



Cape Peninsula
University of Technology

**Development of a remote wireless monitoring system for
large farms**

by

Adriaan Cornelius Rootman

Thesis submitted in fulfilment of the requirements for the degree

Master of Technology: Electrical Engineering

at the Cape Peninsula University of Technology

Supervisor: Mr Jacques Wheeler

Co-supervisor: Prof Ernst Uken

Cape Town

July 2012

CPUT copyright information

The dissertation/thesis may not be published either in part (in scholarly, scientific or technical journals), or as a whole (as a monograph), unless permission has been obtained from the University.

DECLARATION

I, Adriaan Rootman, declare that the contents of this dissertation/thesis represent my own unaided work and that the dissertation/thesis has not previously been submitted for academic examination towards any qualification. Furthermore, it represents my own opinions and not necessarily those of the Cape Peninsula University of Technology.

Signed

Date

ABSTRACT

This research project addresses the unique challenges of extensive farming in terms of monitoring and controlling remote equipment or events. Poorly maintained roads and escalating fuel costs increase difficulty of farming and the time spent on physically monitoring remote sites further reduces financial yields. The research showed that there are very few solutions that implement wireless or electronic technology to overcome the challenges associated with these isolated and arid areas and that a low-cost, long range wireless telemetry solution that is easy to use would be beneficial for the extensive farming industry. It was therefore the aim of this project to develop a remote monitoring and controlling solution that implements wireless technology to convey information of activities around the farm utilising electronic means.

To be able to successfully develop a wireless telemetry solution that will accurately meet the needs of this specific sector of industry, market research was conducted. To guide the research, the QFD (quality function deployment) process for product development has been implemented. The research consisted out of various aspects including a survey, financial considerations and international comparisons. The research also aided in the understanding of the day-to-day activities and also the physical parameters of extensive farms. Also, currently available technologies and products were evaluated to establish whether similarities exist that will aid in the development of a new product.

The development process was based on the results obtained in the market research and resulted in a wireless telemetry solution that overcame all the design challenges and proved to be technically feasible, successfully addressing the application requirements. Zigbee technology was utilized for wireless communication because it provided an off-the-shelf solution with a number of readily available development platforms from various technology providers. A communication range of up to 6 kilometres with a transmitted power of 11dBm was achieved for point-to-point communication and a mesh network topology has been implemented for even longer range and complete coverage on farms. Various types of measurements have been catered for, with custom-designed instrumentation which enabled measurements such as water levels, movement and analogue signals. Also, a basic user interface was developed to enable the user to monitor or control the equipment or events remotely from a personal computer, locally or even over the internet.

The results of this research project showed that by carefully selecting available technologies and understanding the application, it is possible to develop a solution that addresses the monitoring and controlling needs associated with extensive farming. The wireless telemetry system that was developed resulted in a saving equal to 10% of the total expenses of the farms per year. The telemetry system is therefore a financially feasible solution with a payback period of less than 1 year and far below the initial estimated budget. Without the need to physically monitoring equipment and events, an increase in productivity and the expansion of the overall enterprise is a further benefit added unto the monetary savings. In addition to the financial benefits of implementing new wireless technology, this is an opportunity to contribute to a cleaner and more sustained future as a legacy for the next generation by reducing the carbon footprint of the farm.

ACKNOWLEDGEMENTS

I wish to thank:

All the farmers who participated in this research project and filled in the survey. I also want to thank them for their patience while waiting for the results. This research project is focused on the farmers in rural areas and I believe the outcome of this research project will add value to their current enterprise.

The financial institutions in Calvinia and Loeriesfontein of the Northern Cape Province for providing thorough financial information that provided valuable insight for this project.

The Mauerberger foundation for a scholarship that financially enabled me to do this research project. Without this scholarship this project would not have been possible.

My wife, family, friends, and colleagues for continued support and help during the duration of this project.

My supervisor and co-supervisor for their invaluable input, support and guidance during this project.

Most of all I want to thank my Heavenly Father for the opportunity I had to complete my studies and glorify Him for all I have achieved. I also thank my friends in Christ Jesus for continually supporting and praying for me during this project.

TABLE OF CONTENTS

DECLARATION	i
ABSTRACT.....	ii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES	viii
LIST OF TABLES.....	x
GLOSSARY	xi
1. CHAPTER ONE.....	1
Introduction	1
1.1 Introduction.....	1
1.2 Research objectives and expected outcomes.....	1
1.3 Research questions.....	3
1.4 Research boundaries and scope	3
1.5 Research methodology	4
1.6 Thesis structure	4
2. CHAPTER TWO	6
Market research and literature review.....	6
2.1 Defining customer needs	6
2.2 International comparison.....	14
2.3 Comparing the results and estimating a budget	16
2.4 Currently available products and technologies.....	17
2.5 Literature review for technology to be implemented	18
2.6 Instrumentation	20
2.7 Wireless communications	22
2.8 Wireless network options that were considered	29
2.9 OSI (Open Systems Interconnection) model	34

2.10	User interface	34
3.	CHAPTER THREE.....	36
	Project Specifications	36
3.1	Defining product requirements: reference design	36
3.2	Project specifications: wireless communications	39
3.3	Zigbit 900	41
3.4	Project specifications: instrumentation	42
3.5	Project specifications: user interface	44
4.	CHAPTER FOUR.....	46
	System Design and Development.....	46
4.1	Zigbit 900 module and development platform.....	46
4.2	Wireless sensor network	48
4.3	Base station	50
4.4	User interface	57
4.5	End nodes and repeater	65
4.6	System cost	84
4.7	Instrumentation	87
5.	CHAPTER FIVE.....	88
	Test Results	88
5.1	Technical feasibility.....	88
5.2	Financial feasibility.....	92
5.3	QFD: User requirements	94
6.	CHAPTER SIX.....	95
	Conclusions and Recommendations	95
6.1	Conclusions	95
6.2	Recommendations	96
7.	REFERENCES	98
8.	APPENDICES.....	103
	Appendix 1	104

Questionnaire used for market survey 104

Appendix 2 106

Wireless telemetry technology comparison chart..... 106

Appendix 3 107

Zigbee comparison chart 107

Appendix 4 108

Embedded webserver application..... 108

Appendix 5 123

Sensor network server embedded code 123

Appendix 6 142

AA travel cost certificate 142

LIST OF FIGURES

Figure 2.1: Typical application examples of farms included in this study	7
Figure 2.2: Vegetation map of South Africa (South African Government, 2000)	8
Figure 2.3: Breakdown of expenses	13
Figure 2.4: Overall farming costs (Department of Primary Industries, 2009).....	15
Figure 2.5: Block diagram of a basic communications (telemetry) system (Patranabis, 1999, p. 3)	19
Figure 2.6: Updated block diagram for a wireless telemetry system.....	20
Figure 2.7: Representation of an instrument (Dreyer, 2001, p. 1).....	21
Figure 2.8: Analogue to digital converter (Dreyer, 2001, p. 11)	22
Figure 2.9: The electromagnetic spectrum (Microsoft Encarta online encyclopedia, 2008) .	28
Figure 2.10: Cell phone coverage map for South Africa (CellC, 2012)	31
Figure 2.11: Zigbee topologies (Le, 2005).....	33
Figure 3.1: Points of interest for the reference design (Google, 2012).....	37
Figure 3.2: System block diagram	39
Figure 3.3: End node block diagram.....	44
Figure 4.1: Zigbit 900 development platform	47
Figure 4.2: Zigbit 900 development platform block diagram	48
Figure 4.3: Functional block diagram of the base station.....	51
Figure 4.4: SerialNet architecture (Atmel Corporation, 2011)	52
Figure 4.5: Explorer 16 demonstration board from Microchip	53
Figure 4.6: SerialNet command executions (Atmel Corporation, 2011)	54
Figure 4.7: AT-commands demonstration	55
Figure 4.8: Sensor network server flow diagram	56
Figure 4.9: Web server flow diagram.....	58
Figure 4.10: Ethernet functional block diagram	59
Figure 4.11: Ethernet configuration	60
Figure 4.12: Status page of the user interface.....	61
Figure 4.13: Node edit page of the user interface.....	62
Figure 4.14: Date/Time edit page of the user interface.....	62
Figure 4.15: Logging configuration	64
Figure 4.16: Base station test set-up.....	65

Figure 4.17: End node test setup	66
Figure 4.18: Float level switches	68
Figure 4.19: Water level sensing	69
Figure 4.20: Water level sensor test setup	70
Figure 4.21: 555 timer circuit.....	72
Figure 4.22: Movement sensor test setup	74
Figure 4.23: Voltage divider	76
Figure 4.24: Voltage divider circuit	76
Figure 4.25: ADC test setup.....	77
Figure 4.26: Relay driver circuit.....	78
Figure 4.27: Relay driver prototype board	79
Figure 4.28: Relay monitoring circuit	80
Figure 4.29: Power supply	82
Figure 4.30: USB cable regulator for Zigbit 900.....	83
Figure 4.31: End node prototype unit	83
Figure 4.32: End node prototype unit	84
Figure 5.1: Range test results	90

LIST OF TABLES

Table 2.1: Physical parameters	10
Table 2.2: Labour cost calculations	12
Table 2.3: Local farm parameters and operating cost.....	14
Table 2.4: Intensive and extensive farming (Department of Primary Industries, 2009)	16
Table 2.5: Wireless communication comparison (The Zigbee Alliance, n.d.)	30
Table 2.6: Zigbee parameters (Farahani, 2008, p. 7)	32
Table 2.7: ISO 7-layer model (Technology Training Limited, 2012).....	34
Table 3.1: Reference project specifications.....	36
Table 3.2: Requirements vs. Shortcomings.....	40
Table 3.3: Zigbit 900 specifications (Atmel, 2009)	42
Table 3.4: Node types required for reference design.....	43
Table 4.1: Zigbit 900 specifications (Atmel Corporation, 2009).....	46
Table 4.2: Serial interface parameters	53
Table 4.3: Software development overview.....	60
Table 4.4: Liquid level sensing techniques	68
Table 4.5: Cost of base station (prototype).....	85
Table 4.6: Base station (final product).....	85
Table 4.7: Cost of repeater/end node (prototype).....	86
Table 4.8: Cost of repeater/end node (final product)	87
Table 4.9: Sensor cost	87
Table 5.1: Test conditions	90
Table 5.2: Estimated system cost.....	92
Table 5.3: Cost of physically monitoring sites (budget).....	93
Table 5.4: QFD results	94

GLOSSARY

*.csv	Character separated values
AA	Automobile association
AC	Alternating current
ADC	Analogue to digital converter
AM	Amplitude modulation
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASK	Amplitude shift keying
AT-commands	Serial communication standard
bit	A digital “1” or “0”
BPSK	Binary phase shift keying
CB radio	Citizen band
CDMA	Code division multiplexed access
DC	Direct current
DSSS	Direct sequence spread spectrum
EEPROM	Electrical erasable programmable read only memory
EIA	Communication standard
EIRP	Effective isotropic radiation power
EMF	Electro mechanical force
FFD	Full function device
FM	Frequency modulation
FSK	Frequency shift keying
GPIO	General purpose input/output
GPRS	General packet radio services
GSM	Global System for Mobile Communications
HART	Highway addressable remote transducer
HTTP	Hyper-text transfer protocol
ICASA	Independent communication association of South Africa
IEEE	The Institute of Electrical and Electronics Engineers
IP	Internet protocol
IRQ	Interrupt request
ISM	Instrument, Scientific, Medical

ISO Model	Open Systems Interconnection model
JPEG	Joint Photographic Experts Group
LAN	Local area network
LED	Light emitting diode
MAC	Media access control layer of ISO-model
MCU	Microcontroller unit
MP3	Audio file format
MSK	Minimum shift keying
O-QPSK	Offset-quadrature phase shift keying
PAN	Personal area network
PC	Personal computer
PCB	Printed circuit board
PHY	Physical layer of the ISO-model
PIR	Passive infrared
PM	Phase modulation
POI	Point of interest
PSK	Phase shift keying
PV	Photo-voltaic
PWM	Pulse width modulation
QFD	Quality function deployment
RF	Radio frequency
RFD	Reduced function device
RS232	Serial communication standard
SCADA	Supervisory Control and Data Acquisition
SD Card	Secure digital card
SPI	Serial peripheral interface
SRD	Short range devices
TCP	Transmission Control Protocol/
Telnet	Interconnection standard
TIA	Communication standard
TX/RX	Transmit/Receive
UART	Universal asynchronous receiver/transmitter
UDP	User Datagram Protocol
Wifi	Wireless local area network
ZIGBEE	Wireless sensor network protocol

CHAPTER ONE

Introduction

1.1 Introduction

When the unique challenges of extensive farming in terms of monitoring and controlling remote equipment or events are considered, it was found that there are very few solutions available that implement electronic techniques to overcome these challenges. In these isolated and arid areas, poorly maintained roads and escalating fuel costs increase difficulty of farming. Also, the time spent on physically monitoring remote sites further reduces financial yields.

Since monitoring and controlling of remote equipment and events is essential for effective farming, the use of wireless technology is therefore proposed to provide a solution to minimise the time and cost involved in physically monitoring and controlling activities. Currently available electronic and communication technology will form the basis of the research. However, research done by the author showed that the application of instrumentation and wireless technologies in the extensive farming industry is significantly different from the typical industrial or commercial application. Also, circumstances on these large farms do not allow the use of common technologies and standard solutions because of cost, lack of cell phone network coverage, and the long distances between monitoring points (G. Vellidis, 2007). Therefore, this project aims to combine various areas of research which should result in a new and innovative wireless telemetry solution ideally suited for extensive farming applications. This telemetry system will electronically monitor and control events or equipment from a central point in real-time to increase farming efficiency and reduce the current cost of farming (G. Vellidis, 2007).

1.2 Research objectives and expected outcomes

The objective of the research was to develop a wireless telemetry solution that will enable real-time, remote monitoring and controlling functionality for extensive farming applications. The development of the telemetry system was divided in three main sections.

Firstly, the wireless communication and protocol that needed to be employed. Secondly, the most suitable instrumentation required on site had to be identified. This instrumentation will gather the required information that will be relayed to the user. Thirdly, an effective user interface had to be chosen and developed.

Earlier research indicated that the following parameters should govern the planning and development stages of this project:

- low implementation cost
- off-the-shelf components
- low running cost
- extended RF (radio frequency) transmission range
- no licensing fees
- easy implementation and operation
- robust and reliable
- user interface options
- plug and play

The expected outcome of this project is a wireless telemetry system, complete with the electronic instrumentation, RF link and a user interface. These will assist the farmers in farming activities resulting in a reduction in costs such as labour and transport. This telemetry system should have lower capital and running costs than what is currently available and should result in wireless technology that can be used in areas with no cell phone or Wi-Fi coverage. The farmer will be able to cope with more processes simultaneously and thus be more efficient in spending valuable resources. This should benefit the overall enterprise and ultimately have a positive effect on the farming industry in South Africa. Also, if it were possible to enable the farmer to have real-time knowledge of the activities around the farm from a central point, the safety of the farmer can be increased.

In addition to the financial benefits from implementing new wireless technology, there is the opportunity to contribute to a cleaner and more sustained future as a legacy for the next generation by reducing the carbon footprint of the farm. In recent days, some industries of farming are often penalised for not engaging in more environment friendly farming practices.

Also, various incentives from some regulating authorities exist for farmers to utilise methods to reduce the carbon footprint of the farm. Sustainable farming practices in South Africa will pave the way for more farmers to realise the need for sustainable solutions (Hawkins, 2010).

1.3 Research questions

The research questions necessary to guide the research and to evaluate the results are as follows:

- Will it be technically feasible to develop a telemetry system that will meet the user requirements?
- By how much will this system impact the efficiency of farming practices?
- Will this system be economically viable in the chosen farming industry?
- Will it be possible to use standard off-the-shelf components and a generic protocol for this system?

1.4 Research boundaries and scope

Extensive farming usually depicts large farms with low yield due to limited resources, typically with a semi-desert climate that is subject to low seasonal rainfall. For the purposes of the project, an area within South Africa where extensive farming is commonly found was selected as the focus area. This area is also familiar to the author and resembles all the characteristics of extensive farms.

The aim of this research was not to develop a market-ready product with maximum functionality, but rather to investigate the possibilities of integrating various technologies into a wireless telemetry system that will be able to meet the unique requirements of the application.

Therefore, a telemetry system with basic yet sufficient control and monitoring functionality was the focus of this project. This telemetry system consists of a wireless communication link that would be able to transmit real-time data from the remote site over long distances to the user. Furthermore, to be able to measure the physical quantities or events at the remote sites, basic instrumentation needed to be implemented. Also, a user interface providing sufficient functionality to be able to control or monitor the components of the system was implemented in order to prove that the technology or method is feasible and sustainable.

1.5 Research methodology

Since this research project was aimed at the development of a novel idea in a unique market segment where, up to now, the use of wireless technology is very limited, a customer-focused development approach was implemented. This approach ensured that the result of the research project will be relevant to this specific sector in the farming industry.

To direct the research and development approach, the guidelines described in the quality function deployment (QFD) method was used. The QFD method implements four phases to ensure successful design and implementation of any typical new technology or product (Crow, 2002).

For the purposes of this project only the planning phase described in the QFD process was implemented in order to make sure that the specific customer needs are successfully addressed. The following steps were incorporated into the product planning phase:

- define and prioritise customer needs
- analyse competitive or prior generation products
- define projects requirements
- define design specifications

1.6 Thesis structure

Chapter 1

Chapter one provides background to the research project by discussing the research topic in terms of the need that has been identified. This chapter also illustrates how the research was guided for the outcome of this project to be significant and providing a relevant and novel solution. Furthermore, the research methodology is discussed together with the scope of the project to provide an understanding of the results obtained.

Chapter 2

This chapter presents the market research that was conducted with various methods, including the use of historical data and a survey.

Also, a literature and technology review is included, incorporating the theoretical aspects of wireless networking and that of the overall project, such as telemetry, instrumentation, and user interfaces.

The review was done with the expected outcomes in mind, hence focusing on the specific parameters of the project.

Chapter 3

In chapter three the product research that was done is discussed as part of the planning phases of the project. Also, a reference design was specified to use as guideline for this project that will result in a better defined, real-life set of user requirements and should ideally simplify the design. Also, technologies that were used for this project are defined.

Chapter 4

This chapter discusses the design, construction and implementation of the various aspects of the telemetry system. Furthermore, a thorough technical explanation of each aspect and the development thereof are provided to demonstrate technical feasibility of the system.

Chapter 5

After the research a prototype design was implemented and its components tested to verify that the research questions were answered and that the specifications were met. This chapter gives the results and discusses the feasibility of the project.

Chapter 6

Chapter six concludes that the implementation of the various technologies is feasible and can effectively overcome the challenges of extensive farming. In addition, further work is recommended to expand the functionality of the prototype system and to implement a pilot project. User patterns and behaviours could be monitored to investigate the success of the outcomes of this project.

CHAPTER TWO

Market research and literature review

2.1 Defining customer needs

Research was conducted to define and verify the important parameters of the extensive farming industry. This was necessary in order to define the requirements that will govern the development of a remote wireless control and monitoring system which could successfully be implemented in this farming industry. The research was structured as follows:

- identify the physical parameters of farms in the specified area
- identify the needs of the owners/workers currently operating on these farms
- estimate the current cost of farming operations within the defined parameters
- gather financial records to accurately establish the current cost of farming operations and compare these costs with estimations
- compare the available information for the local farming industry to an international source to validate the parameters above

The information mentioned in this section was obtained by the following means:

- a survey conducted in the focus area where this system will typically be implemented
- audited financial statements of farmers from the local financial institutions
- researching of local and international agricultural practices

The information was then processed to form the foundation for defining the project requirements and specifications. This was done to ensure that the monitoring and controlling system that was developed will accurately address the needs of the farming industry found in the rural and arid parts of South Africa.

2.1.1 Typical application examples

Figure 2.1 shows typical areas where this remote monitoring and controlling system will be implemented.



Figure 2.1: Typical application examples of farms included in this study

2.1.2 Survey

Region where the survey was conducted

Information was gathered by means of a survey conducted by the author in the Northern Cape province of South Africa. A total of 25 farms were included in the survey. Figure 2.2 indicates the area where the survey was conducted.

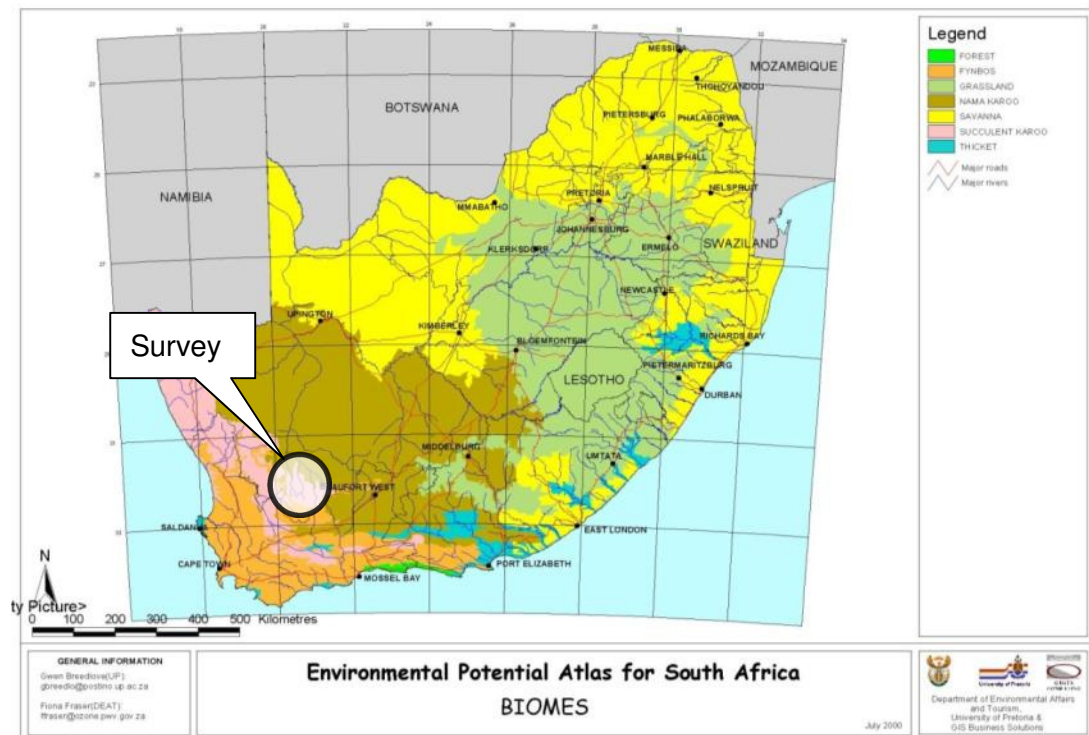


Figure 2.2: Vegetation map of South Africa (South African Government, 2000)

Why this region

Figure 2.2 shows that the area where the survey was conducted is predominately Karoo or semi-desert climate as indicated in brown on the map. The conditions typical to this area are:

- extensive and large farms
- very rural
- long distances and poorly maintained roads
- dry and arid climatic conditions
- limited communication solutions and no cell phone coverage

Furthermore, even though commercial farming is quite common in this area, the cost of farming is very high, resulting in very limited surplus cash flow needed to expand or improve activities around the farm that would add value to the enterprise. The cost of farming is discussed later in this section.

