (int)ssid.SSIDLength);
                    int networkQual = AP.rssi;
                    Double networkCh = Math.Round((0.0002 * (AP.chCenterFrequency) - 481.4));

                    //update window list

```

```

        ListViewItem item = new ListViewItem(networkName);
        item.SubItems.Add(networkCh.ToString());
        item.SubItems.Add(networkQual + " dBm");
        item.SubItems.Add(MAC);
        lstNetworks.Items.Add(item);
        //build wifi list as linear string
        wifi = wifi + networkName + "," + MAC + "," + Convert.ToString(networkQual) + "," +
Convert.ToString(networkCh) + ".";
    }
}
}
}
catch
{
    wifi = "";
}
return wifi;
} //end getWifiList()
#endregion

#region RunControl
private void button1_Click(object sender, EventArgs e)
{
    if (timer1.Enabled == true)
    {
        timer1.Enabled = false;
    }
    else
    {
        timer1.Enabled = true;
    }

    if (button1.Text == "Run")
        button1.Text = "Pause";
    else
        button1.Text = "Run";
}
#endregion

# region MenuItems

#region Exit
//Exit the application
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Dispose();
    Application.Exit();
}

private void button2_Click(object sender, EventArgs e)
{
    this.Dispose();
    Application.Exit();
}
#endregion

# region ComPorts
//User COM port selection. Default COM4
//User select COM port 1
private void toolStripMenuItem2_Click(object sender, EventArgs e)
{
    try

```

```

    {
        serialPort1.Close();
        serialPort1.PortName = "COM1";
        serialPort1.Open();
        foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
    }
}

//User select COM port 2
private void toolStripMenuItem3_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.Close();
        serialPort1.PortName = "COM2";
        serialPort1.Open();
        foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
    }
}

//User select COM port 3
private void toolStripMenuItem4_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.Close();
        serialPort1.PortName = "COM3";
        serialPort1.Open();
        foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
    }
}

//User select COM port 4
private void toolStripMenuItem5_Click(object sender, EventArgs e)
{
    try

```

```
{
    serialPort1.Close();
    serialPort1.PortName = "COM4";
    serialPort1.Open();
    foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}
catch (Exception ex)
{
    //MessageBox.Show(ex.Message);
    GPSstream.Text = ex.Message;
}
}

//User select COM port 5
private void toolStripMenuItem6_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.Close();
        serialPort1.PortName = "COM5";
        serialPort1.Open();
        foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
    }
}

//User select COM port 6
private void toolStripMenuItem7_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.Close();
        serialPort1.PortName = "COM6";
        serialPort1.Open();
        foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
    }
}

//User select COM port 7
private void toolStripMenuItem8_Click(object sender, EventArgs e)
{
    try
```

```

    {
        serialPort1.Close();
        serialPort1.PortName = "COM7";
        serialPort1.Open();
        foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
    }
}

//User select COM port 8
private void toolStripMenuItem9_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.Close();
        serialPort1.PortName = "COM8";
        serialPort1.Open();
        foreach (ToolStripMenuItem item in cOMPortToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
    catch (Exception ex)
    {
        //MessageBox.Show(ex.Message);
        GPSstream.Text = ex.Message;
    }
}
#endregion

#region SampleTimes
//-----
//Sampling by default is 1000ms
//User can set sampling as 1,2,5,10,30,60 secs as required
//Set sampling period to 1000ms
private void toolStripMenuItem10_Click(object sender, EventArgs e)
{
    timer1.Interval = 1000;
    foreach (ToolStripMenuItem item in timerMsToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}

//Set sampling period to 2000ms
private void toolStripMenuItem11_Click(object sender, EventArgs e)
{
    timer1.Interval = 2000;
    foreach (ToolStripMenuItem item in timerMsToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
}

```

```
        ((ToolStripMenuItem)sender).Checked = true;
    }

    //Set sampling period to 5000ms
    private void toolStripMenuItem12_Click(object sender, EventArgs e)
    {
        timer1.Interval = 5000;
        foreach (ToolStripMenuItem item in timerMsToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }

    //Set sampling period to 10000ms
    private void toolStripMenuItem13_Click(object sender, EventArgs e)
    {
        timer1.Interval = 10000;
        foreach (ToolStripMenuItem item in timerMsToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }

    //Set sampling period to 5000ms
    private void toolStripMenuItem14_Click(object sender, EventArgs e)
    {
        timer1.Interval = 30000;
        foreach (ToolStripMenuItem item in timerMsToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }

    //Set sampling period to 60000ms
    private void toolStripMenuItem15_Click(object sender, EventArgs e)
    {
        timer1.Interval = 60000;
        foreach (ToolStripMenuItem item in timerMsToolStripMenuItem.DropDownItems)
        {
            item.Checked = false;
        }
        ((ToolStripMenuItem)sender).Checked = true;
    }
}
#endregion

#endregion

} //end form
} //end namespace
```



## Appendix C3: transmit.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.Threading.Tasks;

// transit.cs utilises the DBconnect class to access and transfer data from a local
// mysql database to a second mysql database.
// The server connection string information is located in the exe.info config file

namespace APpertain
{
    public partial class transmit : Form
    {
        //Member variables

        //local database config data
        String lserver = ConfigurationManager.AppSettings["lserver"].ToString();
        String luid = ConfigurationManager.AppSettings["luid"].ToString();
        String lpassword = ConfigurationManager.AppSettings["lpassword"].ToString();
        String ldatabase = ConfigurationManager.AppSettings["ldatabase"].ToString();
        DBConnect ldbConnect;

        //remote database config data
        String rserver = ConfigurationManager.AppSettings["rserver"].ToString();
        String ruid = ConfigurationManager.AppSettings["ruid"].ToString();
        String rpassword = ConfigurationManager.AppSettings["rpassword"].ToString();
        String rdatabase = ConfigurationManager.AppSettings["rdatabase"].ToString();
        DBConnect rdbConnect;

        #region Constructor
        public transmit()
        {
            InitializeComponent();
            // create new local and remote database objects
            ldbConnect = new DBConnect(lserver, ldatabase, luid, lpassword);
            rdbConnect = new DBConnect(rserver, rdatabase, ruid, rpassword);
        }
        #endregion

        private void timer1_Tick(object sender, EventArgs e)
        {
            textBox1.Text = "Transmitting...";

            // open connection to remote and local databases
            // refer mysqlDB.cs for transferdata method
            // any failure to open connections, transfer data or update will cause failure
            if (rdbConnect.TransferData(ldbConnect))
            {
                Ind2.BackColor = Color.Lime;
                textBox1.Text = "Tx Success";
            }
        }
    }
}

```

```

else
{
    textBox1.Text = "Tx Failed!";
    Ind2.BackColor = Color.Red;
}
}

#region RunControl
// user interface to control program operation
private void button1_Click(object sender, EventArgs e)
{
    if (timer1.Enabled == true)
    {
        timer1.Enabled = false;
        Ind1.BackColor = Color.Red;
    }
    else
    {
        timer1.Enabled = true;
        Ind1.BackColor = Color.Lime;
    }

    if (button1.Text == "Run")
        button1.Text = "Pause";
    else
        button1.Text = "Run";
}
#endregion

#region exit
// user interface to exit the application
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    rdbConnect.Dispose();
    this.Dispose();
}

private void button2_Click(object sender, EventArgs e)
{
    rdbConnect.Dispose();
    this.Dispose();
}
#endregion

#region transmit_time
// user interface to select transmission attempt cycle
private void toolStripMenuItem2_Click(object sender, EventArgs e)
{
    timer1.Interval = 5000;
    foreach (ToolStripMenuItem item in timerToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}

private void toolStripMenuItem3_Click(object sender, EventArgs e)
{
    timer1.Interval = 10000;
    foreach (ToolStripMenuItem item in timerToolStripMenuItem.DropDownItems)

```

```
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}

private void toolStripMenuItem4_Click(object sender, EventArgs e)
{
    timer1.Interval = 15000;
    foreach (ToolStripMenuItem item in timerToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}

private void toolStripMenuItem5_Click(object sender, EventArgs e)
{
    timer1.Interval = 20000;
    foreach (ToolStripMenuItem item in timerToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}

private void toolStripMenuItem6_Click(object sender, EventArgs e)
{
    timer1.Interval = 30000;
    foreach (ToolStripMenuItem item in timerToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}

private void toolStripMenuItem7_Click(object sender, EventArgs e)
{
    timer1.Interval = 60000;
    foreach (ToolStripMenuItem item in timerToolStripMenuItem.DropDownItems)
    {
        item.Checked = false;
    }
    ((ToolStripMenuItem)sender).Checked = true;
}
#endregion
}
}
```