Reasons for conducting this survey

The aim of this survey was to retrieve accurate, real-life information of the actual activities on the farms from the local farmers in the region. The primary two reasons for this were:

- A thorough understanding of the needs of the farmers would enable the product development phase to accurately address these needs and ensure the development of an effective and relevant solution.
- Real-life information was necessary to make accurate estimations regarding the economic issues associated with the farming practices for this region. The results of the financial estimation were then applied in order to discuss the feasibility of a remote monitoring and controlling system.

How the survey was conducted

For the survey, 25 farmers were individually approached and asked to answer a series of questions (see questionnaire in Appendix 1). These questions were aimed at summarising the farmers' needs by trying to understand the activities the farmer spent most of the available time on and what seems to have high priority in the farming operations.

Physical parameters

Table 2.1 provides the averages of the physical parameters of the farms that were included in the survey. This provides a better idea of the physical and practical limitations that needed to be taken into account when the design specifications were considered. To summarise, the two main parameters that have an impact on the choice of technology are the size of the farm and the number of remote points of interest's (POI's) that need to be inspected. Furthermore, these parameters also had an impact on the financial feasibility of the solution that was developed.

Table 2.1: Physical parameters

Parameter	Specification
Average farm size	5 129 hectares
Average number of farms per farmer	2
Average POI's	13
Average distance to a POI from farmstead	4.2 km
Number of times each POI is inspected per week	3
Largest farm	8 000 hectares
Requires monitoring and controlling functionality	Yes
Average amount the farmer is willing to pay	R8 700

Also, two other important aspects were noted from the survey:

- In most cases the only method of communication is via CB (citizen band) radio and/or a Telkom landline.
- Secondly, the average farmer in this region is completely unaware of any system or solutions that could be used to wirelessly control and monitor remote equipment.

Controlling and monitoring requirements

From the survey it was established what activities and conditions the user required to be monitored and controlled:

- atmospheric conditions
- movement (access control, theft and predators)
- agricultural conditions (ground humidity)
- animal movement
- water levels of reservoirs
- electrical equipment (i.e. water pumps)
- PV (photo-voltaic) installations (battery voltages)
- cooling facilities (temperature and operations status)
- farm workers (verify if site has been visited)

This comprehensive list indicates that a definite need could be addressed should a solution be developed to report information back to the user.

In addition to an understanding of the physical parameters of the farms, the survey also provides a perceived monetary value to physically monitor remote equipment. This in turn provides an initial estimated budget that should provide guidance as to what technology can be implemented and at what cost.

Current cost of farming

To gain a better understanding of the actual cost of the day-to-day activities on extensive farms included in the survey, the information in Table 2.1 was used to accurately estimate the cost of current farming practices. Thus, to be able to prove the feasibility of a remote monitoring system, some realistic costs needed to be assigned to these parameters. The results are as follows:

Travel cost

To travel to 13 POI's, 3 times a week with an average of 164 km per week, the total distance travelled per annum is around 8 518 km. When the standard AA travel rate is applied (AA Travel, 2012), the cost amounts to R25 486 per annum. The calculated travel cost is also shown in Appendix 6. However, this amount could be significantly higher for travelling on poorly maintained gravel roads commonly found in this area.

Labour cost

The actual cost for labour required for physically monitoring the points on these farms is calculated by using averages for wages and time spent per point of interest. The labour cost that was used for this calculation was obtained via the financial institutions as discussed in Section 2.3.2. This cost amounts to a total of around R38 000 per annum. Table 2.2 indicates the calculations that were made to be able to estimate this amount.

Table 2.2: Labour cost calculations

Cost of labour for monitoring POI's		
Monitoring Trips per week	3	
Average traveling speed	40	km/h
Weekly distance to travel	82	km
Total travel time per year	208	Hours
Time spent per POI	15	Minutes
Total time spent at POI's per year	507	hours
Total time spent per year	715	hours
Labour cost per hour	R53.125	R/hour
Annual cost	R37 894	R/year

2.1.3 Financial considerations

This section aims to corroborate the information that was obtained by the survey and also the estimations that were made. Financial statements for the local farms were evaluated as part of the research done by the author.

Why financial statements

Research indicated that there is very little statistical or research information available in the public domain for the cost of commercial farming for this specific region. This posed a problem for this research in that the cost of farming could not initially be validated or confirmed. It was then decided to contact local financial institutions to evaluate a number of financial statements of the local farmers. Since the statements that were used are audited financial statements, it provided an accurate method of obtaining data for this region. This then provided enough insight into the cost of farming and together with an understanding of the farming practices, as seen in the survey, provided sufficient information to conduct a realistic feasibility study. For the purposes of this project the expenses of the farms have been categorised and averaged as shown in Figure 2.3.

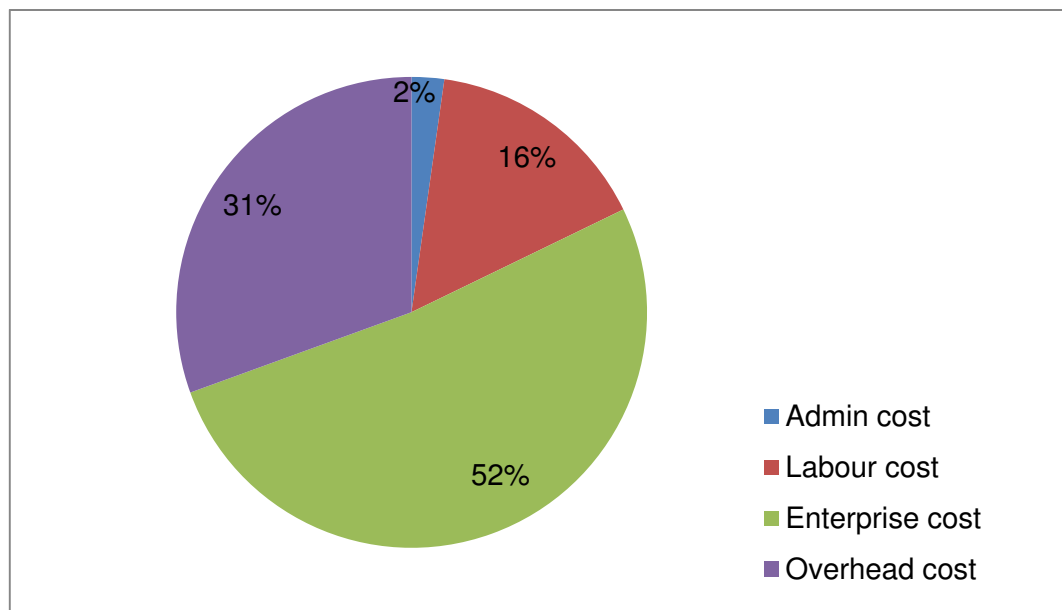


Figure 2.3: Breakdown of expenses

Administration: This includes basic expenses, for example, membership fees and stationary.

Labour: All expenses related to (and including) salaries and wages are categorised in this section.

Enterprise: The expenses that are directly related to the day-to-day farming activities, including fuel and running cost of vehicles, are included as part of the enterprise cost. This type of cost should always pertain to the methods that are implemented for the given farm.

Overhead: In this category, all the fix expenses required for farming are added. This will typically include insurance and taxes.

To assess the financial feasibility of the wireless telemetry system on farms with a cost breakdown as in Figure 2.3, it was important to determine which aspects of the overall cost could be reduced by implementing a wireless telemetry system. As far as administration cost and overhead cost are concerned, a remote monitoring system should not have a significant impact on these costs – these costs will be incurred whether or not the methods of farming are altered. However, a remote monitoring system will have a significant impact on labour cost and enterprise cost.

Table 2.3 provides a summary of farming cost for the farms that was included in the study. This should resemble the typical extensive farm in the Northern Cape province of South Africa.

Table 2.3: Local farm parameters and operating cost

Parameter	South Africa
Farm size	8 000 hectares
Overall cost of farming (total expenses)	R750 000
Estimated number of livestock units	1 100
Operating cost	R655/livestock unit
Operating cost	R90/hectare

The research showed that the labour cost and relevant enterprise cost, which include travel expenses, vehicle maintenance and depreciation, add to 34% of the total cost of farming. Therefore, from Figure 2.3 and Table 2.3 the total average enterprise cost that could be influenced by a wireless telemetry system is calculated at around R255 000 per year for the average farm relevant to this study. This does not imply that all the labour and enterprise cost can be eliminated, however, even a small percentage reduction in cost will result in a significant saving each year. This accumulated saving can be used to repay the implementation of a wireless telemetry solution.

2.2 International comparison

Sheep farms with extensive pastures in the Northern Cape province of South Africa were compared with the sheep farms with intensive pastures found in the South West of Victoria in Australia to verify the research results obtained thus far.

Figure 2.4 illustrates that the relational cost breakdown for the two industries is very similar.

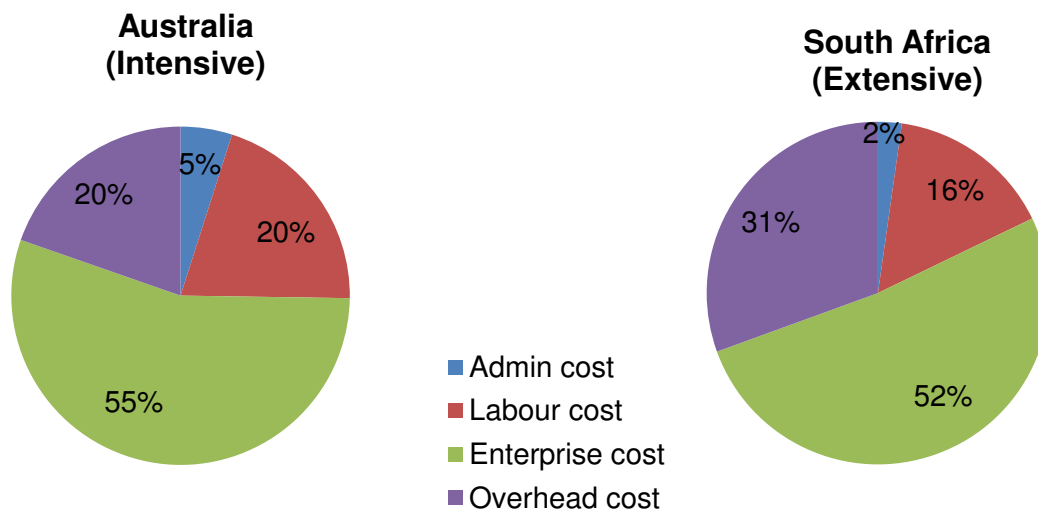


Figure 2.4: Overall farming costs (Department of Primary Industries, 2009)

Table 2.4 shows the comparison between the South African and Australian farming industries and indicates significant differences between these industries. It can be seen from Table 2.4 that the number of livestock units per farm is six times less for extensive farming in South Africa than those for intensive farming in Australia. This should result in a much lower turnover and also financial yield for extensive farms in South Africa. Also, the average extensive farm size in South Africa is eight times more than that of the intensive farms in Australia which results in a very low livestock density and also double the operation cost per livestock unit for the South African Farms. Since the cost per livestock unit is almost double and the farms are much larger in South Africa, the need for improving farming efficiency to optimise financial yield is especially emphasised for the South African farmers who need to implement extensive farming methods over large areas in a semi-desert climate.

**Table 2.4: Intensive and extensive farming
(Department of Primary Industries, 2009)**

Parameter	South Africa	Australia
Farm size	8 000 hectares	1 019 hectares
Livestock units	1 100	6 891
Livestock density	0.14/hectare	6.76/hectare
Operating cost	R90/hectare	R2 500/hectare
Operating cost	R655/Livestock unit	R370/Livestock unit

Furthermore, using labour cost and time to expand the current enterprise rather than just maintaining it, could benefit the farmer by providing enterprise growth with financial gain.

2.3 Comparing the results and estimating a budget

Included in the product requirements is a budget estimation. To be able to provide a budget estimate for project development purposes, the information gathered during the market research was considered and summarised in the following section.

2.3.1 Survey estimates

The calculations made from the information gathered as part of the survey shows that up to R63 380 per year could be spent on physically monitoring all the points on and around the farm. The R63 380 consists of R37 894 per year for labour and R25 486 per year for enterprise cost as described in Section 2.1.2. Keeping in mind that this is a fairly conservative estimation, this cost portrays a theoretical amount but should prove to be relatively accurate.

2.3.2 Results obtained from financial institutions

In an attempt to verify the results in Section 2.3.1, audited financial statements of the farms were used. As part of this process, all relevant labour and enterprise costs that could be reduced by implementing a wireless monitoring and controlling system were added. The total costs amounted to R255 000 per farm per year.

Even though physically monitoring remote site or equipment takes up a significant amount of resources, these costs should only be a portion of the total labour and expenses.

From section 2.3.1 it can be seen that if R63 380 could be saved out of a total of R255 000 a 25% savings on labour and enterprise cost could be achieved. However, if a more realistic assumption would be more than 25%, an actual saving of much more could be achieved.

2.3.3 Initial budget estimation

If a 3-year payback period is acceptable for the farmer, a capital cost of up to R76 500 for a remote monitoring system could be justified when a conservative estimate of a 10% saving is assumed rather than 25% as mentioned in Section 2.3.2. A 10% savings would be a conservative estimate and should easily be justified.

2.4 Currently available products and technologies

After considering the user requirements it was necessary to investigate the technologies that are currently on the market that are similar to the new technology requirements that were developed. This forms part of the QFD process mentioned in Section 1.5 and also forms the next phase in the QFD design approach. Two companies were approached and interviewed that have a thorough overview and understanding of a typical benchmark product applicable to the user requirements.

2.4.1 Company A (Spectrum Communications)

This company specialises in long-range telemetry mainly used in military or government applications. The system operates in the 400 – 500 MHz range, using different transmit and receive frequencies, enabling full duplex communications. Because of the system architecture, the range of this system is virtually unlimited due to high power repeaters.

There are disadvantages to this communication system, but a few key points will be crucial to consider should the system be introduced to a different market. A first drawback is that the transmitted or output power ranges up to 10 Wp. This means that it is not a licence-free system because of ICASA (independent communications authority of South Africa) regulations that specifies a maximum transmitted power of up to 10dBm (South Africa. Department of Communications, 2007:4-13).

This system's interface is also software-based and requires a licence. Thus, including the licence fee from ICASA, the running costs of this system become a major drawback. The cost of solar panels and batteries will also be substantially higher per node. Cost thus makes it unpopular for the larger sector of industry and for private use (Botma, 2008).

2.4.2 Company B (Tramirloc)

This company specialises in products that monitor livestock on larger farms. There are many overlaps between the application of this company's system and the scope of this project. This product's key feature is that it works in the 866 MHz ISM band, with a 10 mW transmitter that provides a range of up to 5 km. The network architecture of this product enables a mesh type network and licence-free communication. The cost of this system is relatively low as power consumption is reduced and off-the-shelf components are used for the development of both the transmitter and the receiver.

Numerous drawbacks were found while investigating this product. Bi-directional communication is not possible when using this system. For this added functionality, hardware and firmware need to be redesigned. The product was developed by a company employee; the protocol and hardware are therefore neither standard nor off-the-shelf. Another disadvantage of this self-developed system is its expansion potential. Improvement is limited while after-sales service becomes a risk to the user. Although mentioned already that the system is inexpensive, it should be taken into consideration that the system took more than five years to develop. From a design-lifecycle perspective, the added engineering hours and resources required to develop the product make it not that inexpensive (Bestbier, 2008).

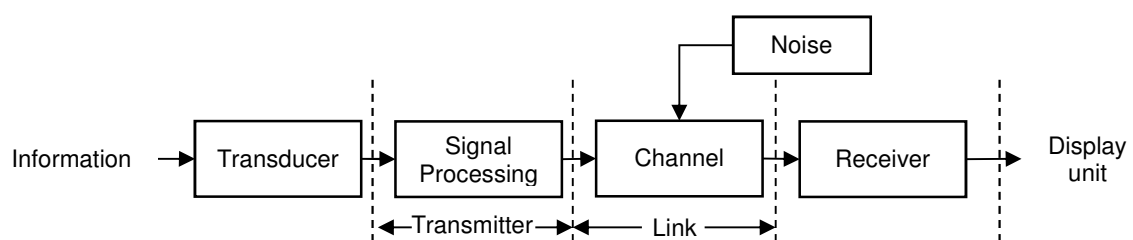
2.5 Literature review for technology to be implemented

To gain a better understanding of the technology available and the theory involved in the development of remote monitoring and controlling solutions, a study was conducted to ensure the development phase would be current and theoretically sound.

The literature research was divided into the three main components of this project, that is:

- instrumentation
- wireless communication
- user interface

These components are discussed in Sections 2.6 through to 2.10. Figure 2.5 shows a typical wireless communication network or telemetry system and how each of these components fits into the system. As part of the wireless telemetry system, application specific instrumentation is required since the signal acquisition circuit and transducer combination acquires information (or to measure) from the physical measurement parameters on a remote site. This is as indicated in the first two blocks of Figure 2.7. This information is then sent through the communication system to the user interface where the transmitter transmits the information through the channel/link to be received by the receiver. The information is then displayed for the user to interpret.



**Figure 2.5: Block diagram of a basic communications (telemetry) system
(Patranabis, 1999, p. 3)**

For the purposes of this research project, the block diagram shown in Figure 2.5 is changed in Figure 2.6 to more accurately indicate the various aspects that will be addressed in this section.

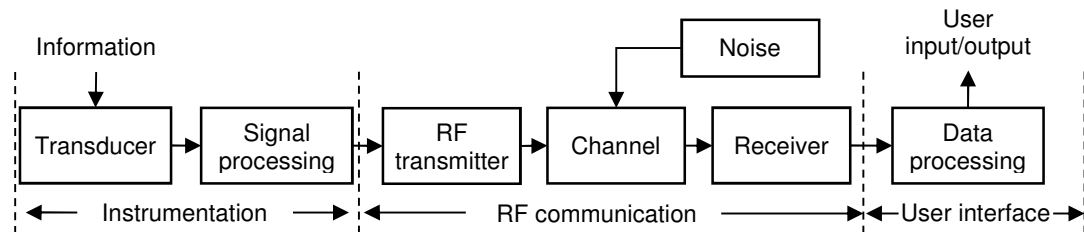


Figure 2.6: Updated block diagram for a wireless telemetry system.

2.6 Instrumentation

Measurements form an important part of the following processes (Ghosh, 2005, p. 1):

- monitoring of processes and operations
- controlling of processes and operations
- analysis

Figure 2.7 indicates that an instrument is a combination of mechanisms that, when combined correctly, provides an output signal (digital or analogue) that represents the physical parameter or quantity that was measured. The instrumentation discipline is often also referred to as metrology or measurements. Furthermore, instrumentation is described as the measurement of non-electrical quantities by electrical methods. Figure 2.7 shows a representation of an instrument in a block diagram. From this diagram it can be seen that the design and implementation of instrumentation incorporate multidisciplinary activities in order to meet the demands of the application (Dreyer, 2001, p. 1).

For example, transmission and display will require input from the telecommunications and computer engineering discipline respectively. For this project this is seen in the application of telemetry with a communication link and a user interface. Some other disciplines of electronics that are important to understand when instrumentation is considered are:

- DSP (digital signal processing)
- embedded programming
- RF (radio frequency) engineering
- electronics

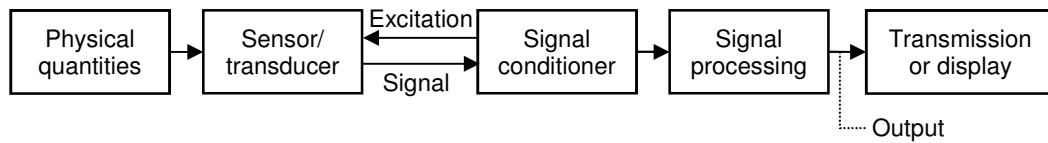


Figure 2.7: Representation of an instrument (Dreyer, 2001, p. 1)

2.6.1 Instrumentation for general purposes

When instrumentation is considered, there are many different categories of measurements (Dreyer, 2001, p. ix):

- electromagnetic variables
- electrical variables
- mechanical variables
- time, frequency and phase
- instrumentation for chemistry and physics
- interferometers and spectrometers
- data acquisition and recording

However, for the purposes of this project, only basic measurements of the most common electrical parameters will be required, that is (Dreyer, 2001, p. ix):

- EMF's (electromotive force) (Volt)
- electrical current (Ampere)
- resistance (Ohm)

The measurement of these electrical parameters could be employed to obtain most physical quantities, activities or events through sensors or transducers and should be considered when standardising nodes. The measurement of these parameters usually requires a microcontroller, a programmable logic controller, or a similar device for signal acquisition and processing.

2.6.2 Sensors and transducers

A transducer (see Figure 2.7) can be defined as a combination of elements that responds to a physical condition or chemical state, producing an output signal relative to the magnitude of the stimulus. (Murty, 2004, p. 8).

A simpler definition will be to say that a transducer is an energy or signal conversion device and forms the most important part of any measurement system (Ghosh, 2005, p. 2). A sensor, on the other hand, is in many cases referred to as a device that includes all the components of the measurement system as shown in a single platform or integrated circuit (IC). A diverse range of sensors and transducers are available. In the case of a wireless telemetry system the selection of the appropriate instrumentation is important for the accurate representation of that which is measured for this project. (Dreyer, 2001, p. ix).

2.6.3 Signal conditioner

One of the key components that are required for measuring electrical parameters as mentioned in this section is an analogue to digital converter (ADC). Figure 2.8 shows a block diagram of a digital instrument where, in the signal path, an ADC is used. A more simplified block diagram could omit the multiplexer, especially when only a single information signal is considered.

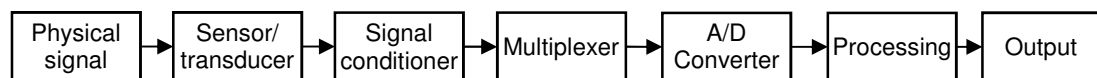


Figure 2.8: Analogue to digital converter (Dreyer, 2001, p. 11)

The implementation of an ADC will be required since the wireless communication platform that was implemented for this project is digital and in many cases a physical quantity will be represented as an analogue signal. Also, signal conditioning and other external circuitry were required for this project and will be discussed together with the development thereof at a later stage in this document.

2.7 Wireless communications

Figure 2.6 shows that wireless communication forms part of the main three sections of research for this project. In this section, relevant principles of RF (radio frequency) communications and related topics will be considered. Wireless communication is the main focus in this project and an important part of the telemetry system due to the physical challenges of the application.

Therefore, special emphasis is placed on the improving and optimisation of the communication system which should aid in the development and procurement stages of this project.

2.7.1 The history of wireless communications

The definition of communication is typically referred to as the conveying of information from one point to another through several electronic processes (Haykin, 1994, p. 1). Wireless communications is widespread over all industries for personal and industrial use and is an important part of our daily activities. Since the beginning of the wireless era, technology evolved and improved, making wireless communication the most popular method of conveying information over a distance. It eliminates the hassle of long and complicated wires and connections procedures (Cirronet, 2002).

The existence of electromagnetic waves was first established in 1887 by Heinrich Hertz and the first radio wave was received in 1902 by Guglielmo Marconi. This marked the dawn of the telecommunication industry. Analogue communication was primarily used until the first digital modulation techniques were employed in France around the year 1930. Digital communication was dormant up until the early 1970s when new technology renewed the interest in this type of communication (Haykin, 1988, p. 9). As the technology improved and industry grew, digital communications became more prominent and preferred. This made a significant impact on the cost of wireless communications. The evidence of this can be seen in the market and industry today, especially when cell phones and WiFi connections are considered.

2.7.2 Analogue vs. digital communications

When wireless communication is considered, the difference between analogue and digital communication techniques is important to understand.

The type of communication that is referred to, describes the information signal that is transmitted. In the case of analogue communications the information signal is typically an audio signal. Examples of this type of signal are AM (amplitude modulation) or FM (frequency modulation) broadcast radio.

Digital communications is the term that is used for describing the information signal that is a form of digital pulse train. This signal could also contain audio information but is represented by a data signal of 1's and 0's.

There are various advantages of digital over analogue communication; digital communication is therefore a more efficient and reliable method of transmitting information.

Some advantages include:

- increased immunity to channel noise and external interference
- flexible system operation
- common signalling format, regardless of the type of information signal
- improved security when encryption is utilised

There are however, two major disadvantages to using digital communication:

- increased channel bandwidth usage
- increased system complexity

Nevertheless, techniques do exist to overcome these problems (Haykin, 1994, pp. 17-19).

2.7.3 Basic limitations and resources of a communications system

The two limitations found in both analogue and digital communication system are noise and distortion. These parameters reduce performance of any communication system and may result in a loss of information, having therefore an impact on the range of communication.

The two resources in a communication system are average transmitted power and channel bandwidth. In general, the design objective is to efficiently utilise these two resources. Thus a communication system is typically a band-limited or a power-limited system (Haykin, 1994, p. 14).

When digital communication techniques are utilised, the bandwidth of a signal is directly proportional to the bit rate of a digital signal. Equation 1 shows that an increase in bit rate ($2/T$) will result in an increase in signal bandwidth (B). This is referred to as bandwidth efficiency (Haykin, 1988, p. 332).

$$B = 2/T \dots [1]$$

In general the design objective is to efficiently utilize the average transmitted power and the channel bandwidth.

2.7.4 The channel/propagation medium

In long range wireless communication, another important parameter to consider is the propagation medium, also known as the channel. When transmission occurs in a rural area unaffected by other electromagnetic transmissions, the range of the transmitted signal dramatically improves, unlike a busy city centre where there is increased RF interference. Buildings and other structures also influence the performance of a communication system in the city centre. Likewise, the same apply with mountains and trees in rural areas because of the phenomena called multipath fading and reflection (Cirronet, 2002, p. 4).

2.7.5 Transducers/ Antennas

Transducers or antennas are used to convert electrical energy into electromagnetic energy, which is then fed into the propagation medium. This is a key feature of a communication system. There are various types of antennas designed for specific applications (Coleman, 2004, p. 228). Improving a system's antenna design is typically a low-cost solution to increase range without increasing the system's complexity or transmitted power. Equation 2 shows the basic parameters of an antenna that could be altered to optimise a communication system (Coleman, 2004, p. 13).

$$P_t = \frac{Pr}{GrGt} \left(\frac{4\pi R}{\gamma} \right)^2 \dots [2]$$

Where P_t is the transmitted power, Pr is the received power, γ is the wavelength and G refers to antenna gain at the transmitter (Gt) and receiver (Gr) respectively.

2.7.6 Receiver sensitivity and selectivity

One method for improving the range of a system is to increase transmission power. However, this is the most ineffective and expensive way of improving system performance. Equation 2 indicates that the required transmitted power (P_t), will increase exponentially as the distance from the transmitter (R) increases.

Instead of trying to extend the range by increasing the transmitter power, it is possible to improve the sensitivity of the receiver only, which can then offer much better results. This is achieved by introducing a low noise amplifier (LNA) to the front end of the receiver with an improved noise figure (NF) as shown in Equation 3. The sensitivity of the receiver is normally defined in terms of the minimum detectable signal (MDS) (Coleman, 2004, p. 14).

$$\text{MDS} = [(F-1)T + T_A]kBS_0 \quad \dots [3]$$

Where MDS denotes the minimum detectable signal, F is the noise factor, B is the bandwidth, S_0 is the noise signal level, k is Boltzmann's constant and T_A is the absolute temperature in Kelvin.

Another parameter that can be varied to have an impact on the range of communication is the channel bandwidth. By reducing the receiver's bandwidth the range of a system is increased because the signal noise level is reduced. This also results in better adjacent channel rejection (Coleman, 2004, p. 13) and can be verified with Equation 3.

2.7.7 Modulation Schemes

When it is necessary to transmit digital data over a band-pass channel, the incoming data/information should be modulated onto a carrier signal with fixed frequency limits imposed by the channel (Haykin, 1988, p. 273). Modulation is defined as the process by which some characteristics of a carrier are varied in accordance with the modulating or information signal (Haykin, 1988, p. 273).

Modulation schemes have developed and improved over the years as digital wireless communications have become more popular. The modulation scheme used in a communication system will depend on the type of information signal being transmitted. The basic schemes that are used for analogue information signals are AM (amplitude modulation) and PM (phase modulation and is also referred to as FM (frequency modulation)). The most common of these are probably FM because it gets used every day for broadcast radio worldwide.

When it is required to transmit digital data the schemes mentioned above could be used. As the demand for a higher data rate increased, these schemes proved inadequate. New modulation schemes were developed to enable this information rate to increase. The following modulation schemes are widely used: PSK (phase shift keying), FSK (frequency shift keying), MSK (minimum shift keying) and variations of these as well as AM and PM. The modulating signal for these modulation schemes is a digital pulse train (Haykin, 1988).

When a more complicated modulation scheme is used, the system typically becomes more expensive but with added functionality. The majority of digital communications today are done through spread spectrum technology, where the transmitted signal is spread over a much wider bandwidth. Excellent noise and jamming immunity are some of the advantages of this technique and it is therefore used extensively in Wifi and Zigbee applications. This technique also enables communications to occur simultaneously in a narrower band and is referred to as code division multiplexing access (CDMA) (Ipatov, 2005, p. 1).

2.7.8 Frequency allocation charts (Industrial, Scientific, and Medical band)

When the electromagnetic spectrum is considered, communication begins at frequencies as low as 3 Hz and increases to 300 GHz. These are described as radio frequencies (RF) and can be seen in Figure 2.9. When wireless communication is implemented, radio frequencies from 300 kHz to 30 GHz are used. For non-specific, short-range devices (SRD) and telemetry, that will be used for this project, the typical frequency of operation is from 433 MHz up to 2.4 GHz (South Africa Department of Communications, 2007:4-13). This is the unlicensed instrumentation, scientific and medical band (ISM), which is a worldwide standard. This makes it easier to use off-the-shelf components for unlicensed communication that still adheres to limitations set by the governing authority (Texas Instruments, 2005, p. 2).

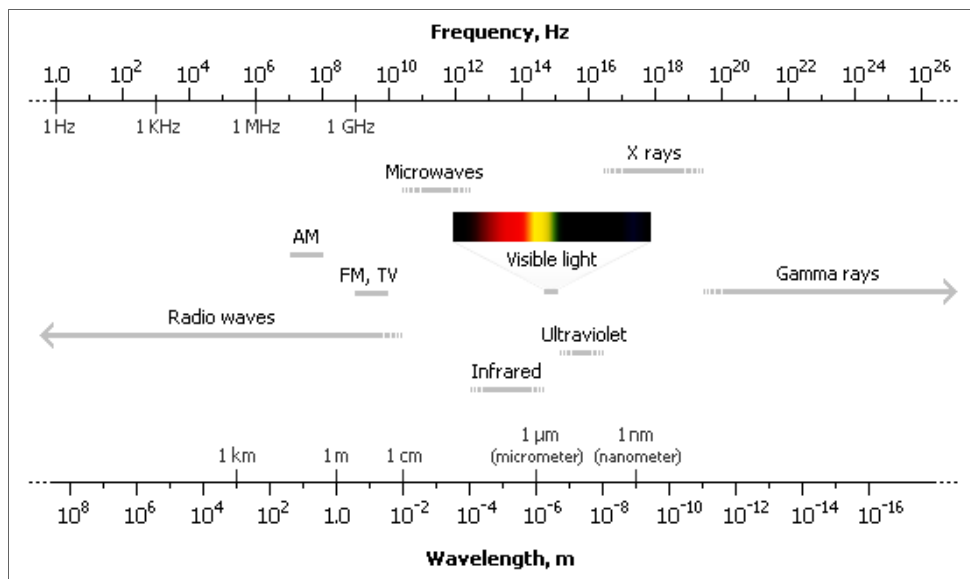


Figure 2.9: The electromagnetic spectrum (Microsoft Encarta online encyclopedia, 2008) .

2.7.9 Frequency of operation

As the frequency of operation of a system increases, the range is affected significantly and, equally important, the signal is subjected to line-of-sight transmission. As the frequency of operation increases, available channel bandwidth also increases, enabling higher data rates (Elpro Technologies, 2008, pp. 3-4).

2.7.10 Application of wireless communications

Examples of everyday use of wireless communication technology are radio, telephone, cell phone, satellite communication, internet, telemetry, etc. When wireless communications for telemetry was investigated, it was found to be a very broad industry with numerous applications spread over many industries. The application of wireless communication for telemetry seems to be similar to the application of this project and should aid in the development for the specific needs thereof.

Telemetry

Telemetry is used for supervising remote equipment, events, conditions and automated processes using private radio networks or other communication techniques.

These applications typically require information to be transmitted at various intervals, from seconds to hours. This information is then relayed back to a user interface from where the user is able to apply this information constructively. Furthermore, a substantial portion of the total cost of any telemetry system is made up out of the equipment used for the communication system and the installation thereof (Gnome Technology, 2008, p. 3).

Here are some of the fields where the application of wireless telemetry is typically found:

- utility metering
- transportation
- atmospheric or environmental
- financial
- security systems

Some of these fields, for example environmental applications, have correlation with the application of this project and the field experience should prove useful when a system is designed (Alta Pacific Technology Solutions Group, inc, 2010).

2.8 Wireless network options that were considered

For this project, a range of the different options for protocols and architecture were considered and are listed below:

- Wi-Fi
- generic digital communications
- simple analogue communications (transmit on error)
- Zigbee
- proprietary wireless sensor networks protocols
- satellite communications
- wireless HART (Highway Addressable Remote Transducer)
- SCADA (supervisory control and data acquisition)
- CB Radio (66-88 MHz)
- cell phone networks

These options are summarised in a table in Appendix 2 and should aid in the technology selection stages of this project.

2.8.1 Technology Comparison

When the physical challenges associated with extensive farming, mentioned in previous sections are considered, it is clear that a unique solution would be required to meet the needs of the application. The table in Appendix 2 summarises the wireless network options and provides an overview of various industrial and commercial solutions that were investigated. This summary is simplified in Table 2.5 in order to compare the common communication techniques available.

Appendix 2 and Table 2.5 show that the wireless sensor network solution (or Zigbee) would be best suited due to its added network functionality. Also, the “success metrics” field in Table 2.5 indicates that the low power and low cost of the Zigbee protocol set it apart from the other options. It was also noted that as far as most other parameters in Table 2.5 is concerned, the Zigbee protocol falls short to compete. However, this should not be a problem since Zigbee is targeted to fit into the low data rate market. This market is common to the sensor application and also in line with the requirements of this project.

Table 2.5: Wireless communication comparison (The Zigbee Alliance, n.d.)

Market name (Standard)	Zigbee (802.15.4)	--- (GSM/GPRS/CDMA)	WiFi (802.11b)	Bluetooth (802.15.1)
Application focus	Monitoring & control	Voice and Data	Web, email, media	Cable replacement
System resources	4kB-32kB	16MB+	1MB+	250kB+
Battery life (days)	100-1000+	1-7	0.5-5	1-7
Network size	Unlimited	1	32	7
Data rate (kB/s)	20-250	64-128+	11000+	1-10+
Transmission range (meter)	100+	1000+	1-100	1-10+
Success metrics	Reliability, Power, cost	Reach, quality	Speed, Flexibility	Cost, convenience

2.8.2 Cell phone networks

Since cell phone communication has become very popular in recent days, the option was investigated to use the current cell phone networks and compatible hardware that is very common and easy to implement. However, Figure 2.10 shows an extract of a magazine that indicates the cell phone coverage in South Africa. If this map is compared to the map in Figure 2.2, it is clear that the focus area of this project has very little cell phone coverage and therefore emphasizes the need for a solution other than what is commonly available.

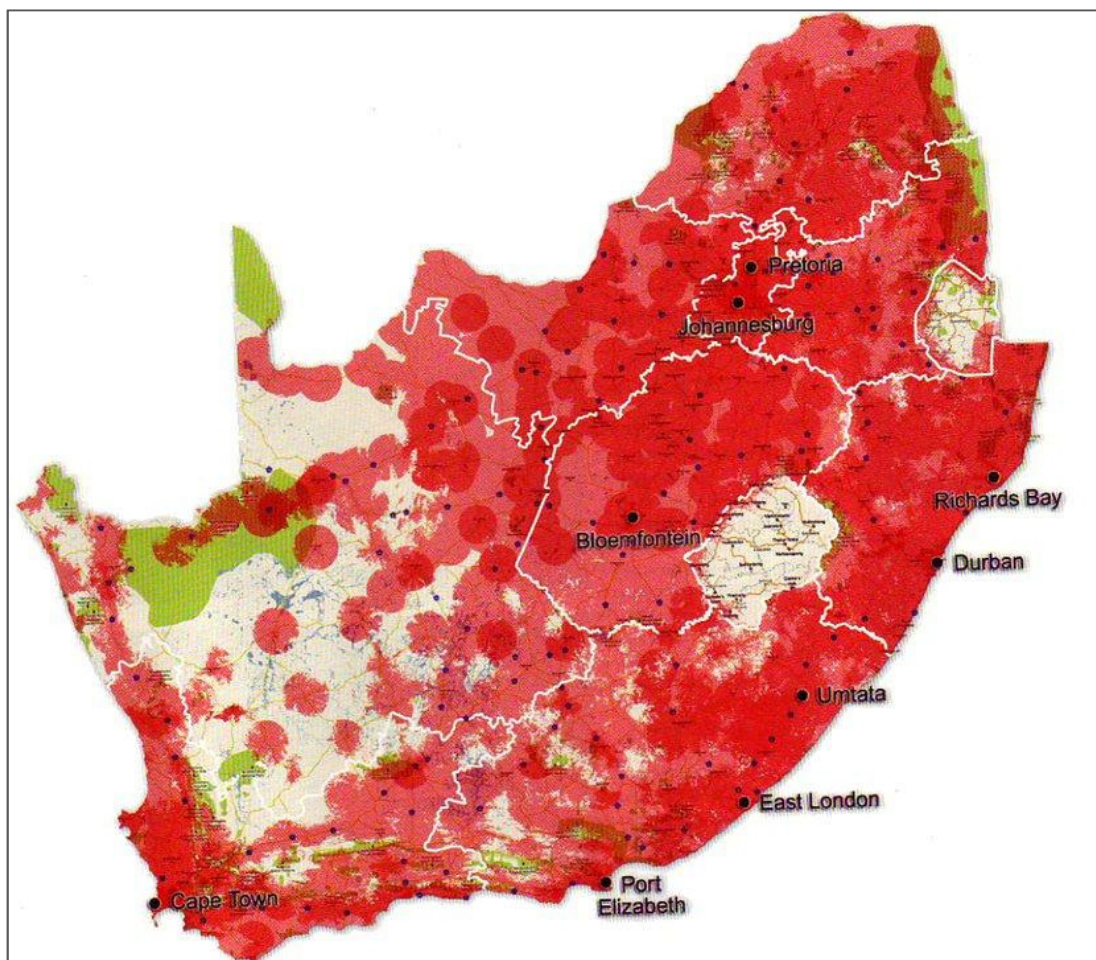


Figure 2.10: Cell phone coverage map for South Africa (CellC, 2012)

2.8.3 Zigbee

Zigbee is an international standard that defines a set of communication protocols for low data rate and short range wireless networking.

These networks typically operate in the 868 MHz, 915 MHz and 2.4 GHz bands, with low data rates of around 250 kbps. These frequency bands are also incorporated in the ISM band. Furthermore, the Zigbee protocol is optimised for low power consumption and is therefore typically battery powered.

Since the Zigbee protocol has adapted the Physical Layer (PHY) and the Medium Access Control Layer (MAC) from the IEEE 802.15.4 standard, this protocol is often referred to as the latter. The reduced implementation cost and simplified communication due to reduced data rate make the Zigbee protocol ideal for telemetry and also this research project (Farahani, 2008, p. 2).

Zigbee communication parameters

For reference, Table 2.6 gives an overview of the frequency of operation, modulation scheme and bit rate associated with the Zigbee standard for communication. These parameters are relevant in the later stages of this project especially for the technology chosen for the implementation of this project.

Table 2.6: Zigbee parameters (Farahani, 2008, p. 7)

Frequency (MHz)	Number of channels	Modulation	Bit-rate (kB/s)	Spreading method
868-868.6	1	BPSK	20	Binary DSSS
902-928	10	BPSK	40	Binary DSSS
868-868.6	1	ASK	250	20-bit PSSS
902-928	10	ASK	250	5-bit PSSS
868-868.6	1	O-QPSK	100	16-array orthogonal
902-928	10	O-QPSK	250	16-array orthogonal
2400-2483.5	16	O-QPSK	250	16-array orthogonal

Device roles and types

Two types of nodes are typically found in a Zigbee network. They are:

- reduced function devices (RFD)
- full functional devices (FFD)

These nodes are further defined to have a specific role within the Zigbee protocol and are noted as (Farahani, 2008, p. 9):

- coordinator
- router
- end device

A coordinator manages all the communication and is required by the Zigbee network to function properly. The rest of the network can consist out of any number of routers and end devices to form the Zigbee Personal Area (PAN) network.

Mesh networking

Since the Zigbee protocol enables mesh networking, the range can be extended by adding routers that can relay the information to other nodes that are out of range from the network coordinator. The coordinator in the Zigbee network is responsible for managing all the network parameters, data flow, and communication to the interface outside the network. An end device is typically a reduced function device (RFD) and will be in sleep mode almost all of the time.

Figure 2.11 provides a graphical representation of the different networking topologies supported by the Zigbee protocol.

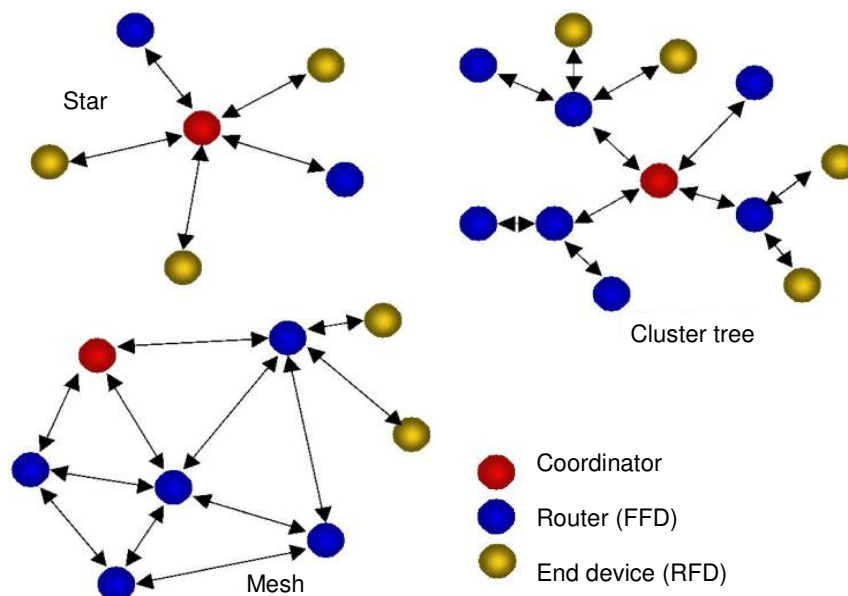


Figure 2.11: Zigbee topologies (Le, 2005)

2.9 OSI (Open Systems Interconnection) model

Although the ISO 7-layer protocol model is beyond the scope of this project, mention of this model is made since some of the layers in this model are referred to throughout this document. A brief description of each layer is also provided in Table 2.7.

Table 2.7: ISO 7-layer model (Technology Training Limited, 2012)

Layer (nr)	Function	Examples
Application (7)	User interface	Telnet, HTTP
Presentation (6)	Handles encryption & changes to syntax	ASCII/JPEG/M P3
Session (5)	Manages multiple applications and maintains synchronisation points	Operating systems
Transport (4)	Provides reliable or best-effort delivery and optional error and flow control	TCP, UDP
Network (3)	Provides logical end-to-end addressing used by routers and hosts	IP
Data link (2)	Creates frames from data bits, uses MAC addresses to access endpoint and provides error detection but no correction	802.3, 802.2, frame delay
Physical (1)	Specifies electrical parameters such as voltage, cable type, speed etc.	EIA/TIA, V.35

2.10 User interface

A user interface is a part of a system and enables people to interact with the system. The most common user interface is a screen, mouse and a keyboard used as part of a personal computer. In essence, a user interface consists out of two components, namely input and output. The input component is used by the user to communicate the need of the user in terms of the functions that need to be performed by the system. Examples of input devices are a mouse and a keyboard. The output is in most cases either a printer or a screen as found with a personal computer (Galitz, 2007, p. 4).

Research indicated that there is a significant amount of literature available on the topic of user interfaces.