## Appendix C4: mysqlDB.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Data;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace APpertain
{
    public class DBConnect : IDisposable
    {
        private bool disposed = false;
        private MySqlConnection DBconn;
        private string server;
        private string database;
        private string uid;
        private string password;
        //private DataTable dataTable;

        #region Constructor
        public DBConnect(string s, string d, string u, string p)
        {
            Initialize(s, d, u, p);
        }

        // Initialise the object with the required mysql connection string information
        private void Initialize(string s, string d, string u, string p)
        {
            try
            {
                server = s;
                database = d;
                uid = u;
                password = p;
                string strconn = "Server=" + server + ";" + "uid=" + uid + ";" + "Password=" + password +
                    ":" + "Database=" + database + ";";
                DBconn = new MySqlConnection(strconn);
            }
            catch
            {
                System.Windows.Forms.MessageBox.Show("Invalid username/password, please try again");
            }
        }

        #endregion

        // Open the connection
        private bool OpenConnection()
        {
            try
            {
                DBconn.Open();
                return true;
            }
            catch
            {

```

```

        return false;
    }
}

// Close the connection
private bool CloseConnection()
{
    try
    {
        DBconn.Close();
        return true;
    }
    catch
    {
        return false;
    }
}

// Insert new gps and wifi data from client
public void Insert(String id, String T, Double Lt, Double Lg, String W, String S)
{
    string strInsert = "INSERT INTO
SurveyResult(id,ts,lat,lon,wifi,sent)VALUES(@id,@ts,@lat,@lon,@wifi,@sent)";
    if (this.OpenConnection() == true)
    {
        //create command and assign the query and connection from the constructor
        MySqlCommand MySqlCommand;
        MySqlCommand = DBconn.CreateCommand();
        MySqlCommand.CommandText = strInsert;
        MySqlCommand.Parameters.AddWithValue("@id", id);
        MySqlCommand.Parameters.AddWithValue("@ts", T);
        MySqlCommand.Parameters.AddWithValue("@lat", Lt);
        MySqlCommand.Parameters.AddWithValue("@lon", Lg);
        MySqlCommand.Parameters.AddWithValue("@wifi", W);
        MySqlCommand.Parameters.AddWithValue("@sent", S);
        MySqlCommand.ExecuteNonQuery();
    }
    //close connection
    this.CloseConnection();
}

// add data to the remote mysql database.
public void Insert(String id, String T, Double Lt, Double Lg, String W)
{
    string strInsert = "INSERT INTO SurveyResult(id,ts,lat,lon,wifi)VALUES(@id,@ts,@lat,@lon,@wifi)";
    if (this.OpenConnection() == true)
    {
        //create command and assign the query and connection from the constructor
        MySqlCommand MySqlCommand;
        MySqlCommand = DBconn.CreateCommand();
        MySqlCommand.CommandText = strInsert;
        MySqlCommand.Parameters.AddWithValue("@id", id);
        MySqlCommand.Parameters.AddWithValue("@ts", T);
        MySqlCommand.Parameters.AddWithValue("@lat", Lt);
        MySqlCommand.Parameters.AddWithValue("@lon", Lg);
        MySqlCommand.Parameters.AddWithValue("@wifi", W);
        MySqlCommand.ExecuteNonQuery();
    }
    //close connection
    this.CloseConnection();
}

//Update local db for successful sent data central server

```

```

public void Update(int entry)
{
    string strUpdate = "UPDATE SurveyResult SET sent='YES' WHERE entry=" + entry;
    //Open connection
    if (this.OpenConnection() == true)
    {
        //create mysql command
        MySqlCommand MySqlCmd = new MySqlCommand(strUpdate, DBconn);
        MySqlCmd.ExecuteNonQuery();
    }
    //close connection
    this.CloseConnection();
}

// Generate filtered table for export based on unsent data in local mysql database
public DataTable GetData()
{
    string strSelect = "SELECT * FROM SurveyResult WHERE sent !='YES'";
    using (DataTable dataTable = new DataTable())
    {
        if (this.OpenConnection() == true)
        {
            using (MySqlCommand MySqlCmd = new MySqlCommand(strSelect, DBconn))
            {
                MySqlCmd.ExecuteNonQuery();
                using (MySqlDataAdapter adapt = new MySqlDataAdapter(MySqlCommand))
                {
                    adapt.Fill(dataTable);
                }
            }
        }
    }
    //close connection
    this.CloseConnection();
    //return results of unsent data
    return dataTable;
}