The full extent of this topic is however beyond the scope of this project and therefore only a few practical options will be considered that will meet the needs of the user.

User interface as part of a wireless telemetry system

Figure 2.5 shows that the user interface is the last block in the functional block diagram and depicts the third main area of research. The user interface will serve as the input and output mechanism for the telemetry system. This means that the information that is gathered by the sensors will be conveniently available to the user from a central point. Also, sensors with controlling functionality could be controlled via the same user interface. In addition to ease of use, the two main challenges or practical limitations for the user interface with regards to this project are low cost and low power consumption. Further details regarding the user interface is discussed in the product planning and implementation stages of this document.

CHAPTER THREE

Project Specifications

3.1 Defining product requirements: reference design

The survey results section indicated that the design axioms can be derived from this type of survey feedback given by farm owners – these are, however, very extensive. One of the farms in the survey that conforms to the criteria for this project was therefore used to develop a reference design. This made it possible to use a real-life scenario on a farm to design a system that will most accurately provide a solution within the actual farm setting. This reference design is defined in the following sections.

3.1.1 Parameters of reference design

The farm that was chosen as the reference design is located in the Northern Cape near the Namaqualand area. This farm is a good representation of the target market for this project as per the survey results. The farm specifications are listed in the Table 3.1.

Table 3.1: Reference project specifications

Item	Description
Farm size	3 500 hectares
Number of POI's	19
Number of times inspected per week	4
Average distance to POI	2.3 km
Average distance between POI's	<1 km
Longest distance to POI	5 km
Longest distance between POI's	2.9 km

A Google Earth screenshot of the farm layout is shown in Figure 3.1. This image shows all the different sites where monitoring or controlling is required and also gives a visual representation of the distances between all these sites. Also, the different types of equipment or measurements are mentioned.

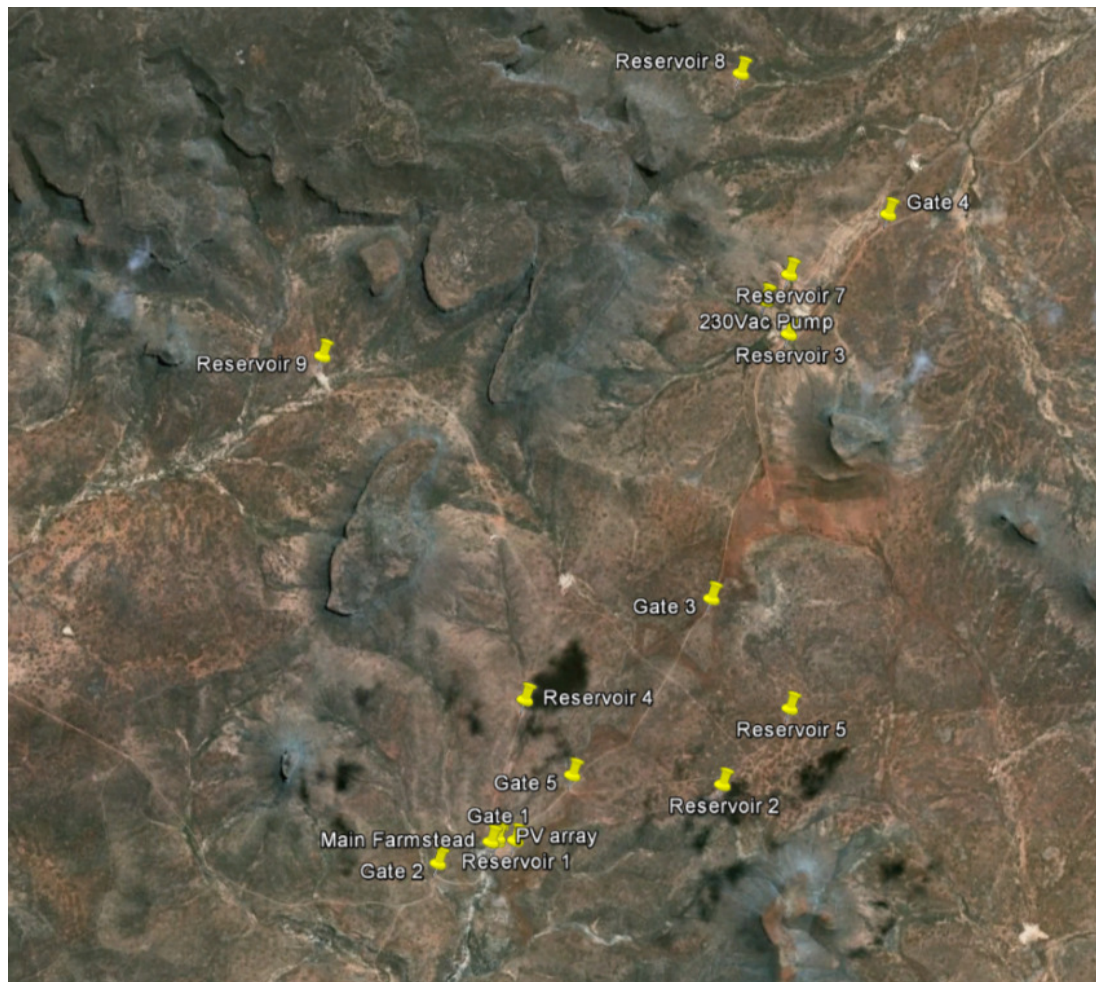


Figure 3.1: Points of interest for the reference design (Google, 2012)

3.1.2 Project requirements of the reference design

In this section the basic specifications will be discussed which influenced the type of solution that was selected for this reference design. This section will refer to the block diagram in Figure 2.6 as each component in the diagram is addressed.

Range and network architecture

Table 3.1 shows that the furthest POI from the base unit (farm house) is 5 km. The longest distance between nodes is calculated at only 2.9 km, which is more viable. Thus, from this it was decided that the communication system needs to be operating in a mesh-type configuration with transmitting and receiving capabilities of about 3 km. Bi-directional communications will also be required and should be achieved when a mesh network architecture is implemented.

Sustainability and ease of deployment

Considering the remote areas where the system will be implemented and the difficult accessibility and distance between POI's, some important observations were made. The devices, also known as nodes, installed at each POI needed to be self-powered. The power supply for these nodes should be designed in such a way that little to no maintenance is required.

Nodes in the network should be able to be configured automatically and the network will need to be "self-healing", thus be able to recover from power outages or network failures. This should result in an easy to install and off-the-shelf solution.

Low cost and maintenance

The results obtained via the survey indicated that the average amount the user will be willing to pay is around R9 000. Considering the number of nodes required (about 14, including base stations), the amount available per node is about R640. This was an important consideration for the system design and initial product specifications. The system cost estimations, payback period and budget will be discussed in Section 5.

Reliable and user-friendly

When a system is proposed, it is important to take into consideration the average technical understanding the user might have, which could be limited. It is therefore important to develop an intuitively pleasing system with maximum built-in user-friendliness and reliability. This will ultimately determine whether a system such as this could be successfully implemented.

System block diagram

In Figure 3.2 a basic system block diagram of the proposed system is shown. The sections that follow will discuss each component in more detail.

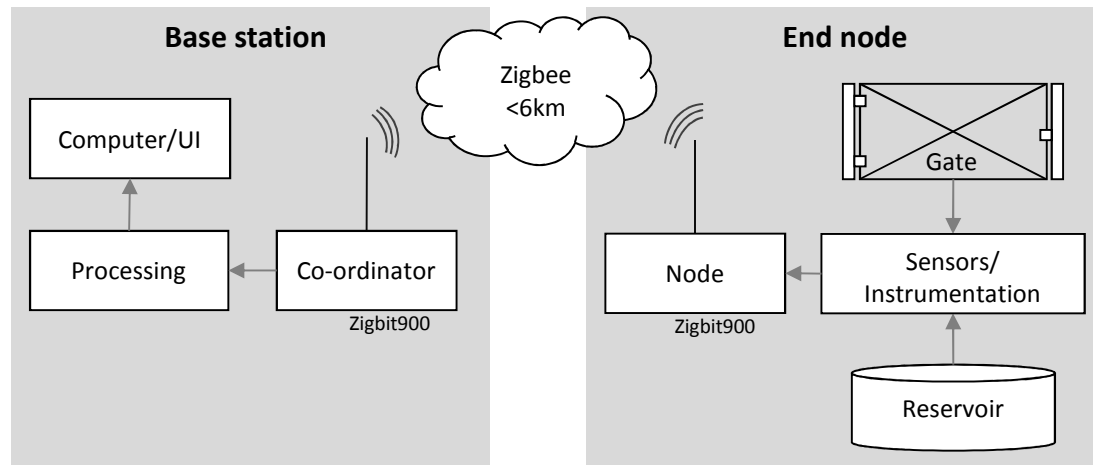


Figure 3.2: System block diagram

3.2 Project specifications: wireless communications

In this section the various options are investigated to establish the technology that was used to implement the wireless communication for the reference design.

3.2.1 Typical shortcomings of available wireless communication systems

In Section 2.4 two products have been discussed that are currently available on the market; various shortcomings or limitations have been found with these products. This does not necessarily imply that these products are not good; it simply means that they were not ideally suited for this project. Table 3.2 shows the requirements for the system derived from the research as well as the shortcomings of currently available products investigated for this project.

Table 3.2: Requirements vs. Shortcomings

Requirements	Shortcomings
Long range	Short range
Low cost	Expensive to implement and maintain
Low power consumption	High power consumption
No licence/running cost	ICASA licensing required
Effective/easy	Complex setup and operation
Plug and play	Difficult installation & setup
Bi-directional communication	Limited functionality

3.2.2 Zigbee range of communication

Considering all the different technologies available, a Zigbee-based protocol and hardware were chosen. This decision was based on the advantages this type of wireless sensor network has. Some advantages are the self-healing and configuring functionality of the protocol, low power consumption, functionality, and the readily available Zigbee-related products, such as development tools and software (Stewart, 2007). However, after choosing to use the Zigbee protocol as the wireless communication between the nodes and base station, research indicated that a comprehensive list of options is available on the market. Appendix 3 shows a comparison between the different Zigbee communication options that were investigated for this project. When Zigbee-related products are considered, the main limitation seemed to be the range when operating at a carrier frequency of 2.4 GHz and 10 dBm effective isotropic radiation power (EIRP). A range of more than +/-100 meters is not often achieved with the commonly found protocol-related hardware and will not be able to provide sufficient range for the required application. Considering that the average "large" farm has a radius of 8 000 m, this technology will be ineffective on such a farm and an unrealistic number of nodes would be required. Also, various Zigbee products are available that have an increased range achieved by increasing the EIRP. This, however, is not an ideal option for this project since an increased transmitted power requires licensing from ICASA. Also, financial feasibility might be affected due to the cost implication of the increased power consumption.

However, the research showed that certain parameters could be varied to introduce a change in the physical range for the link budget calculation of the system. Practically, two parameters could easily be varied with off-the-shelf components without adding additional cost or increasing the power consumption or complexity.

Firstly, when the frequency of operation (carrier signal frequency) is reduced, it is possible to increase the range (Cunningham, 2008, p. 7). Secondly, using a modulation technique that reduces the number of signalling points could improve the range by increasing the signal-to-noise ratio (Haykin, 1988).

Research was then conducted to find products with a lower frequency of operation and which also utilise BPSK (Binary Phase Shift Keying) modulations rather than O-QPSK (Offset Quadrature Phase Shift Keying). These will reduce the effective bit rate (Atmel, 2009), but considering that only basic telemetry data will be transmitted, this should not restrict the functionality of the system.

3.3 Zigbit 900

At the time this research was conducted, the only known product available that could meet all the requirements for this project was the Zigbit 900 that is based on an Atmel chipset from the company Meshnetics. These modules stand out with a range of 6 000 m, with a transmitted power of only 11 dBm and a frequency of operation at below 1 GHz. The Zigbit 900 module is not a single chip solution and therefore offers significant advantages especially when performance, development cost and even mass production are considered. With a balanced RF output, the Zigbit 900 eliminated the need for any RF development which reduces the bill of materials and time-to-market for a wide range of RF applications. Another advantage of the Zigbit 900 module is that includes both RF and MCU on a single module. This module could be mounted on a basic double layer PCB (printed circuit board) with minimum external components since all the required passive components are incorporated on the Zigbit module. Also, embedded Zigbee software enables standard based communication and could be implemented for various applications. (Atmel Corporation, 2009)

A development kit was then acquired from the suppliers for the demonstration, evaluation and development of the system for this project. Some of the important parameters that make this product ideal for the project are listed in Table 3.3.

Table 3.3: Zigbit 900 specifications (Atmel, 2009)

Parameter	Description
Range	6 000 meter
Transmitted power (EIRP)	10 dBm
Protocol	Zigbee PRO
Frequency	<1 GHz
Battery lifetime	<10 years

Further advantages include a standard serial AT command-set for easy set-up and prototyping, a small footprint, and readily available development kits with support from the manufacturer.

However, during the procurement process the company that manufactured and supported this product had financial difficulties and was acquired by Atmel (Perlegos, 2009). This transition made it very difficult to get support, or the necessary development hardware. Because of the lack of support from suppliers, additional nodes necessary to build a fully functional demonstration system could not be acquired. However, the functionality and range of these units have been tested to verify that the hardware was performing according to the relevant parameters as mentioned in the product's marketing material. It was therefore possible to continue using the Zigbit 900 development boards for the project's design and implementation.

3.4 Project specifications: instrumentation

The market survey results indicated that a wide variety of equipment and events would need to be monitored and controlled by the user (listed in Section 2.1.2). Most of the equipment and events can be monitored by using similar instrumentation to obtain the required information, which is then relayed back to the base station. Initially, a node was designed to accommodate all the possible requirements listed in the market survey. This was an attempt to standardise all the nodes and also reduce the number of node types.

However, this was found to be too expensive and impractical and would add unnecessary complexity. One node per monitoring point proved to be more cost effective and practical, especially considering installation by the user. Furthermore, a non-intrusive solution is also required to minimise installation cost, time and complexity.

3.4.1 Zigbit 900 interface

The interface between the instrumentation and the microcontroller can be implemented by these three methods that are supported by the Zigbit 900 development platform (Atmel, 2009):

- digital I/O (input/output)
- ADC (Analogue to Digital Converter)
- PWM (Pulse Width Modulation)

The digital I/O will primarily be used for the acquisition of the sensor data because, in general, most of the sensors provide a digital “on” or “off” output. The exception is when the analogue to digital converter of the micro-processor is used for the measurement of the DC voltage. For the controlling (switching on or off) of remote equipment, the same digital I/O interface could be used, but with a different software configuration.

Based on the reference design, the number of node types for this project has been reduced to those listed in Table 3.4. These node types are discussed in more detail in the following sections.

Table 3.4: Node types required for reference design

Point of interest	Qty	Control	Monitor
Reservoir	9	Yes	Yes
Gate	5	No	Yes
PV array (DC equipment)	2	Yes	Yes
Water Pump	2	Yes	Yes
Valve	1	Yes	Yes

3.4.2 End node functional block diagram

Figure 3.3 shows an overview of the Zigbit 900 module implemented as an end node with the required measurements for the reference design. The design and implementation of this diagram will be discussed in Section 4.

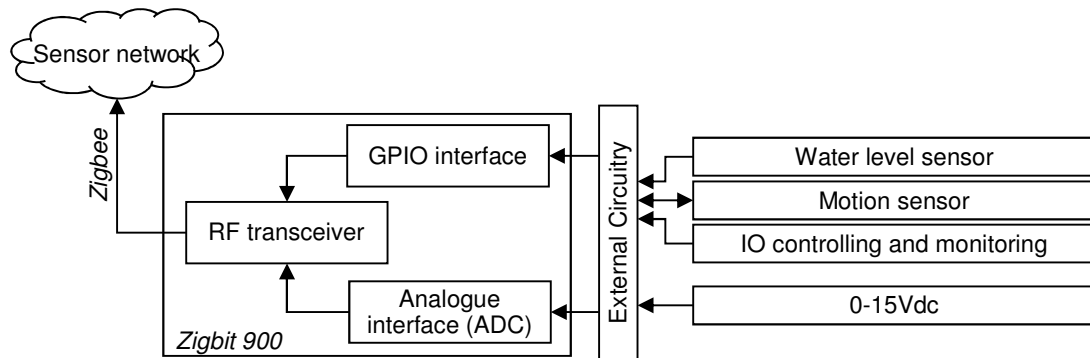


Figure 3.3: End node block diagram

3.5 Project specifications: user interface

Management of the sensor network and the user interface is done by the base station, which is located at the main farmstead. Numerous options are available for the user interface of the system. Before selecting the type of interface, it is important to understand and analyse the project requirements.

The main features or requirements for functionality of the user interface should include the following:

- monitoring
- controlling
- logging of events
- alarm function for critical events

Some options that could be used as a user interface were considered:

- personal computer (PC)
- television set
- LED display panel and buzzer combination

These and variations of these are relatively inexpensive methods, proving effective and simple to use when displaying the information from different sites or nodes to the user. Furthermore, because of the lack of electricity on these farms, having a low power, stand-alone unit to provide the logging and alarm functions is necessary, using battery power when needed.

In most cases the user should have a personal computer available. This means that the user will have general computer literacy, enough to be able to use a computer as a user interface for the wireless sensor network. An embedded web server utilising TCP/IP technology (Transmission Control Protocol/Internet Protocol) was therefore used to implement the user interface. This interface provided a low cost and low power solution with numerous advantages and expansion capabilities. The user will then have access to all the nodes via a generic web browser on a remote or local computer. The possibility of connecting the wireless sensor network to the outside world via the internet will also become important in the future; this, however, falls beyond the scope of this project. The proposed system layout for the base station is shown in Figure 4.3.

CHAPTER FOUR

System Design and Development

4.1 Zigbit 900 module and development platform

In section 3.2 it is explained why the Zigbit 900 Zigbee module and development platform was chosen for this project. When the OSI model for communication is considered, research showed that the Zigbit 900 development platform by Meshnetics provides all the necessary layers, from the Physical layer all the way to the Application layer. Thus, the development of the wireless communication system should be relatively easy and adhere to the initial requirement that stipulated that the solution for this product should ideally be an “off-the-shelf” solution. Table 4.1 summarises the specifications provided by the manufacturer of the components regarding the communication functionality.

Table 4.1: Zigbit 900 specifications (Atmel Corporation, 2009)

Parameter	Value
Range	<6 000 m line of sight
Frequency	868 MHz
Data rate	<1Mb/s
Battery life time	<10 years
Network topology	point-to-point, mesh, star, tree
Communication protocol	Zigbee PRO – BitCloud
Hardware platform	Atmel AVR
Max transmitter power	+12 dBm
Receiver sensitivity	-110 dBm

Figure 4.1 shows the Zigbit 900 RF/MCU module mounted on the Meshnetics development platform. This platform with supporting software and applications was used for development requirements of this project.

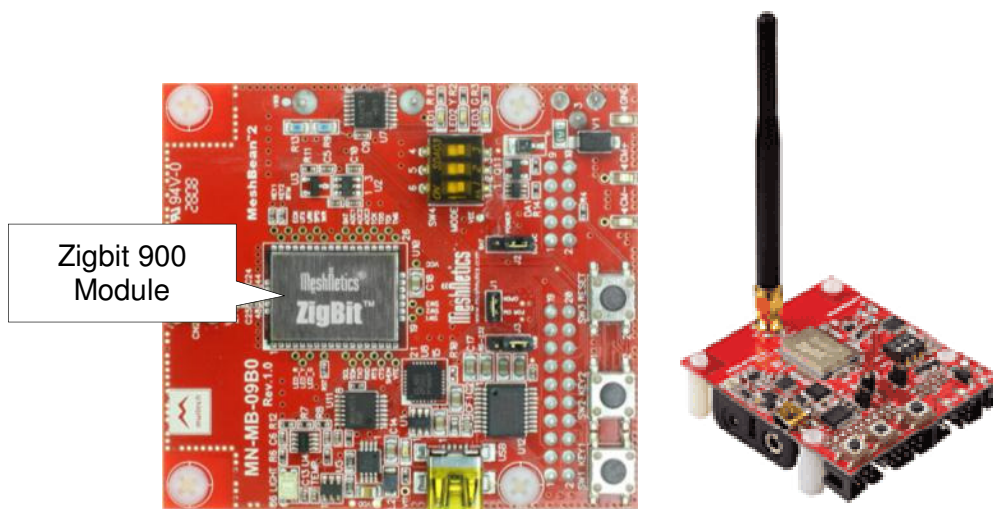


Figure 4.1: Zigbit 900 development platform

Furthermore, Figure 4.2 shows the functional block diagram of the Zigbit 900 module that is the heart of this development platform, indicating all the components and interfaces available on the Zigbit 900 module.

All these interfaces are either connected to on-board sensors or data communication devices installed on the development board. Also, a breakout connector is available, providing external access to all the required interfaces. For the development of this project, the following interfaces which are externally available, would mostly be used and is discussed in the following sections:

- UART (Universal Asynchronous Receiver Transmitter)
- GPIO (General purpose Input/Output)
- analogue

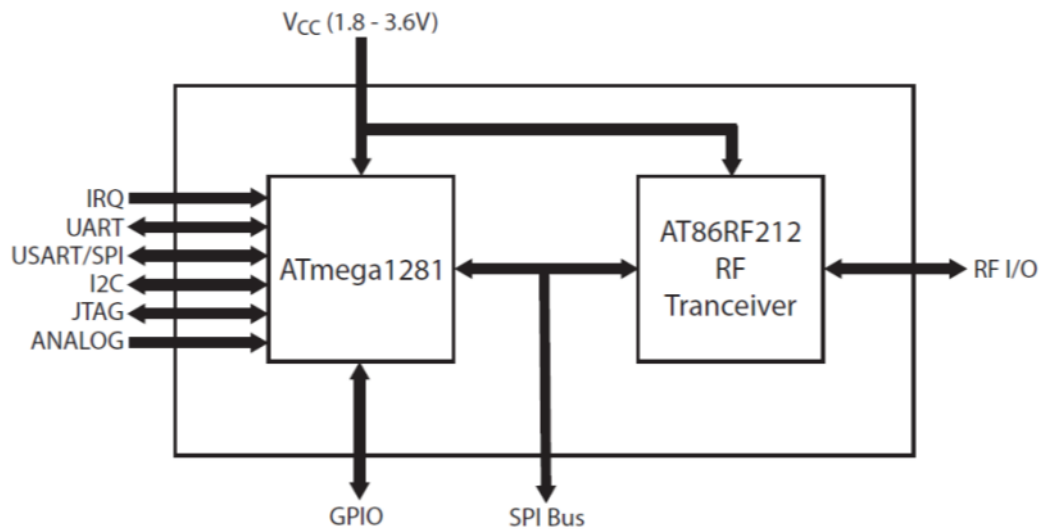


Figure 4.2: Zigbit 900 development platform block diagram (Atmel Corporation, 2009)

4.2 Wireless sensor network

The Zigbit 900 development platform was implemented for this project as an off-the-shelf solution. This development platform includes fully functional Zigbee PRO compliant applications and hardware as stipulated by the Zigbee Alliance. This section will discuss the implementation of these components for the purposes of this project.