// transfer unsent data from local DB to remote DB
// as each entry in unsent list is successfully inserted to remote DB
// the id data allows reference back to local DB for update to sent = YES
public bool TransferData(DBConnect tx)
{
    if (this.OpenConnection() == true)
    {
        DataTable buffer = new DataTable();
        buffer = tx.GetData();
        foreach (DataRow row in buffer.Rows)
        {
            String id = row["id"].ToString();
            String T = row["ts"].ToString();
            Double Lt = Convert.ToDouble(row["lat"]);
            Double Lg = Convert.ToDouble(row["lon"]);
            String W = row["wifi"].ToString();
            //transfer to destination
            this.Insert(id, T, Lt, Lg, W);

            //update source as successful
            int entry = Convert.ToInt32(row["entry"]);
            tx.Update(entry);
        }
    }
}

```

```

        buffer.Dispose();
        this.CloseConnection();
        return true;
    }
    else
    {
        this.CloseConnection();
        return false;
    }
}

// Delete data from database
public void Delete(int seq)
{
    string strDelete = "DELETE FROM SurveyResult WHERE seq = @seq";
    if (this.OpenConnection() == true)
    {
        MySqlCommand MySqlCommand = new MySqlCommand(strDelete, DBconn);
        MySqlCommand.ExecuteNonQuery();
    }
    //close connection
    this.CloseConnection();
}

public void Backup()
{
    //not implemented
}

public void Restore()
{
    // not implemented
}

//Implement IDisposable.
public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}
protected virtual void Dispose(bool disposing)
{
    if (!disposed)
    {
        {
            DBconn.Dispose();
            disposed = true;
        }
    }
}

// finalization code.
~DBConnect()
{
    // Simply call Dispose(false).
    Dispose(false);
}

} //End class
} //End namespace

```

## Appendix C5: Configuration File

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
  </configSections>
  <appSettings>
    <add key="lserver" value="localhost"/>
    <add key="luid" value="root"/>
    <add key="lpassword" value="$Project15"/>
    <add key="ldatabase" value="MySQL_client"/>
    <add key="rserver" value="10.215.135.70"/>
    <add key="ruid" value="root"/>
    <add key="rpassword" value="$Project15"/>
    <add key="rdatabase" value="MySQL_alldata"/>
  </appSettings>
  <connectionStrings>
    <add name="WiFi_GPS_Survey.Properties.Settings.Database1ConnectionString"
      connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Database1.mdf;Integrated Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
  <startup>

    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0,Profile=Client"/></startup>
</configuration>
```



## **Appendix D: Code developed in PHP**

## Appendix D1: gpsdata1.php

```
<?php
```

```
?>
```

```
<!DOCTYPE html >
```

```
<html>
```

```
<head>
```

```
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
```

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
```

```
<title>Wifi survey</title>
```

```
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js"></script>
```

```
<script type="text/javascript">
```

```
//<![CDATA[
```

```
var customColour = {
```

```
  "": {
```

```
    colour: '#AA0000'
```

```
  }
```

```
};
```

```
var customIcon = {
```

```
  "": {
```

```
    icon: 'http://labs.google.com/ridefinder/images/mm_20_red.png'
```

```
  }
```

```
};
```

```
function load() {
```

```
  var map = new google.maps.Map(document.getElementById("map"), {
```

```
    center: new google.maps.LatLng(-26.81, 151.90),
```

```
    zoom: 14,
```

```
    mapTypeId: 'satellite'
```

```
  });
```

```
  var infoWindow = new google.maps.InfoWindow;
```

```
  // Retrieve the XML data and prepare for Google Map Marker
```

```
  downloadUrl("phpsqlajax_genxml.php", function(data) {
```

```
    var xml = data.responseXML;
```

```
    var markers = xml.documentElement.getElementsByTagName("marker");
```

```
    for (var i = 0; i < markers.length; i++) {
```

```
      var id = markers[i].getAttribute("id");
```

```
      var ts = markers[i].getAttribute("ts");
```

```
      var lat = markers[i].getAttribute("lat");
```

```
      var lon = markers[i].getAttribute("lon");
```

```
      //var wifi = markers[i].getAttribute("wifi");
```

```
      var wifi = markers[i].getAttribute("wifi");
```

```
      var point = new google.maps.LatLng(
```

```
        parseFloat(markers[i].getAttribute("lat")),
```

```
        parseFloat(markers[i].getAttribute("lon")));
```

```
      var html = "<b>" + id + "</b> <br/>" + lat + "<br/>" + lon + "<br/>" + wifi;
```