4.2.1 Radio frequency development and range of communication (Physical Layer)

Chapter 2 indicated that the physical range or distance between nodes that can be achieved by the implemented technology will be one of the main constraints with regard to feasibility of this project. In terms of range, the specific advantages provided by the hardware of the Zigbit 900 module have also been discussed in Chapter 3. The development platform includes the physical layer of the communication protocol and hence no development was required in terms of the RF link for the purposes of this project. Preliminary hardware tests were conducted to verify that the maximum achievable range of the hardware was the same as what was claimed by the manufacturer in the datasheets and promotional literature. The results of these tests will be discussed in more detail in the results section.

4.2.2 BitCloud Zigbee PRO stack

The BitCloud protocol stack that is included in the development platform from Meshnetics provides full functional API's (application programming interface) that is compliant to the Zigbee PRO standard protocol (802.11.4). In addition, a serial communications platform and interface are provided with the development kit that enables the user to have access to the full set of Zigbee API's and hardware via serial AT-commands. This interface is implemented via a serial communication interface (RS232) which provided a sufficient method for the setup and interface with the wireless sensor network and all the nodes within the network. This was very beneficial since a fully featured Zigbee Pro development environment eliminated the need for intensive embedded software development and still provides full mesh networking capabilities with all the requirements for Zigbee PRO.

4.2.3 Battery life

The components included in the development kit are all optimised for low power consumption and long battery life. However, as with the range, the theoretical specification for this hardware seemed overly optimistic and needed to be verified.

The development kit included the embedded software required for the test. Upon further investigation of the tools and protocol stack, it was found that a battery life of 10 years can only be achieved by an end node and not the repeater or coordinator in the network because only the end node may be put into sleep mode.

The battery life of the router or coordinator, thus devices not in sleep mode, was tested to be no more than three days or 72 hours with 2 x AA, 1.5V alkaline batteries. The lifetime of the battery was found to depend mainly on the following:

- transmitted power
- processing overhead
- networking duty cycle

These three parameters should be considered to optimise battery lifetime. The lifetime of the end node battery, however, was not tested due to the expectant lifetime of up to 10 years; the only assumptions could be made from duty cycle and drawn current.

Also, when instrumentation is added to the end node, the battery lifetime once again was affected and this made modelling the lifetime of the battery more difficult. Also, since the coordinator and routers will have to be powered continuously for the sensor network to function properly, a constant power supply needs to be implemented for these nodes. Development of the power supply for these nodes is discussed in more detail in Section 4.5.3.

4.3 Base station

The base station is implemented at a central point and will typically be found at the main farmstead where the user will configure, monitor and control the components of the system. The base station functionality includes the following:

- Zigbee network coordinator
- data management
- logging or storing of data
- alarm functionality
- user interface

The block diagram in Figure 4.3 shows an overview of the base station and all the components required to provide the necessary functionality. Also, the communication techniques used for communication between these components are indicated. Each of the components will be discussed in the next sections.

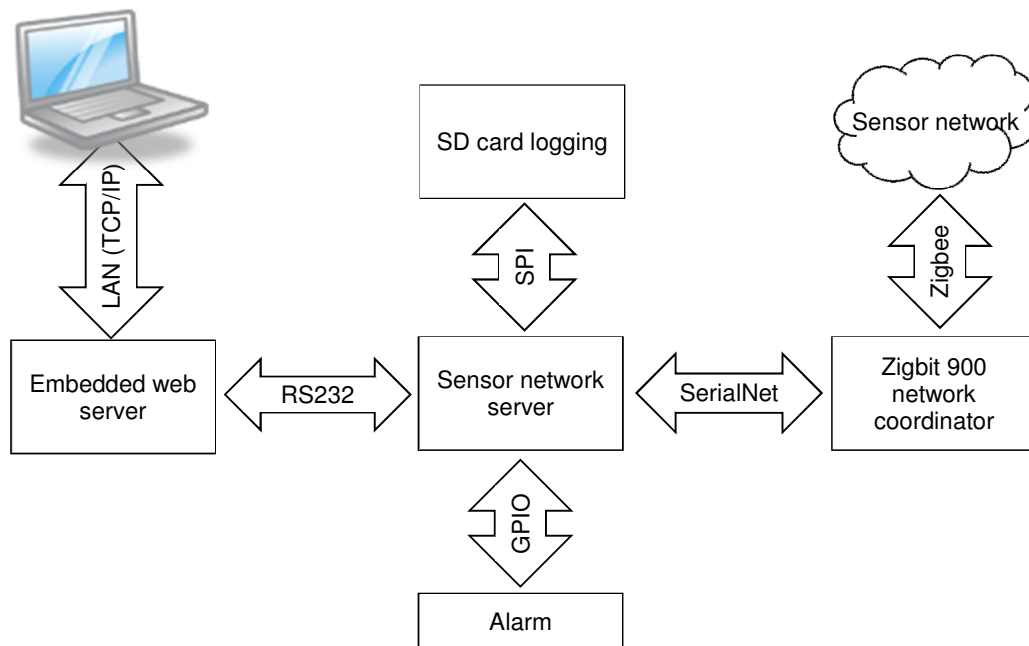


Figure 4.3: Functional block diagram of the base station

4.3.1 Wireless sensor network server

Section 2.8.3 stated that the coordinator of the sensor network is responsible for the management of the network and the communication between the nodes and the user interface. Since the system parameters require a stand-alone unit, a personal computer or laptop could not be used to manage and monitor the sensor network as there is not a constant power supply available. Therefore, an embedded platform was implemented using an additional development board with a MCU to communicate with the sensor network via the coordinator using the extensive serial AT-command set. This platform is the host device. The AT-command set is called SerialNet and is provided with the development platform from Meshnetics/Atmel. Figure 4.4 shows the SerialNet architecture.

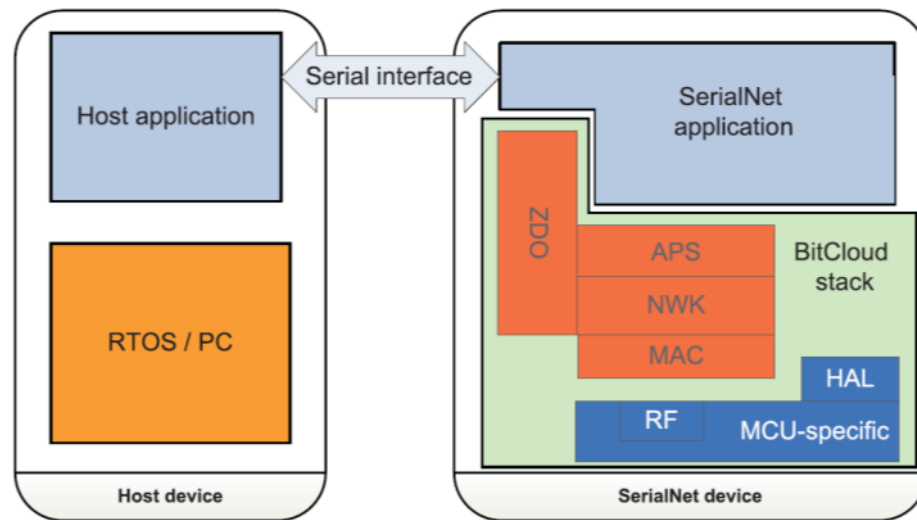


Figure 4.4: SerialNet architecture (Atmel Corporation, 2011)

The hardware setup that included the serial interface between the Microchip development board together with the Zigbit 900 development board provided server-like functionality. This functionality manages and stores data while it also controls the wireless sensor network communication, thus providing important functionality for the user interface and operating as a server. This sensor network server also processes all the raw packet data, then stores and manages this data for easy access by the user and other add-on applications. With this configuration, communication between the Zigbee network and the sensor network server is simplified.

Hardware configuration

The server is implemented using an Explorer 16 Microchip development board. This development board is shown in Figure 4.5.

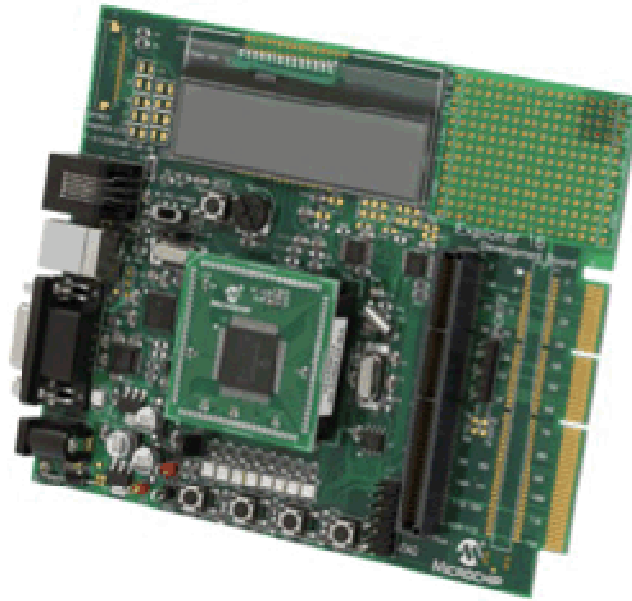


Figure 4.5: Explorer 16 demonstration board from Microchip

The microcontroller (24FJ126G010) that was used for implementing the server has two serial hardware based UART communication ports; one was configured to communicate with the on-board MCU on the Zigbit 900 module. The embedded programming for the sensor network server was done using MPLab IDE software and C-code. This was compiled using the C30 compiler, which is also provided free from Microchip. To summarise, dedicated hardware for the interface between the Zigbit 900 module (coordinator) and the server was implemented to ensure that optimum communication is achieved. The serial data connection is configured as follows:

Table 4.2: Serial interface parameters

Parameter	Value
Baud rate	38 400
Data bits	8
Parity	None
Stop bits	1
Flow control	None

SerialNet

SerialNet is a user application on top of Atmel's BitCloud Zigbee PRO stack which provided a quick and easy solution for easy access to the network and node information. SerialNet utilises a set of AT-commands that is implemented using RS232. When using SerialNet, an AT-command is sent to the Zigbee network coordinator (SerialNet device) from the host device. The requested information is then obtained by the coordinator and sent back as a reply to the host device's request. This process is illustrated in Figure 4.6.

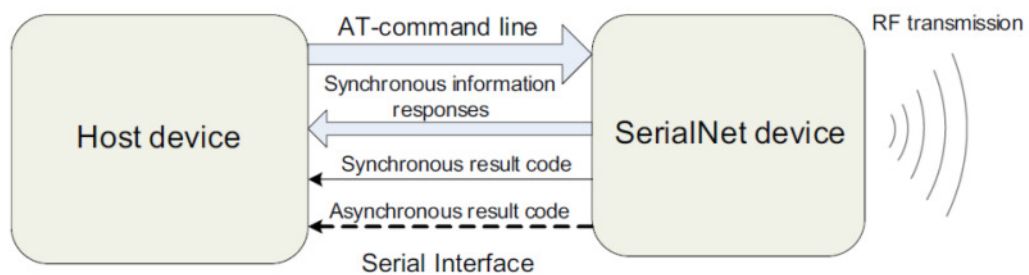


Figure 4.6: SerialNet command executions (Atmel Corporation, 2011)

An example of this would be to obtain the GPIO state or ADC value stored in the local hardware register, the S-register, of a specific node. The coordinator then relays the command to the specific node using the network address of the node. The node then replies with the values that were requested to the coordinator. These values are then sent back via the serial interface from where the request originated, namely the host device. This data can then be processed for the user interface. A screenshot of the AT-commands and the use thereof is shown in Figure 4.7. This screenshot show the process of configuring and setting one of the GPIO pins of the Zigbit 900 module using an application called ClearTerminal which is free of charge.

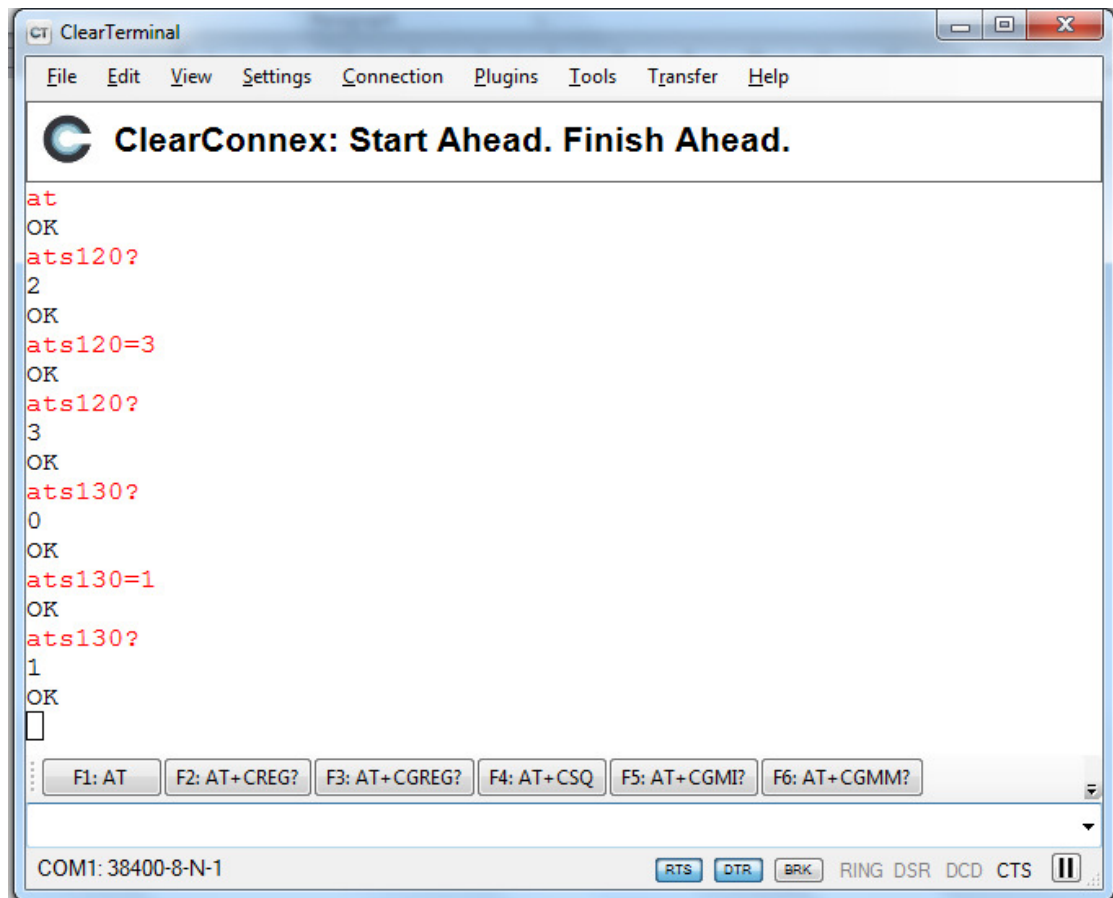
The image shows a screenshot of a 'ClearTerminal' application window. The window title is 'ClearTerminal' and it has a menu bar with 'File', 'Edit', 'View', 'Settings', 'Connection', 'Plugins', 'Tools', 'Transfer', and 'Help'. Below the menu bar is a header area with the ClearConnex logo and the text 'ClearConnex: Start Ahead. Finish Ahead.'. The main area of the window displays a series of AT-commands and their responses in a monospaced font. The commands and responses are: 'at' followed by 'OK'; 'ats120?' followed by '2' and 'OK'; 'ats120=3' followed by 'OK'; 'ats120?' followed by '3' and 'OK'; 'ats130?' followed by '0' and 'OK'; 'ats130=1' followed by 'OK'; and 'ats130?' followed by '1' and 'OK'. At the bottom of the window, there is a status bar showing 'COM1: 38400-8-N-1' and several control buttons: 'RTS', 'DTR', 'BRK', 'RING DSR', 'DCD', 'CTS', and a pause button. A row of function key shortcuts is also visible: 'F1: AT', 'F2: AT+CREG?', 'F3: AT+CGREG?', 'F4: AT+CSQ', 'F5: AT+CGMI?', and 'F6: AT+CGMM?'.

Figure 4.7: AT-commands demonstration

The process of sending and receiving data from the sensor network server to the Zigbee network is demonstrated in the code extract shown in Appendix 5.

The biggest disadvantage of using SerialNet is that hardware interrupts are not supported. Thus, the sensors that would need to provide alarm functionality needed to be polled at regular intervals. In most other cases, the user should be able to refresh the local data and get updated values from the nodes only when needed since the user is expected to only monitor the remote equipment or events at a daily interval. This was indicated by the market research. Three polling intervals were introduced into the embedded application to make sure that the network traffic is not too high while still providing sufficient functionality for the alarm function. The polling rate for demonstration purposes is 5 minutes, 1 hour and 12 hours.

These rates were configured by setting the event priority to high, medium, or low, respectively. This will ensure that not all the nodes were continuously polled, but only the ones that are seen as critical events. The polling problem furthermore explains the need for dedicated hardware, hence the sensor network server.

Figure 4.8 shows a basic flow diagram of the embedded software developed for the sensor network server to manage all these processes.

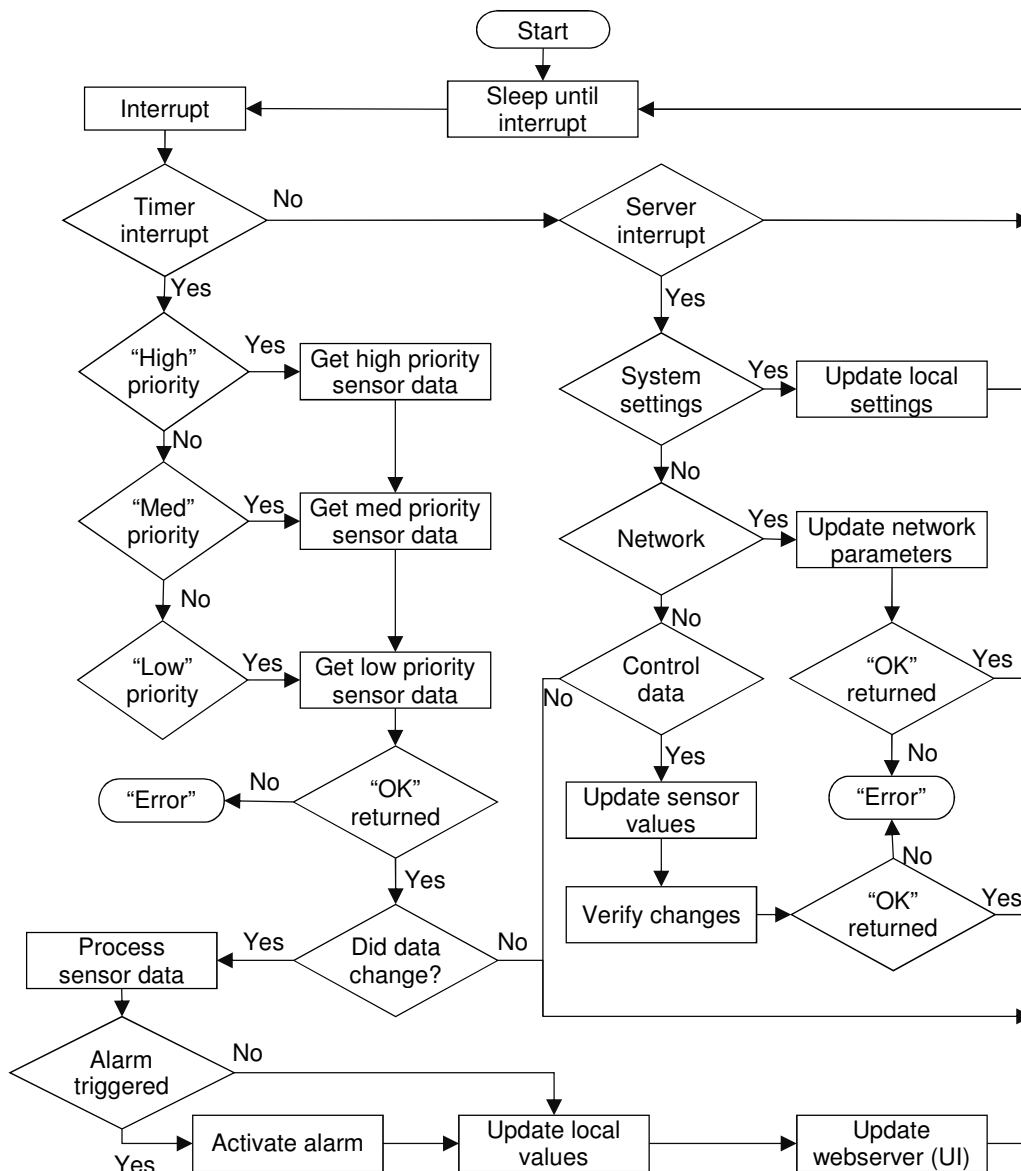


Figure 4.8: Sensor network server flow diagram

It is important to note that this system is not a real time event or data logger. If this was a requirement, a different approach would have been considered and again guided with the same process of selecting the technology used as was done in Section 3.

4.4 User interface

In section 3.5 the main features or requirements are mentioned for the functionality of the user interface and should include the following:

- monitoring
- controlling
- logging of events
- alarm function for critical events

These are discussed in more detail in the following sections.

4.4.1 Embedded webserver (controlling and monitoring)

Software implementation

By using a web-based (TCP/IP) user interface, the functionality could be very broad and provided sufficient control and monitoring functionality required for this project. However, this could become resource intensive and complicated. Also, to implement this web base platform requires a server or computer of some sort. Since the requirements with regard to functionality for this project are limited, implementing this interface on an embedded platform was possible. This was also ideal when power consumption is considered. Figure 4.3 shows the base station configuration and how the user interface was implemented as part of the base station. Since the author is familiar with the Microchip platform, the initial interface was developed in conjunction with Microchip development tools and integrated software libraries. This interface was purely text-based in order to reduce complexity and development time, and for demonstration purposes only. Furthermore, a website had to be developed that would serve as the interface between the user and the embedded Zigbee API's.

This website incorporated the following functionality within the web pages:

- monitoring of specific nodes
- controlling of specific nodes
- overview of events and latest activity of all nodes
- configuration of node functionality (name, type, priority, and alarm)
- configuration of node's networking parameters (address, type)
- system configuration (time, number of nodes, etc.)

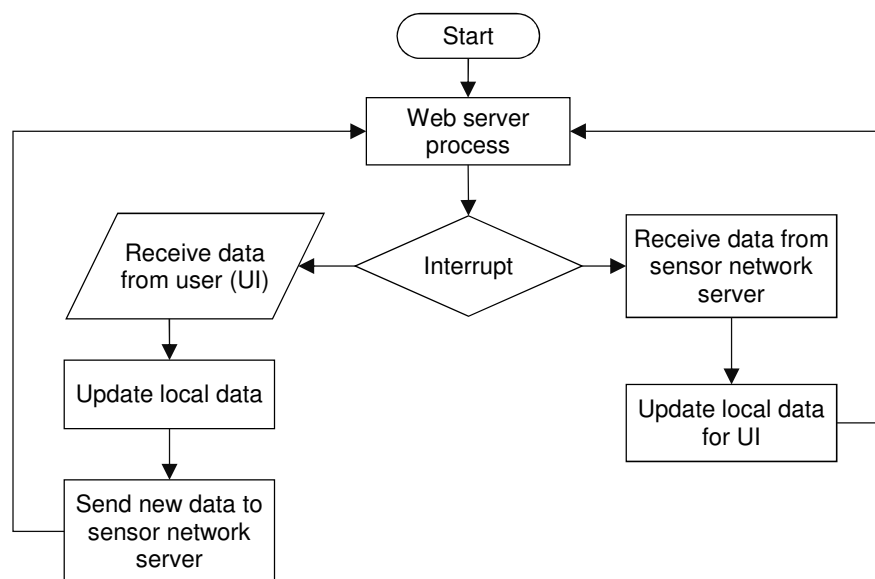


Figure 4.9: Web server flow diagram

Dynamic variables were used as the method to pass information from the webpage, which is the webserver application, to the embedded C-code on the webserver development board. This was done using the “get()” and “post()” command as standard functions of the internet protocol (TCP/IP). The information is then transmitted via the communication interface RS232 to the sensor network server to be processed and then either stored or sent to the appropriate node. The flow diagram in Figure 4.9 gives an overview of this process. A complete code extract of the web server is shown in Appendix 5.

Hardware configuration

Since the user interface forms part of the base station discussed in Section 4.3, the ideal was to implement this platform on the same hardware that is used for the sensor network server. Preliminary tests were conducted with this configuration but were unsuccessful. This could be due to the following:

- Transmission times through the sensor network varied significantly.
- A significant amount of resources were consumed by the web server.
- The internal interrupt structure became very complicated since both the webserver and sensor network communication was done through the internal UART's of the microcontroller.

These occurrences caused the webserver to hang continuously and communication to the network to fail. Implementing the web server on a second dedicated development board to simplify and speed up the development process was used as an alternative. The same Microchip platform that was used for the sensor network server was duplicated and used for the embedded web interface.

Thus, an additional development board was used to manage the TCP/IP data transmission for the user interface (webserver).

For the implementation of the webserver, another Explorer 16 development board from Microchip (Figure 4.5) was used together with an Ethernet add-on module, also from Microchip. This Ethernet module incorporates an ENC28J60 device that includes the MAC and PHY layers for communication. The microcontroller uses an internal UART to communicate via SPI with the external Ethernet module. This configuration is shown in Figure 4.10 and Figure 4.11.

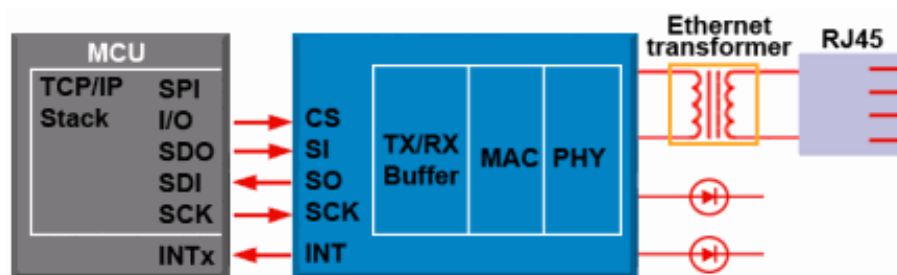


Figure 4.10: Ethernet functional block diagram

The Explorer 16 development board supports up to two serial interfaces and the Ethernet interface utilises one of these hardware serial interfaces. This means that the microcontroller still had one UART available for communication with the sensor network server as discussed in Section 4.3.1. Thus, all the data is retrieved from the sensor network server via the webserver and when commands are given by the user, the commands are sent to the sensor network server. The communication between the webserver and the sensor network server is also implemented via serial communication (RS232).

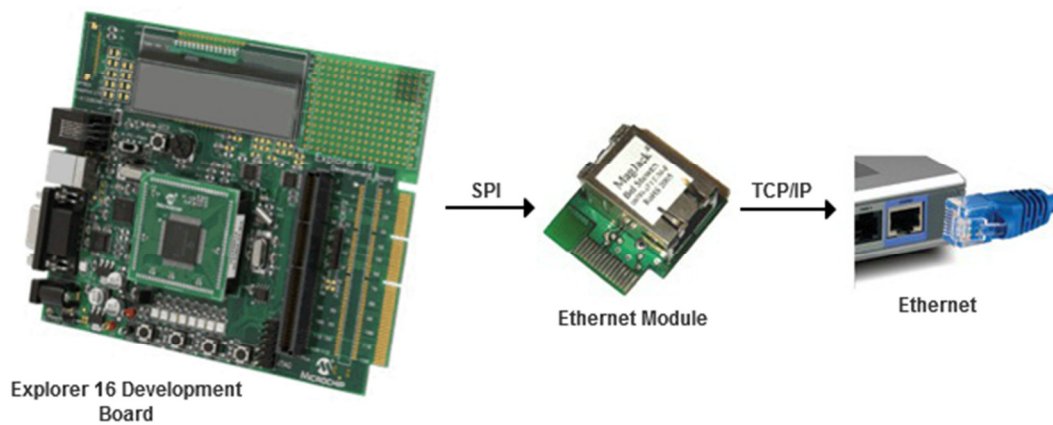


Figure 4.11: Ethernet configuration

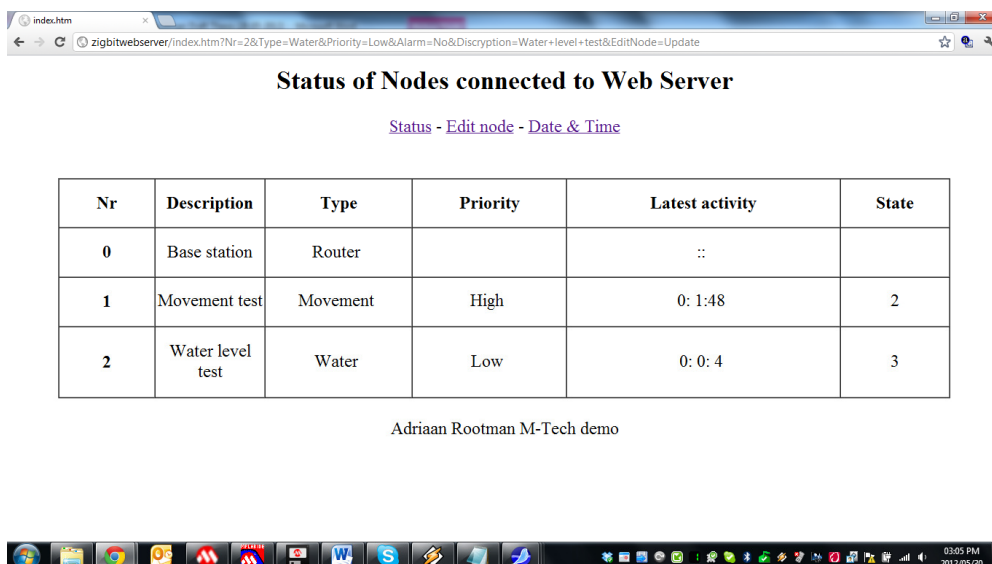
To summarise the development of the base station, Table 4.3 shows the software development breakdown as implemented for the webserver and the sensor network server.

Table 4.3: Software development overview

Application	Developed with	Technology	Interface
Sensor network server	C, C30, MPLab IDE	Microchip	UART/RS232
Embedded webserver	C, C30, MPLab IDE	Microchip	UART/RS232
Coordinator	SerialNet (AT-Commands)	Meshnetics	UART/RS232
Website	HTML, PHP, Java	SeaMonkey (HTML editor)	Dynamic Variable TCP/IP

The user interface

To summarise, the embedded webserver serves as the interface from where the user configures, monitor, and control the network and also the nodes. The user connects to this interface using standard LAN or Wifi connection to access the embedded webserver. It provides a highly compatible and versatile solution that is accessible even through devices such as smart phones or tablets, either locally over a LAN (local area network) connection or over the internet. Figure 4.12 to Figure 4.14 show the user interface which is accessed using the Google Chrome web browser that was implemented for the testing phases of the project. This text-based interface seems to be very basic and limited in functionality mainly due to the limited requirements of the application. For testing purposes, limited time was spent on developing a graphical user interface (GUI) with a professional appearance. The method of managing and presenting the data will ensure that a database or GUI application could easily be implemented on top of the lower level code already developed to handle all the communication and data processing. Furthermore, since only three nodes were available for developing the system, only two functional end nodes could be implemented. The number of nodes and table length were therefore hard coded to simplify the demonstration application. Figure 4.12 show the status of the two connected nodes.



Status of Nodes connected to Web Server

[Status](#) - [Edit node](#) - [Date & Time](#)

Nr	Description	Type	Priority	Latest activity	State
0	Base station	Router		::	
1	Movement test	Movement	High	0: 1:48	2
2	Water level test	Water	Low	0: 0: 4	3

Adriaan Rootman M-Tech demo

Figure 4.12: Status page of the user interface

Figure 4.13 shows the webpage from where the node is configured for operation by setting the node type, priority and alarm functionality. A description or a descriptive name could also be provided.

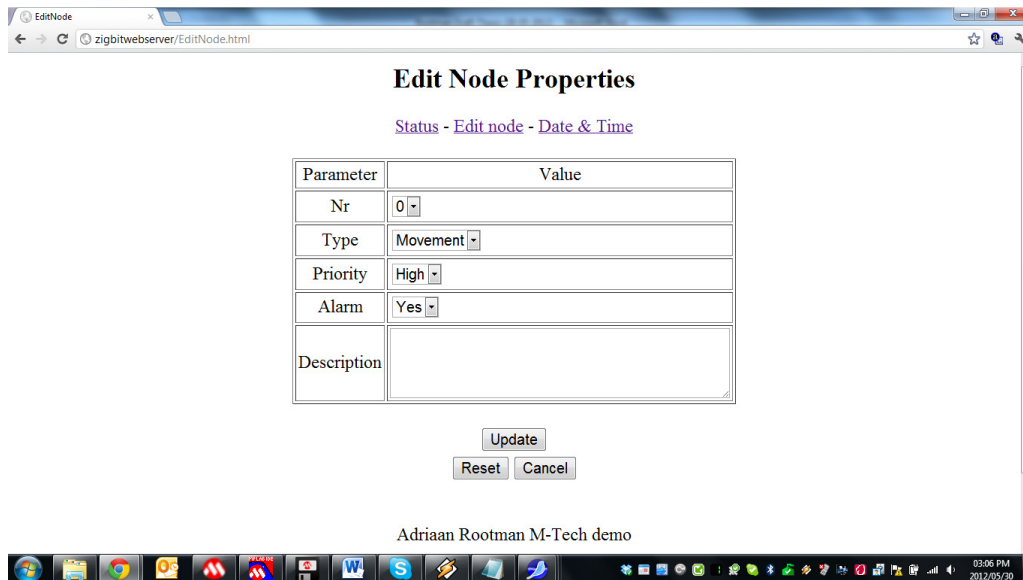


Figure 4.13: Node edit page of the user interface

Since the embedded application uses a real time clock for timekeeping purposes, it is possible to update the system time as shown in Figure 4.14.

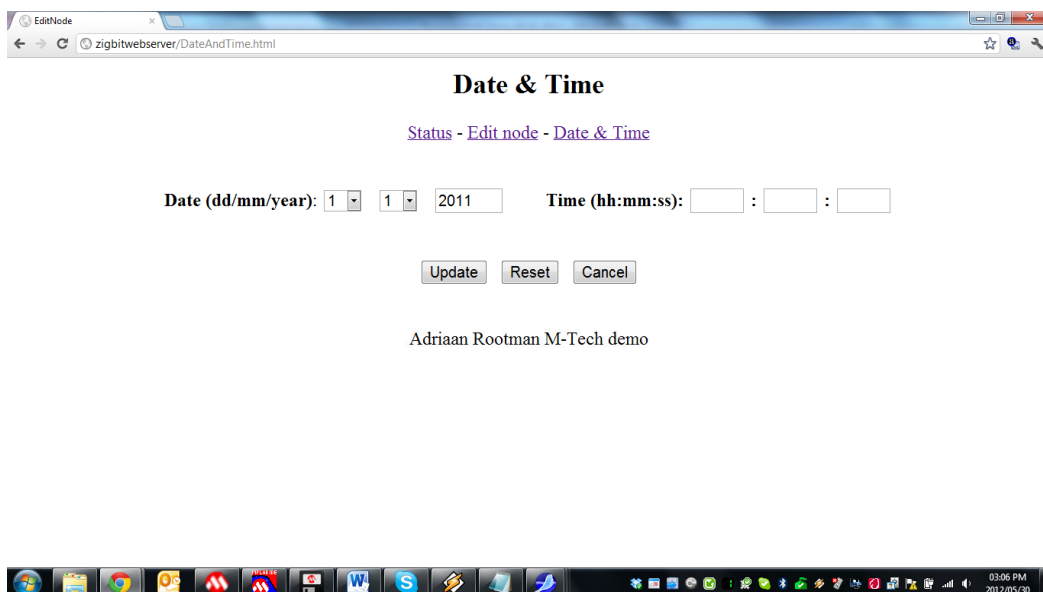


Figure 4.14: Date/Time edit page of the user interface

For the testing purposes the node information was stored in a structure and stored in the EEPROM (electrically erasable programmable read-only-memory) found on the Explorer 16 development board.

Additional time will still need to be spent in the development and testing of the user interface to ensure a fully functional and stable user level application. However, the user interface only constitutes for a part of the scope for this research project and development was stopped once all the required functionality was verified.

4.4.2 Alarm functionality

For events on remote sites around the farm that require immediate attention, alarm functionality was incorporated. This functionality could utilise a LED and buzzer combination on the sensor network server, implemented on a Microchip development board. The user will be alerted even when there is no power supply available or computer connected to the server. This alarm event will then indicate to the user that a high priority event has occurred that needs immediate action. Examples of high priority events will typically be movement or low water levels. This functionality will typically be set by the user for each node and can be enabled or disabled.

Section 4.3.1 showed that, to enable alarm functionality, the sensor network server will need to poll these sensors at regular intervals. This will result in a minimum resolution that is equal to the quickest poll rate of the sensor network server.

4.4.3 Data logging (SD card)

The system could enable easy logging of events by means of a SD (secure digital) card connected to the sensor network server. The server records all the events in a text file as character separated values (*.csv) that could be evaluated and processed in a software package such as Microsoft Excel. The amount of data that could be logged depends on the size of the SD card. Furthermore, the logged data is not volatile, making this method ideal for this purpose. An external module was used to implement a SD card for the logging functionality. This module is a Microchip development tool and can easily be integrated with the Explorer 16 development board. This configuration is shown in Figure 4.15.

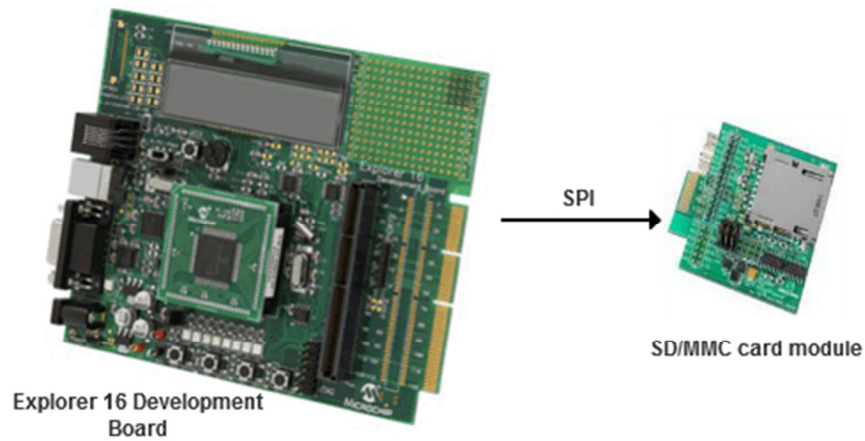


Figure 4.15: Logging configuration

However, for the demonstration system, this logging functionality could not be implemented as the Microchip development boards that were used for this project only supports two hardware UART's each. These serial interfaces are used by the following:

- TCP/IP adapter for the Microchip development board
- communication between the network coordinator and the Sensor network server (SerialNet)
- communication between the embedded web server and the sensor network server

Time constraints prevented the implementation of this communication interface on different hardware or in software.

4.4.4 Test setup for base station

A test setup for the base station was implemented and tested. This setup enabled the hardware and software development and testing as it was discussed in the previous section and is shown in Figure 4.16.

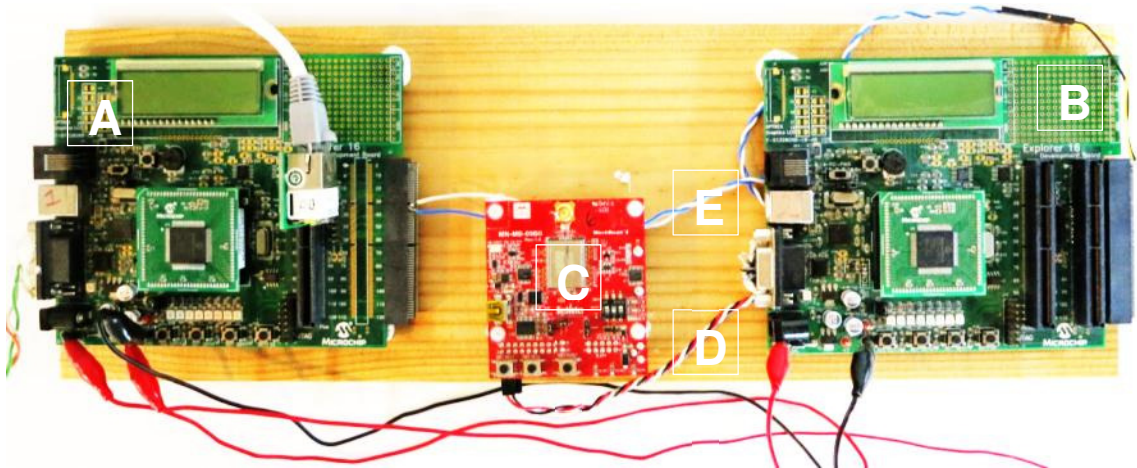


Figure 4.16: Base station test set-up

Where:

A is the embedded web server

B is the sensor network server

C is the Zigbee network coordinator

D is the RS232 interface used by SerialNet

E is the RS232 serial communication used for communication between the sensor network server and the embedded server

4.5 End nodes and repeater

Since the network is implemented as a mesh network, all nodes were designed to be used as data acquisition and controlling devices regardless if the node was configured as a router or an end device in the Zigbee network. In this section the development of these nodes will be discussed in terms of the implementation of the Zigbit 900 development nodes and the required instrumentation for the controlling and monitoring of remote equipment.

The design of a generic power supply is also discussed. A test setup for the end-nodes and repeaters are shown in Figure 4.17.



Figure 4.17: End node test setup

4.5.1 Node configuration and node types

Section 3 explained the technology that was chosen and the reasons why this technology would be the ideal solution for this project. Figure 4.2 shows the available interfaces from the microcontroller on the Zigbit 900 development board.

They are:

- Jtag
- SPI
- UART
- I2C
- Analogue to digital converter (ADC)
- IRQ (interrupt request)

However, for this project only the ADC and GPIO's were used since they are fully supported by SerialNet which was used for this project. This simplified the design and should not limit functionality.

Node types

As obtained from the market research, each node should have the ability to be configured to be able to function as either one of these:

- movement sensing
- water level indication
- monitoring of remote equipment (on/off)
- controlling of remote equipment (on/off)
- measurement of analogue signals

These configurations are typically done through the user interface. When this setup is done, very few parameters are actually changed on the sensor node itself, except for a few latches to configure bi-directional pins to be either inputs or outputs. The node configuration primarily changes the way the sensor network server requests and processes the data to and from the nodes. This means that the nodes could be used for any of the types of sensing required, depending on the external circuitry, physical connections, and software configuration of the sensor network server.

The external circuitry enables the different types of sensors that needed to be implemented to effectively and safely communicate with the generic sensor nodes. The design and implementation of each of these node types, including the external circuitry, are discussed in the following section.

Water levels of reservoirs

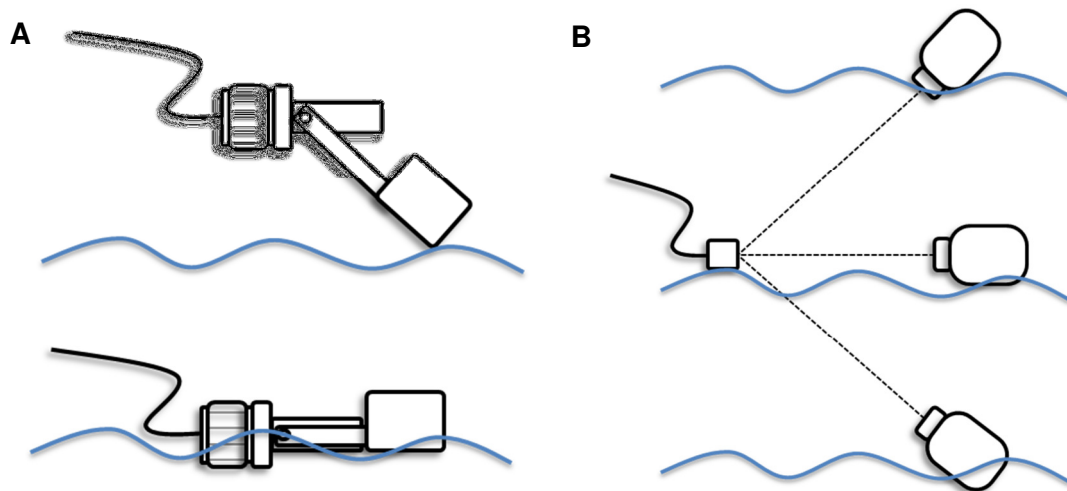
Table 4.4 shows the various options that were investigated for water level sensing of reservoirs around the farm. Furthermore, the options have been categorised by cost, complexity, functionality and power consumption. This is done to make sure that the most suitable type of technology is used according to the initial product specifications as derived from the survey.