```
      // alter colour of marker and circle based on wifi or no wifi data.
```

```
      var icon = customIcon[wifi] || { icon: 'http://labs.google.com/ridefinder/images/mm_20_green.png'}
```

```
        var colour = customColour[wifi] || { colour: '#00933B'};
```

```
      // Marker creation for each GPS way point
```

```
      var marker = new google.maps.Marker({
```

```
        map: map,
```

```
        position: point,
```

```
        icon: icon.icon
```

```
      });
```

```
      // Circle creation to estimate coverage at each waypoint
```

```
      var circle = new google.maps.Circle({
```

```
        map: map,
```

```

        radius: 10, // metres
        fillColor: colour.color, //'#AA0000',
        strokeColor: colour.colour,
        fillOpacity: 0.1,
        strokeOpacity: 0.1,
    });
    circle.bindTo('center', marker, 'position');
    bindInfoWindow(marker, map, infoWindow, html);
}
});
}

function bindInfoWindow(marker, map, infoWindow, html) {
    google.maps.event.addListener(marker, 'click', function() {
        infoWindow.setContent(html);
        infoWindow.open(map, marker);
    });
}

function downloadUrl(url, callback) {
    var request = window.ActiveXObject ?
        new ActiveXObject('Microsoft.XMLHTTP') :
        new XMLHttpRequest;

    request.onreadystatechange = function() {
        if (request.readyState == 4) {
            request.onreadystatechange = doNothing;
            callback(request, request.status);
        }
    };

    request.open('GET', url, true);
    request.send(null);
}

</script>

</head>

<body onload="load()">
    <div id="map" style="width: 1000px; height: 600px"></div>
</body>

</html>

```

## Appendix D2: phpsqlajax\_genxml.php

This file accesses the database and creates an XML file with all the filtered information. The XML file is then read by the main php file that presents the markers on a Google Map.

```

<?php
require 'phpsqlajax_dbinfo.php';

function parseToXML($htmlStr)
{
$xmlStr=str_replace('<','&lt;',$htmlStr);
$xmlStr=str_replace('>','&gt;',$xmlStr);
$xmlStr=str_replace('"','&quot;',$xmlStr);
$xmlStr=str_replace("'",'&#39;',$xmlStr);
$xmlStr=str_replace("&","&amp;",$xmlStr);
return $xmlStr;
}
// Opens a connection to a mysql server
$con=mysqli_connect ($host,$user,$pswd,$db);
if (!$con) {
    die('Not connected : ' . mysqli_error());
}
// Set the active mysql database
$db_selected = mysqli_select_db($con,'MySQL_alldata');
if (!$db_selected) {
    die ('Can\'t use db : ' . mysqli_error());
}

// Select all the rows in the survey results table
$query = "SELECT * FROM mysql_alldata.surveyresult";
$result = mysqli_query($con,$query);
if (!$result) {
    die('Invalid query: ' . mysqli_error($con));
}

header("Content-type: text/xml");

// Start XML file, echo parent node
echo '<markers>';

// Iterate through the rows, printing XML nodes for each
while ($row = @mysqli_fetch_assoc($result)){
    // ADD TO XML DOCUMENT NODE
    echo '<marker ' ;
    echo 'id="' . $row['id'] . "' ";
    echo 'ts="' . $row['ts'] . "' ";
    echo 'lat="' . $row['lat'] . "' ";
    echo 'lon="' . $row['lon'] . "' ";
    //echo 'ssid="' . $row['ssid'] . "' ";
    //echo 'mac="' . $row['mac'] . "' ";
    //echo 'rssi="' . $row['rssi'] . "' ";
    //echo 'channel="' . $row['channel'] . "' ";
    echo 'wifi="' . $row['wifi'] . "' ";
    echo '>';
}

// End XML file
echo '</markers>';

?>

```

### Appendix D3: phpsqlajax\_dbinfo.php

The username and password to the database as separated into another php file to provide a level of security. When the main php file opened from a web browser, the user has no visibility to this information; they only see the reference file name which the programmer would place in a secure location to prevent access.

```
<?php
$host = 'localhost';
$user = 'root';
$pswd = '$Pr0ject15';
$db = 'MySQL_alldata';
?>
```

## **Appendix E: MySQL Configuration**

## Appendix E1: MySQL Server Configuration

```

# Other default tuning values
# MySQL Server Instance Configuration File
# -----
# Generated by the MySQL Server Instance Configuration Wizard
#
#
# Installation Instructions
# -----
#
# On Linux you can copy this file to /etc/my.cnf to set global options,
# mysql-data-dir/my.cnf to set server-specific options
# (@localstatedir@ for this installation) or to
# ~/.my.cnf to set user-specific options.
#
# On Windows you should keep this file in the installation directory
# of your server (e.g. C:\Program Files\MySQL\MySQL Server X.Y). To
# make sure the server reads the config file use the startup option
# "--defaults-file".
#
# To run run the server from the command line, execute this in a
# command line shell, e.g.
# mysqld --defaults-file="C:\Program Files\MySQL\MySQL Server X.Y\my.ini"
#
# To install the server as a Windows service manually, execute this in a
# command line shell, e.g.
# mysqld --install MySQLXY --defaults-file="C:\Program Files\MySQL\MySQL Server X.Y\my.ini"
#
# And then execute this in a command line shell to start the server, e.g.
# net start MySQLXY
#
#
# Guildlines for editing this file
# -----
#
# In this file, you can use all long options that the program supports.
# If you want to know the options a program supports, start the program
# with the "--help" option.
#
# More detailed information about the individual options can also be
# found in the manual.
#
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.6/en/server-configuration-defaults.html
#
#
# CLIENT SECTION
# -----
#
# The following options will be read by MySQL client applications.
# Note that only client applications shipped by MySQL are guaranteed
# to read this section. If you want your own MySQL client program to
# honor these values, you need to specify it as an option during the

```

```
# MySQL client library initialization.
#
[client]
no-beep

# pipe
# socket=0.0
port=3306

[mysql]

default-character-set=utf8

# SERVER SECTION
# -----
#
# The following options will be read by the MySQL Server. Make sure that
# you have installed the server correctly (see above) so it reads this
# file.
#
# server_type=3
[mysqld]

# The next three options are mutually exclusive to SERVER_PORT below.
# skip-networking

# enable-named-pipe

# shared-memory

# shared-memory-base-name=MYSQL

# The Pipe the MySQL Server will use
# socket=MYSQL

# The TCP/IP Port the MySQL Server will listen on
port=3306

# Path to installation directory. All paths are usually resolved relative to this.
# basedir="C:/Program Files/MySQL/MySQL Server 5.6/"

# Path to the database root
datadir=C:/ProgramData/MySQL/MySQL Server 5.6/Data

# The default character set that will be used when a new schema or table is
# created and no character set is defined
character-set-server=utf8

# The default storage engine that will be used when create new tables when
default-storage-engine=INNODB

# Set the SQL mode to strict
```



```
sql-
mode="STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
"

# Enable Windows Authentication
# plugin-load=authentication_windows.dll

# General and Slow logging.
log-output=FILE
general-log=0
general_log_file="GLENN_LAPTOP.log"
slow-query-log=1
slow_query_log_file="GLENN_LAPTOP-slow.log"
long_query_time=10

# Binary Logging.
# log-bin

# Error Logging.
log-error="GLENN_LAPTOP.err"

# Server Id.
server-id=1

# The maximum amount of concurrent sessions the MySQL server will
# allow. One of these connections will be reserved for a user with
# SUPER privileges to allow the administrator to login even if the
# connection limit has been reached.
max_connections=151

# Query cache is used to cache SELECT results and later return them
# without actual executing the same query once again. Having the query
# cache enabled may result in significant speed improvements, if your
# have a lot of identical queries and rarely changing tables. See the
# "Qcache_lowmem_prunes" status variable to check if the current value
# is high enough for your load.
# Note: In case your tables change very often or if your queries are
# textually different every time, the query cache may result in a
# slowdown instead of a performance improvement.
query_cache_size=0

# The number of open tables for all threads. Increasing this value
# increases the number of file descriptors that mysqld requires.
# Therefore you have to make sure to set the amount of open files
# allowed to at least 4096 in the variable "open-files-limit" in
# section [mysqld_safe]
table_open_cache=2000

# Maximum size for internal (in-memory) temporary tables. If a table
# grows larger than this value, it is automatically converted to disk
# based table This limitation is for a single table. There can be many
# of them.
tmp_table_size=33M

# How many threads we should keep in a cache for reuse. When a client
```

```
# disconnects, the client's threads are put in the cache if there aren't
# more than thread_cache_size threads from before. This greatly reduces
# the amount of thread creations needed if you have a lot of new
# connections. (Normally this doesn't give a notable performance
# improvement if you have a good thread implementation.)
thread_cache_size=10