Table 4.4: Liquid level sensing techniques

Item	Cost	Complexity	Functionality	Power
Ultrasonic	High	High	High	Yes
Optical	Med	Med	Med	Yes
Conductive probe	Med	Med	Med	Yes
Float switch (reed type)	Low	Low	Med	No

Table 4.4 shows that a float switch is ideal for the farming application. This switch provides sufficient functionality at low cost and does not need an external power supply that added to the cost of the node.

Sensor type: Two types of low cost float levels switches commonly found on the market were investigated for this project: a micro switch tethered type and a horizontal reed type. These are shown in Figure 4.18 where A is a reed type float level switch and B a micro switch tethered type.

**Figure 4.18: Float level switches**

The advantage of the switch shown in B of Figure 4.18 is that it has a built-in mechanical hysteresis that makes it ideal for controlling the filling or emptying process of a tank by switching a pump on or off.

For the purposes of this project this will not be sufficient as the resolution of the measurements will not be high enough, only providing a full or empty signal. Therefore, the horizontal reed type switch was used.

Sensor configuration: Three float switches indicating up to four possible water levels were installed. The number of switches can be increased if a higher resolution is required; for the purposes of this demonstration only three were used. Figure 4.19 shows the configuration of the sensors and how the four levels are obtained for each of the four different scenarios.

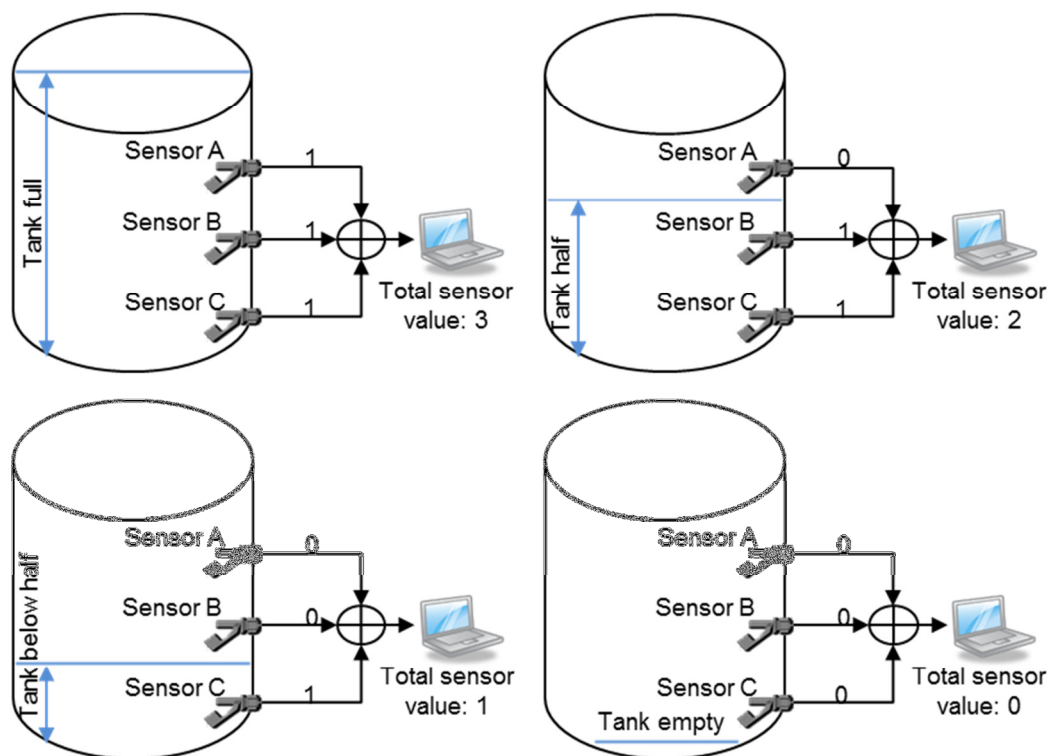


Figure 4.19: Water level sensing

External circuit: The Zigbit 900 module together with the SerialNet application supports up to three digital inputs. Thus, the three float level switches are directly connected, each one to a digital input pin and to digital ground. Thus, if the float level switches are activated, the reed-type switch will connect the digital input pin to ground making it a digital "0".

In software: When the state of these three inputs is added, a value of zero to three is obtained. Therefore, a resolution of up to four water levels within the tank is obtained and relayed back to the base station. The calculation is done by the sensor network server before the information is sent to the user interface.

Testing: Figure 4.20 shows the test setup for the measurement of water levels for this project.

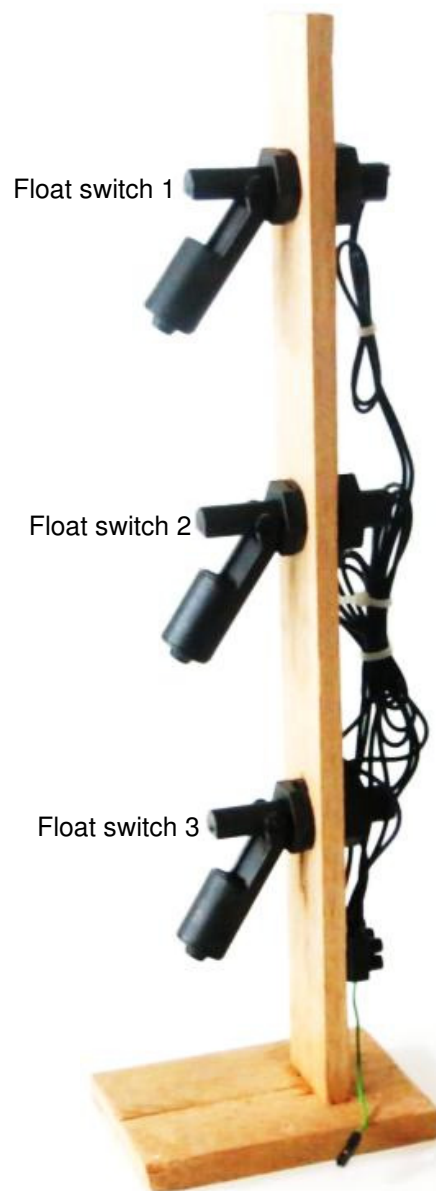


Figure 4.20: Water level sensor test setup

Gate and movement

The measurement of movement could be difficult and in many cases requires some advanced and expensive circuitry and sensors. However, for the farm application it should be kept as simple, robust, and generic as possible to achieve sufficient functionality to provide useful feedback for the user.

The instrumentation should be able to work together with the wireless communication system and already available hardware. Some of the options that were investigated are:

- reed switch
- micro switch
- PIR (passive infrared)
- accelerometer/vibration switches
- ultrasonic movement
- optical

Each of the listed options provides certain distinct advantages, but in most cases also disadvantages which will render the technology ineffective or difficult to implement for this specific application. Due to the physical construction of the typical gate on a farm, it is very difficult to provide solution that is reliable and robust that will not result in false triggers due to wind, for example. Furthermore, high solar radiation will cause most PIR solutions to become less efficient and is therefore eliminated as options for this project. Therefore, for demonstration purposes an optical solution, similar to what can be found at security gates in residential applications, was used since it is a proven, low cost, yet robust solution. This enabled a generic solution that could be used to measure a wide variety of movement, utilising a normal I/O port on the microcontroller.

Sensor type: An infrared beam typically used for security gates was used for this project after all the other option has been investigated and user requirements have been considered. The infrared beam provided a reliable cost effective but yet sufficient solution for measuring a wide variety of movement, including vehicles animals etc.

This circuit implements a 555 timer configured for mono stable operation. Thus, when an event occurs, in other words, the infrared beam is broken; the relay of the sensor will trigger the 555 timer. The output will then be an active high until the Zigbit 900 controller resets the event when it has been acknowledged by the sensor network server at the base station.

In software: The sensor together with the external circuitry utilised two GPIO's: a digital input and a digital output. When an event occurs, the 555 timer circuit will pull the digital input to ground. When this event has been acknowledged by the sensor network server, a signal is sent back to reset the 555 timer via a digital output pin of the Zigbit 900 module. The sensor will then again be ready to capture a movement. This event is then relayed to the user interface and alarm if applicable.

Testing: The test circuit and sensors are shown in Figure 4.22

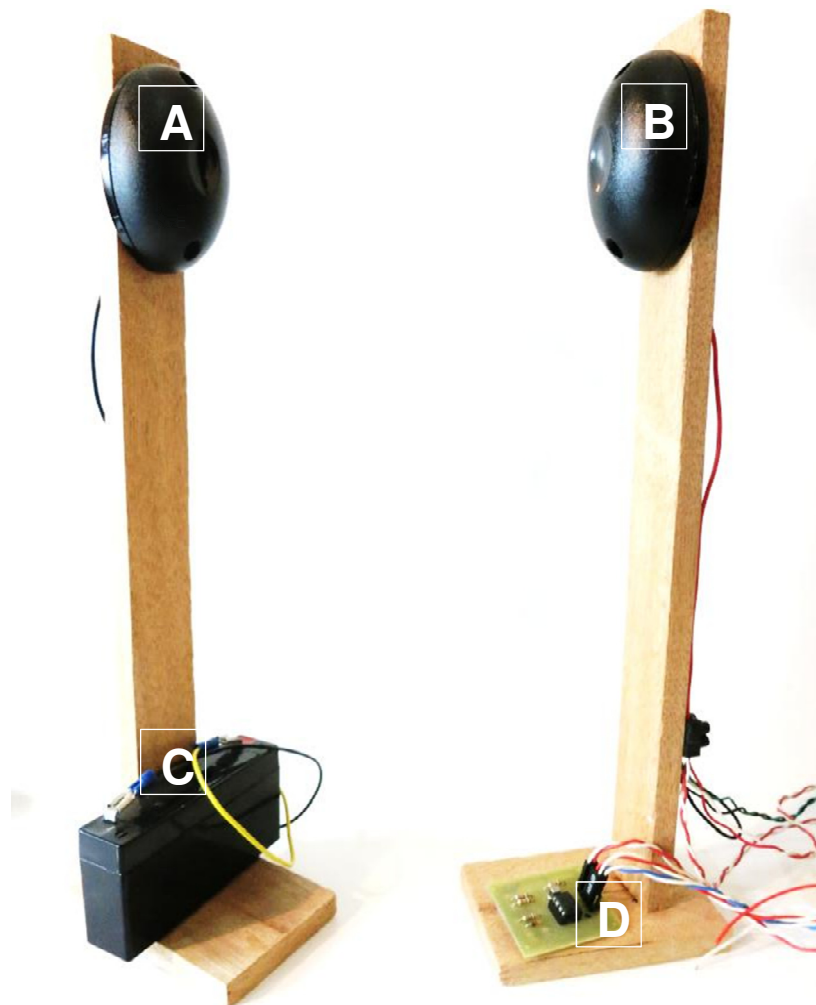


Figure 4.22: Movement sensor test setup

In Figure 4.22, items labelled A and B are the infrared transmitter and transceiver respectively and C is a 6 Vdc battery used as power supply. D is the 555 timer circuit.

Voltage measurement for DC equipment

The reference design required a DC voltage measurement of up to 20 Vdc. Section 3.4 mentions that the Zigbit hardware has an on-board analogue to digital converter available. This enables a simple method of measuring direct current electromechanical force (emf). This then provided a solution for measuring a wide range of equipment, even including industry standard 4-20 mA or 0-30 Vdc instrumentation signals.

Furthermore, parameters such as temperature and humidity could easily be added to the design as long as the sensor has an analogue output. However, for the reference project only the measurement of the battery levels of nearby telecommunication equipment was required. The terminal voltage of the battery varied between 10 and 15 Vdc. A basic instrumentation circuit was implemented for this that translated the 0-15 Vdc signal to a 0-1.25 Vdc signal. This signal could be measured by the microcontroller's ADC of the specific node.

The sensor: For the measurement of analogue signal, the Zigbit 900 module provides up to three separate on-board ADC's with an 8-bit resolution on the analogue measurements. These converters are referenced to a 1.25Vdc reference voltage regulator that is installed on the development board.

Configuration: To be able to measure voltages that are higher than the reference voltage, a few instrumentation options are available. However, the simplest of all would be a voltage divider circuit. With this circuit a certain EMF is reduced, using simple resistors to reduce the voltage that needs to be measured to a much smaller value while still representative of the larger value. The theory behind this is shown in Equation 4 and Figure 4.23.

$$V_{out} = \left(\frac{Z_2}{Z_1 + Z_2} \right) * V_{in} \quad \dots(4)$$

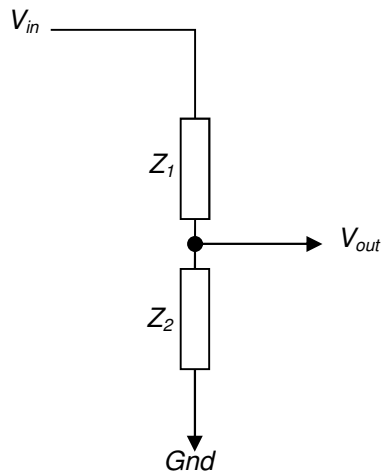


Figure 4.23: Voltage divider

External circuitry: A basic voltage divider circuit was implemented as external circuitry. This circuit translates the 0-15Vdc signal to a 0-1.25Vdc signal that can be measured by the microcontroller's ADC of the specific node on the Zigbit 900 development board. The external circuit for measuring the EMF's is shown in Figure 4.24.

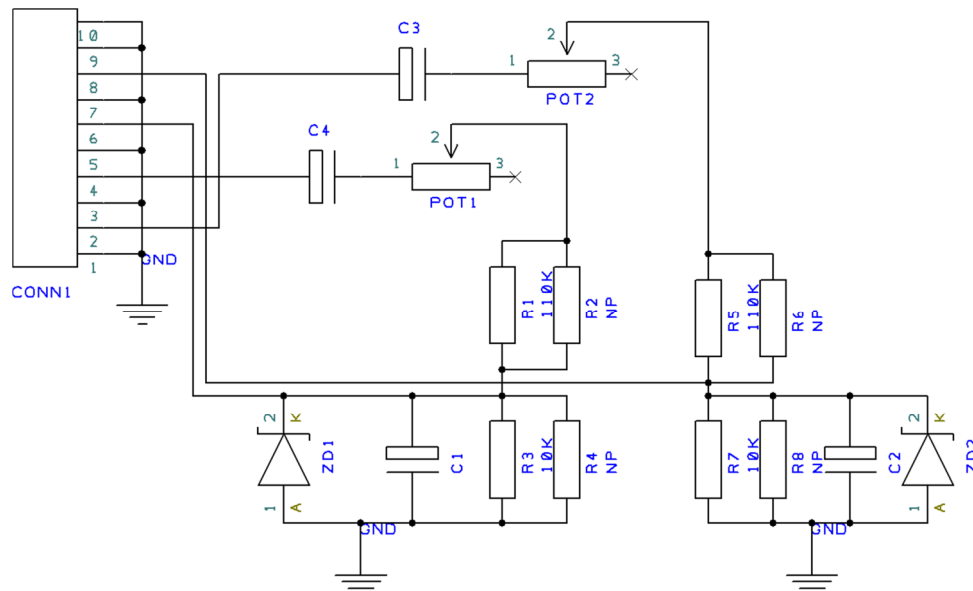


Figure 4.24: Voltage divider circuit

This circuit enables the measuring of two separate EMF's, with individual trimming pods for calibration and clamping Zener diodes for protecting the input of the microcontroller. If the measuring of any other EMF's is required, the ratio of the divider circuit could be changed to accommodate the different EMF's. This is possible as long as the maximum input EMF into the MCU's ADC does not exceed 1.25Vdc, which is the reference EMF for the microcontroller's ADC.

In software: An 8-bit ADC value is requested by the sensor network server. This 8-bit value is a representation of the actual voltage that is measured where 0 would be 0 Vdc and 255 would be 1.25 Vdc. The 8-bit value is then manipulated with a mathematical calculation that results in a value representing the actual voltage before the voltage divider circuit. The formula that will be used for this calculation is as follows:

$$V_{dc} = \left(\frac{V_{max}}{255} \right) * (8bit\ value) \quad \dots(5)$$

In Equation 5, Vdc is the actual measured voltage before the voltage divider, Vmax is the maximum voltage the divider circuit is designed for and the 8-bit value is the actual value acquired from the node.

Testing: Figure 4.25 shows the test setup for measuring DC voltages.

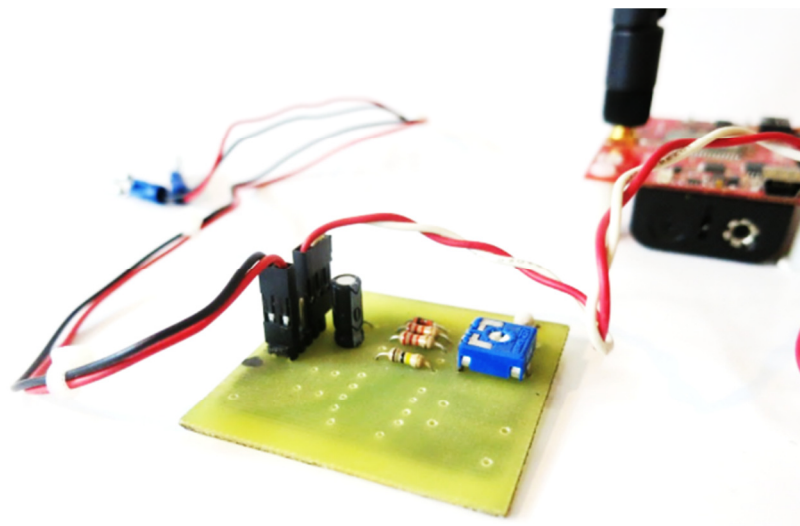


Figure 4.25: ADC test setup

On/Off relay control

Sensor: For controlling remote equipment, the easiest solution was to utilise a normally open relay that is controlled from the base station. This then enabled the user to control any external equipment that could be switched directly or via a larger contactor if necessary.

Configuration: The interface for switching equipment on or off is the GPIO interface that uses a single digital output pin. Therefore, relay driver circuits had to be implemented because of the limited voltage and current that can be supplied from the digital pins of the microcontroller on the Zigbit 900 development board.

External circuitry: The schematic for the external relay driver circuit is shown in Figure 4.26.

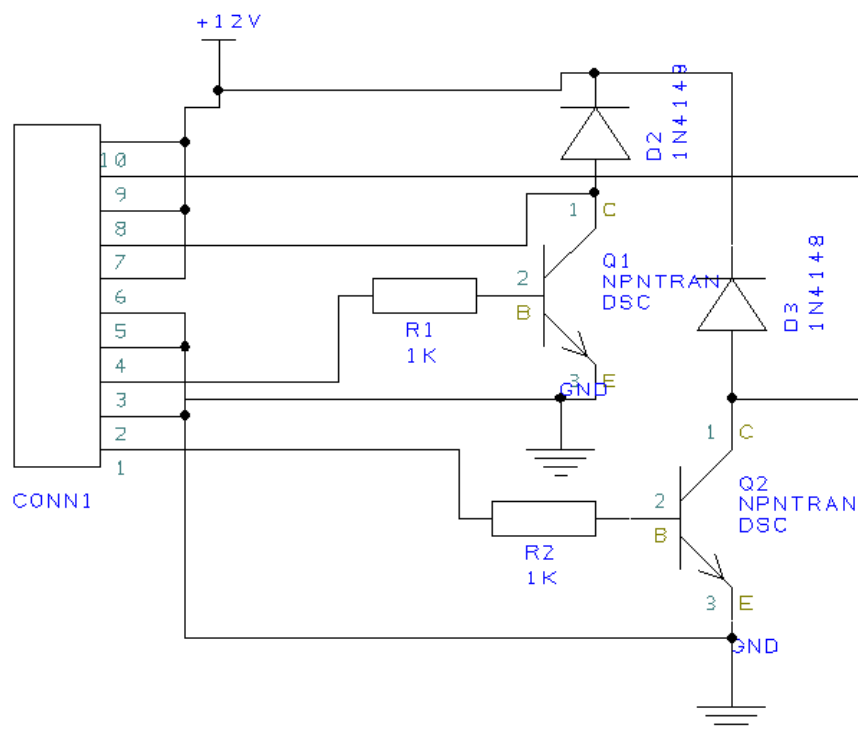


Figure 4.26: Relay driver circuit

Two individual driver circuits were implemented to be able to control two individual devices. A digital output of the Zigbit 900 module is connected to a transistor circuit that energises a 12 Vdc coil of a relay to switch on a device or trigger an event. A digital “1” or 3.3 Vdc will switch on the device and the digital “0” or 0 Vdc will switch the device off.

In software: The node for the on/off relay is the only node that will require input from the user. Therefore, a command will be issued by the user to switch a remote device on or off. This command will be given from the user interface, which is the web server. This command is then sent to the sensor network server and in turn transmitted to the associated node via the network coordinator. The node will then adjust the output pin to the appropriate value, either a digital “0” or a digital “1”.

Testing: Figure 4.27 shows the prototype board that was used for testing. An output pin of the Zigbit 900’s MCU is connected to the input pins of this board.

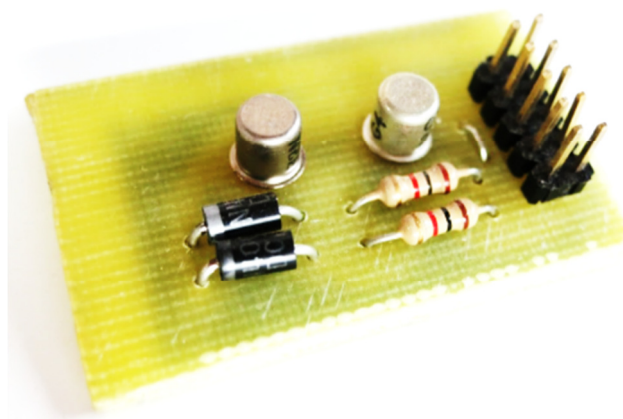


Figure 4.27: Relay driver prototype board

4.5.2 On/Off relay control and monitoring

Most of the remote equipment or events listed in Section 2.8 that have not yet been covered by the methods mentioned previously in this section could be monitored or controlled with on/off instrumentation. On/off monitoring and controlling of DC and AC equipment are easily implemented by means of relays and the I/O interface of the microcontroller.

Monitoring of remote equipment

Sensor: When the monitoring of remote equipment is required that has not yet been provided for in the previous designs, the GPIO interface of the Zigbit 900's microcontroller could be used. A single digital input will be implemented for this.

Configuration: For monitoring AC or DC equipment, a relay could be used. This relay connects one of the microcontrollers digital input pins to ground through the contact of the relay and can therefore indicate a one or a zero at the base station.