**** MyISAM Specific options
# The maximum size of the temporary file MySQL is allowed to use while
# recreating the index (during REPAIR, ALTER TABLE or LOAD DATA INFILE.
# If the file-size would be bigger than this, the index will be created
# through the key cache (which is slower).
mysiam_max_sort_file_size=100G

# If the temporary file used for fast index creation would be bigger
# than using the key cache by the amount specified here, then prefer the
# key cache method. This is mainly used to force long character keys in
# large tables to use the slower key cache method to create the index.
mysiam_sort_buffer_size=58M

# Size of the Key Buffer, used to cache index blocks for MyISAM tables.
# Do not set it larger than 30% of your available memory, as some memory
# is also required by the OS to cache rows. Even if you're not using
# MyISAM tables, you should still set it to 8-64M as it will also be
# used for internal temporary disk tables.
key_buffer_size=8M

# Size of the buffer used for doing full table scans of MyISAM tables.
# Allocated per thread, if a full scan is needed.
read_buffer_size=64K
read_rnd_buffer_size=256K

**** INNODB Specific options ****
# innodb_data_home_dir=0.0

# Use this option if you have a MySQL server with InnoDB support enabled
# but you do not plan to use it. This will save memory and disk space
# and speed up some things.
# skip-innodb

# Additional memory pool that is used by InnoDB to store metadata
# information. If InnoDB requires more memory for this purpose it will
# start to allocate it from the OS. As this is fast enough on most
# recent operating systems, you normally do not need to change this
# value. SHOW INNODB STATUS will display the current amount used.
innodb_additional_mem_pool_size=5M

# If set to 1, InnoDB will flush (fsync) the transaction logs to the
# disk at each commit, which offers full ACID behavior. If you are
# willing to compromise this safety, and you are running small
# transactions, you may set this to 0 or 2 to reduce disk I/O to the
# logs. Value 0 means that the log is only written to the log file and
# the log file flushed to disk approximately once per second. Value 2
# means the log is written to the log file at each commit, but the log
# file is only flushed to disk approximately once per second.
```

```
innodb_flush_log_at_trx_commit=1
```

```
# The size of the buffer InnoDB uses for buffering log data. As soon as  
# it is full, InnoDB will have to flush it to disk. As it is flushed  
# once per second anyway, it does not make sense to have it very large  
# (even with long transactions).
```

```
innodb_log_buffer_size=3M
```

```
# InnoDB, unlike MyISAM, uses a buffer pool to cache both indexes and  
# row data. The bigger you set this the less disk I/O is needed to  
# access data in tables. On a dedicated database server you may set this  
# parameter up to 80% of the machine physical memory size. Do not set it  
# too large, though, because competition of the physical memory may  
# cause paging in the operating system. Note that on 32bit systems you  
# might be limited to 2-3.5G of user level memory per process, so do not  
# set it too high.
```

```
innodb_buffer_pool_size=180M
```

```
# Size of each log file in a log group. You should set the combined size  
# of log files to about 25%-100% of your buffer pool size to avoid  
# unneeded buffer pool flush activity on log file overwrite. However,  
# note that a larger logfile size will increase the time needed for the  
# recovery process.
```

```
innodb_log_file_size=48M
```

```
# Number of threads allowed inside the InnoDB kernel. The optimal value  
# depends highly on the application, hardware as well as the OS  
# scheduler properties. A too high value may lead to thread thrashing.
```

```
innodb_thread_concurrency=9
```

```
# The increment size (in MB) for extending the size of an auto-extend InnoDB system tablespace file  
when it becomes full.
```

```
innodb_autoextend_increment=64
```

```
# The number of regions that the InnoDB buffer pool is divided into.
```

```
# For systems with buffer pools in the multi-gigabyte range, dividing the buffer pool into separate  
instances can improve concurrency,  
# by reducing contention as different threads read and write to cached pages.
```

```
innodb_buffer_pool_instances=8
```

```
# Determines the number of threads that can enter InnoDB concurrently.
```

```
innodb_concurrency_tickets=5000
```

```
# Specifies how long in milliseconds (ms) a block inserted into the old sublist must stay there after its  
first access before
```

```
# it can be moved to the new sublist.
```

```
innodb_old_blocks_time=1000
```

```
# It specifies the maximum number of .ibd files that MySQL can keep open at one time. The  
minimum value is 10.
```

```
innodb_open_files=300
```

```
# When this variable is enabled, InnoDB updates statistics during metadata statements.
```

```
innodb_stats_on_metadata=0
```

```
# When innodb_file_per_table is enabled (the default in 5.6.6 and higher), InnoDB stores the data and
indexes for each newly created table
# in a separate .ibd file, rather than in the system tablespace.
innodb_file_per_table=1

# Use the following list of values: 0 for crc32, 1 for strict_crc32, 2 for innodb, 3 for strict_innodb, 4
for none, 5 for strict_none.
innodb_checksum_algorithm=0

# The number of outstanding connection requests MySQL can have.
# This option is useful when the main MySQL thread gets many connection requests in a very short
time.
# It then takes some time (although very little) for the main thread to check the connection and start a
new thread.
# The back_log value indicates how many requests can be stacked during this short time before
MySQL momentarily
# stops answering new requests.
# You need to increase this only if you expect a large number of connections in a short period of time.
back_log=80

# If this is set to a nonzero value, all tables are closed every flush_time seconds to free up resources
and
# synchronize unflushed data to disk.
# This option is best used only on systems with minimal resources.
flush_time=0

# The minimum size of the buffer that is used for plain index scans, range index scans, and joins that
do not use
# indexes and thus perform full table scans.
join_buffer_size=256K

# The maximum size of one packet or any generated or intermediate string, or any parameter sent by
the
# mysql_stmt_send_long_data() C API function.
max_allowed_packet=4M

# If more than this many successive connection requests from a host are interrupted without a
successful connection,
# the server blocks that host from performing further connections.
max_connect_errors=100

# Changes the number of file descriptors available to mysqld.
# You should try increasing the value of this option if mysqld gives you the error "Too many open
files".
open_files_limit=4161

# Set the query cache type. 0 for OFF, 1 for ON and 2 for DEMAND.
query_cache_type=0

# If you see many sort_merge_passes per second in SHOW GLOBAL STATUS output, you can
consider increasing the
# sort_buffer_size value to speed up ORDER BY or GROUP BY operations that cannot be improved
with query optimization
# or improved indexing.
sort_buffer_size=256K
```

```
# The number of table definitions (from .frm files) that can be stored in the definition cache.
# If you use a large number of tables, you can create a large table definition cache to speed up opening
of tables.
# The table definition cache takes less space and does not use file descriptors, unlike the normal table
cache.
# The minimum and default values are both 400.
table_definition_cache=1400

# Specify the maximum size of a row-based binary log event, in bytes.
# Rows are grouped into events smaller than this size if possible. The value should be a multiple of
256.
binlog_row_event_max_size=8K

# If the value of this variable is greater than 0, a replication slave synchronizes its master.info file to
disk.
# (using fdatsync()) after every sync_master_info events.
sync_master_info=10000

# If the value of this variable is greater than 0, the MySQL server synchronizes its relay log to disk.
# (using fdatsync()) after every sync_relay_log writes to the relay log.
sync_relay_log=10000

# If the value of this variable is greater than 0, a replication slave synchronizes its relay-log.info file to
disk.
# (using fdatsync()) after every sync_relay_log_info transactions.
sync_relay_log_info=10000
```

## Appendix E2: Local Database Design

```
CREATE DATABASE `mysql_client` /*!40100 DEFAULT CHARACTER SET utf8 */;
use mysql_client;
CREATE TABLE `surveyresult` (
  `entry` int(32) NOT NULL AUTO_INCREMENT,
  `id` varchar(60) NOT NULL,
  `ts` varchar(60) NOT NULL,
  `lat` float(10,6) NOT NULL,
  `lon` float(10,6) NOT NULL,
  `wifi` text,
  `sent` varchar(5) DEFAULT NULL,
  PRIMARY KEY (`entry`)
) ENGINE=InnoDB AUTO_INCREMENT=431 DEFAULT CHARSET=utf8;
```

### Appendix E3: Remote Database Design

```
CREATE DATABASE `mysql_alldata` /*!40100 DEFAULT CHARACTER SET utf8 */;
use mysql_alldata;
CREATE TABLE `surveyresult` (
  `entry` int(32) NOT NULL AUTO_INCREMENT,
  `id` varchar(60) NOT NULL,
  `ts` varchar(60) NOT NULL,
  `lat` float(10,6) NOT NULL,
  `lon` float(10,6) NOT NULL,
  `wifi` text,
  PRIMARY KEY (`entry`)
) ENGINE=InnoDB AUTO_INCREMENT=1421 DEFAULT CHARSET=utf8;
```