External circuitry: The electronic circuit to demonstrate this method is shown in Figure 4.28.

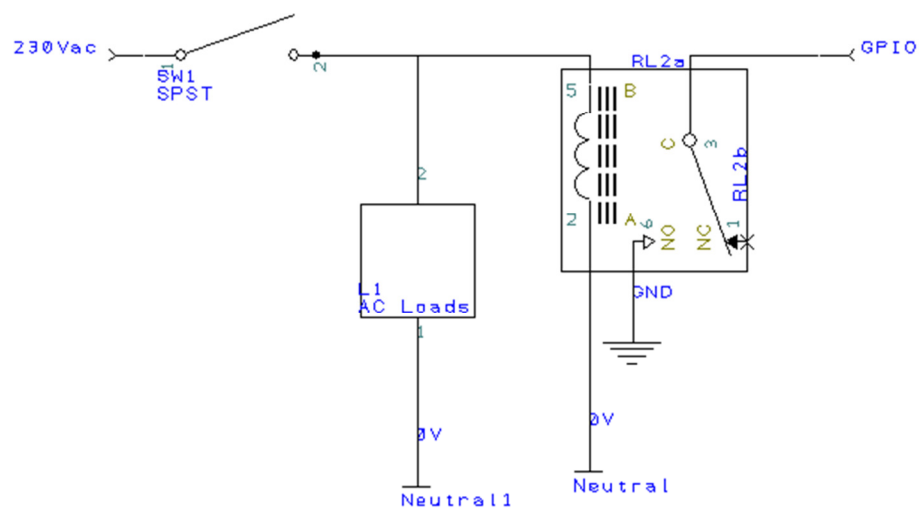


Figure 4.28: Relay monitoring circuit

An ideal non-intrusive solution would be to implement a clip-on current transformer to measure the AC current on the supply of such a unit. This method, however, would not be possible to implement using only the GPIOs of the microcontroller. In this case, external electronic circuitry or an ADC interface could be designed. This was, however, not required for the reference design and thus not implemented.

In software: A single digital input is implemented to verify one of two states, which is “on” or “off”. This state will be displayed on the user interface after it has been acquired by the sensor network server’s polling routine.

The event could be polled or updated by the user depending on the priority setup of the node. These events could also be configured as critical events that require alarm functionality.

4.5.3 Power supply

When a power supply for a remote telemetry system is considered, the nodes in the field should be self-sustainable. This is important because there is typically no power available in remote areas of the target market and loss of power will result in the system malfunctioning. Constant power supply in the system design also has significant impact on the cost of the final product. (Morais, 2008:1)

The power consumption for the Zigbit 900 Module from the manufacturer datasheet is noted as 15 mA and 20 mA for receiving and transmitting mode respectively (Atmel, 2009). A conservative power consumption of 500 mAh per day was assumed. This low power consumption is achievable because of the low transmitted power of <12 dBm. According to the datasheet the batteries used for powering the Zigbit 900 module could be used for up to 10 years. This will, however, only be true for end devices in sleep mode with a duty cycle of 1%. This battery life will not be achievable for this project since many of the nodes were implemented as routers that cannot be put into sleep mode because of the Zigbee protocol requirements (Cirronet, Inc, 2005). A generic power supply was therefore chosen instead, regardless of the node configuration.

Furthermore, in addition to the Zigbit 900 module the nodes, sensors, and electronics installed in the field all require a power source for operation. An external power supply had to be implemented for the additional power needs, providing enough power for the transmitting and receiving of data by the Zigbit 900 module. In many cases the instrumentation will require an industry standard 12 Vdc power supply.

The power supply that was implemented for demonstration purposes for this project consisted of a 10 Wp solar panel, 7 Ah, 12 Vdc battery, and a solar regulator.

This configuration for the power supply was chosen because the components were easy to acquire for the testing purposes, although it is not necessarily the most optimal or cost effective solution. However, this provided ample power for the sensors, electronics and Zigbit 900 module. Implementation of the power supply is easy because of the standard, off-the-shelf components that were used.

Also, the advantage of this configuration with the choice of components is that it provides an industry standard 12 Vdc supply that can be used for any sensor or auxiliary components. Figure 4.29 shows a schematic representation of the power supply.

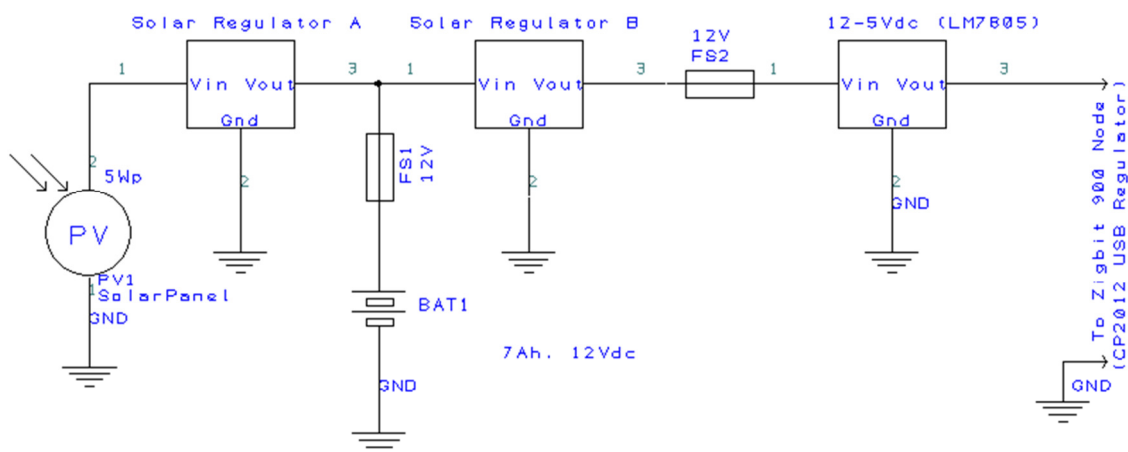


Figure 4.29: Power supply

Furthermore, the Zigbit900 evaluation board that was used for this project has an integrated USB to serial converter (SI2012) on-board. This device has a built in voltage regulator and can be used to power the Zigbit 900 development board. Also, a USB connector is available that can be used to supply the Zigbit 900 module with power using the on-board voltage regulator. A 12 Vdc to 5V dc linear regulator is then used for the power supply from the 12Vdc battery and is shown in Figure 4.30.

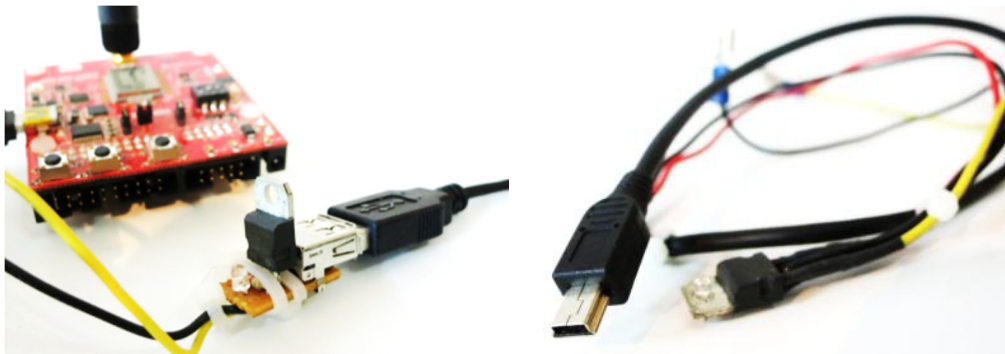


Figure 4.30: USB cable regulator for Zigbit 900

4.5.4 Prototype of end node and repeater

For the purposes of this research project, a field test prototype was implemented to verify the required parameters of the project in terms of functionality and other practical considerations. This was a fully functional unit that relayed the battery voltage back to the user interface. This prototype is shown in Figure 4.31 and Figure 4.32.

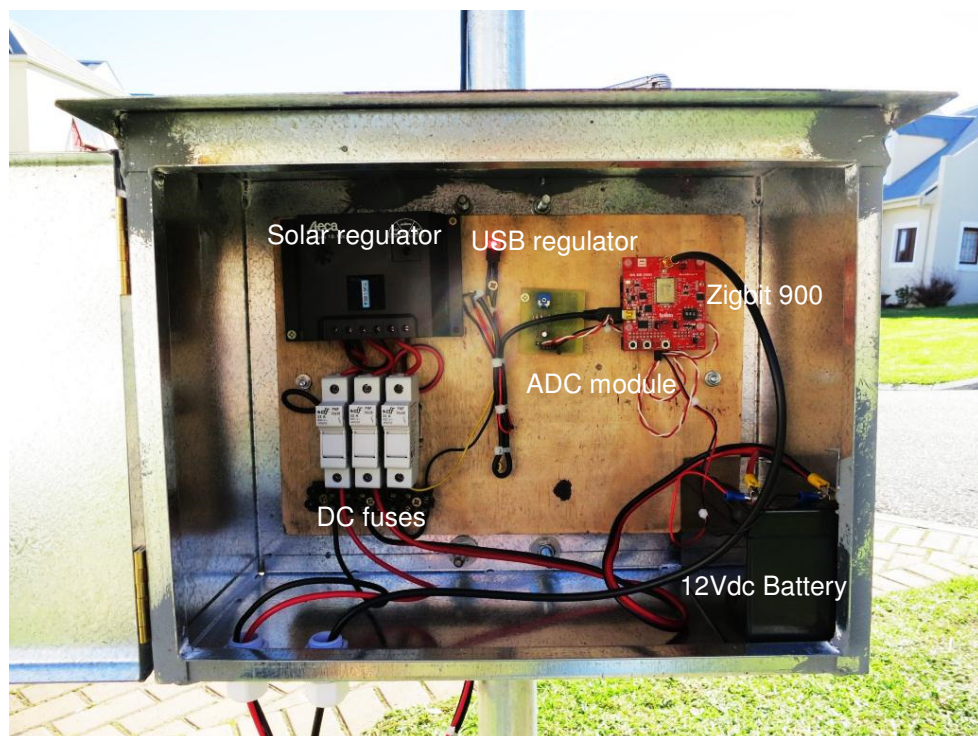


Figure 4.31: End node prototype unit



Figure 4.32: End node prototype unit

4.6 System cost

As part of the system design it is important to establish the cost incurred for the development of the system. For this initial cost estimation, the cost of the individual components are added in an attempt to estimate the monetary value of such a system to be used as an guideline in the financial feasibility study. The cost for each part of the system will be discussed in the following sections.

4.6.1 Base station

Table 4.5 shows the cost breakdown of the components and development tools that were used in the development of the base station. All these amounts are retail and include VAT (value added tax).

The breakdown in Table 4.5 is further refined and an estimate of what the actual cost for developing such a system could be is shown in Table 4.6.

The amount for the final base station is substantially less than the prototype base station since the use of development or evaluation platforms are not necessary. However, the development and other cost are not included in this breakdown in Table 4.6 as the estimation of these additional cost falls beyond the scope of this document.

Table 4.5: Cost of base station (prototype)

Item	Supplier	Qty	Cost	Total
Explorer 16 evaluation board	Microchip	2	950	1900
SPI Ethernet controller	Microchip	1	280	280
SPI SD/MMC card module	Microchip	1	275	275
Zigbit 900 evaluation board	Atmel/Meshnetics	1	1230	1230
Power supply (AC/DC)	Generic	1	100	100
RF COAX (LMR195)	RF Design	10	10	100
SMA connectors (m/f)	RF Design	2	12	24
Enclosure	Generic	1	100	100
Cables/Connectors, etc.	Generic	1	50	50
				R 4,059.00

Table 4.6: Base station (final product)

Item	Supplier	Qty	Cost	Total
Main PCB for components	PCB manufacturer	1	50	50
Microcontroller	Microchip	2	35	70
Additional components	Generic	1	100	100
Misc. connectors	Generic	1	100	100
Zigbit 900 Module	Atmel	1	150	150
PCB for Zigbit 900	PCB Manufacturing	1	25	25
Additional components (ZB900)	Generic	1	100	100
Additional Connectors	Generic	1	100	100
Enclosure	Generic	1	100	100
Power supply	Generic	1	100	100
Additional cable, etc.	Generic	1	100	100
				R 995.00

4.6.2 End Node/Repeater

Table 4.7 shows the cost breakdown of the components and development tools that were used in the development of the end node, including the instrumentation required for on-site measurements of equipment or other parameters. All these amounts are retail and include VAT (value added tax).

Table 4.7: Cost of repeater/end node (prototype)

Item	Supplier	Qty	Price	Total
Zigbit 900 MHz evaluation board	Atmel/Meshnetics	1	1230	1230
Instrumentation PCB	Generic	1	25	25
Sensor (Float, Relay, Movement)	Generic	1	300	300
Battery (6Ah, 12Vdc)	Generic	1	50	50
Solar regulator (6A, 12/24Vdc)	Steca	1	190	190
Solar panel (10Wp, 12Vdc)	Generic	1	230	230
Weather proof enclosure	Generic	1	250	250
Connector, Cabling etc.	Generic	1	100	100
RF coax	RF Design	2	10	20
SMA connectors (M/F)	RF Design	2	12	24
Mounting pole	Generic	1	50	50
				R 2,469.00

The breakdown in Table 4.7 is further refined and an estimate of what the actual cost for developing such a system could be is shown in Table 4.8. Since the use of development or evaluation boards is not necessary for the final product, the amount for the repeater/end node is substantially less than that of the prototype. However, the development and other cost are not included in the breakdown of the final product since the estimation of these additional cost falls beyond the scope of this document.

Table 4.8: Cost of repeater/end node (final product)

Item	Supplier	Qty	Cost	Total
Zigbit 900 Modules	Atmel	1	150	150
Additional components & PCB	Generic	1	125	125
Instrumentation PCB	Generic	1	25	25
Sensor (Float, Relay, Movement)	Generic	1	300	300
Battery (6Ah, 12Vdc)	Generic	1	50	50
Solar regulator (6A, 12/24Vdc)	Steca	1	80	80
Solar panel (5Wp, 12Vdc)	Generic	1	150	150
Weather proof enclosure	Generic	1	250	250
Connector, Cabling etc.	Generic	1	100	100
RF coax	RF Design	2	10	20
SMA connectors (M/F)	RF Design	2	12	24
Mounting pole	Generic	1	50	50
				R 1,324.00

Even though the base station is responsible for most of the functionality in terms of user interface and network coordination, the end nodes (or repeaters) are more expensive than the base station when compared. This is mainly due to the stand-alone power supply needed for the end nodes. The power supply consists of a solar panel, regulator, and battery combination and the external sensors that need to be implemented. These results will be discussed in more detail in the results section.

4.7 Instrumentation

Table 4.9 gives an overview of the cost of the sensors that is used for this project. The cost incurred for the sensors are relatively low with a maximum of R350, as seen in Table 4.9.

Table 4.9: Sensor cost

Function	Item	Cost
Motion sensor	Infrared sensor	R250
Water level sensing	Float switches (x3)	R350
On/Off monitoring	Relay	R100
On/Off controlling	Relay	R100
Voltage monitoring	External Circuit	R0

CHAPTER FIVE

Test Results

To answer the research questions of this project, certain parameters of the system needed to be tested and verified to prove that the technology is technically feasible for this project. Furthermore, a financial breakdown needed to be investigated to prove the system is economically viable.

5.1 Technical feasibility

The following section discusses the results after tests for each individual component were concluded.

5.1.1 Range of communication

With the application in mind, one of the most important parameters of the communication system used for the project is the range that can be achieved by the hardware for a specific transmitted power. In South Africa, the transmitted power is limited by the ICASA regulations. The range for a specific, limited transmitted power was tested in the Northern Cape in conditions typical to the market sector relevant to this project. Furthermore, an additional test was carried out in Stellenbosch with very few similarities, especially considering that Stellenbosch is a highly populated area with the presence of various other wireless communications. These communications flood the electromagnetic spectrum which should result in reduced performance. An example is cell phone networks which operated in the same frequency band as that of the Zigbit 900 modules.

For the test in the Northern Cape the first of the two nodes that was used for the range test, was mounted on a 2.5 meter pipe. This node served as the stationary node that will transmit random packets of data continuously to the receiving node. The receiving node was then connected via a USB (virtual serial communication) port to a laptop that was moved away from the stationary node until acceptable communication was no longer available.

This then gave a measure of the performance over a long range that will be typical to the application environment for this project. This was a preliminary hardware test that could verify that the maximum range of communication is the same as on the technical datasheet. This range was a concern since very few other solutions on the market could achieve the range of 6 km as with the Zigbit 900 modules. The typical range of wireless sensor networks is usually around 100 meters which made it difficult to suppose that the Zigbit 900 modules could achieve a range of up to 6 km.

For the Stellenbosch test, a fair amount of the system development was concluded which included the instrumentation and user interface. This test therefore included the configuration of three nodes as with a typical application. With this configuration, an end node, repeater, and the coordinator were used and accessed through the user interface that was developed for this project. This was done to verify the functionality of the system over an extended range.

Figure 5.1 shows the placement of the transmitter and receiver with the distance between these nodes for both the Northern Cape and Stellenbosch tests. The screenshots were made using Google Earth. From the screenshots can be seen that the range is in excess of 6 km (Northern Cape test) which showed the transmitter and receiver hardware will be suitable for the project. The range that was achieved during the test in Stellenbosch is less than the Northern Cape test. This could be due to nearby cell phone towers and other radio frequency interference. Furthermore, the presence of buildings and trees, and an uneven ground level will most probably have an impact on the range as well. However, since this system will be implemented in rural areas similar to the Northern Cape test, the Stellenbosch test served as a worst case scenario even though it still achieved a range of 5.68 km.

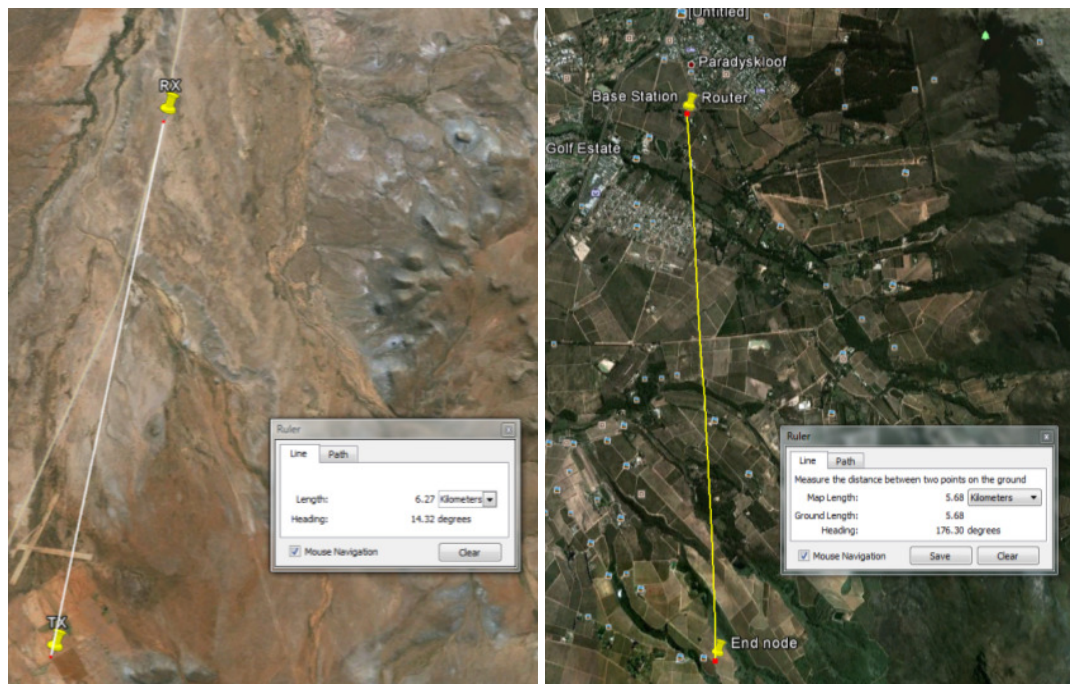


Figure 5.1: Range test results

Table 5.1 shows the test conditions of the Northern Cape test.

Table 5.1: Test conditions

Parameter	Value
Humidity	Low (<15%)
Ambient temperature	30°C
Time of day	17:00
Power source	2xAA, 1.5V alkaline batteries
RF interference	None
Line of sight	Yes
Transmitted power	Max (+11dBm)

Zigbee protocol and SerialNet

The *BitCloud Zigbee PRO* stack was utilised which provided the full functionality required for the communication for this project. The SerialNet application that was used as an interface between the user interface and the wireless sensor network provided sufficient control and monitoring functionality for the purposes of this research project.

This is a standard application that was provided with the development kit from the manufacturer, again making it an off-the-shelf solution. Furthermore, mesh networking is supported with the *BitCloud Zigbee PRO* stack that made it ideal for long range communication for this project.

Power supply

The power supply as described in this document was tested and found to be more than adequate for the purposes of this project. The size of the components that was used for the demonstration purposes can be reduced in future designs to reduce the cost. The power supply provided a constant 12 Vdc supply for the external sensors, the circuitry and also the sensor node through a 5Vdc linear voltage regulator.

User interface

The interface as discussed was developed to verify its ability to supply the required functionality for the system. Successful monitoring, controlling and logging for the sensor network were achieved with the demonstration system. This was done over the internet using TCP/IP and was accessed via a web browser on a computer and even a cell phone. This interface was mainly used to prove the concept for functionality and will still require development that could provide a stable, user-friendly interface for commercial purposes. Furthermore, since the interface was implemented on an embedded platform, the cost for the interface was very low. No input or output devices were required other than a personal computer or smartphone, which is most probably already available from the user. The availability of a personal computer or smartphone assisted in meeting the initial user requirements that specified that a low cost solution is required.

On-site instrumentation

One of the challenges of this project was to implement a cost-effective yet sufficient method for controlling and monitoring remote equipment, events or physical parameters from a central point on a farm. Industry standard solutions were sought to meet the requirements for the project and are discussed in the market research section of this document. The individual sensor components were implemented and tested as part of the systems.

These tests showed that the sensors provide generic, robust and sufficient functionality without adding unnecessary complexity, cost or power consumption to the system. Table 4.9 indicates that the cost for each sensing technique is relatively low and adheres to the initial specification, stipulating that a low cost solution needed to be implemented.

5.2 Financial feasibility

In this section the financial feasibility of the proposed wireless telemetry system is discussed. This forms an important part of answering the research questions as well as validating whether objectives were met as mentioned in Chapter 1 of this document.

5.2.1 Cost for remote monitoring system

From Table 4.6 and Table 4.8 the cost for the system is estimated to be R1 000 per base station, and R1 300 per router and end device. These component costs are roughly estimated based on retail values per one unit and should be less when high volume procurement and manufacturing are considered. For the average farm, as shown in Table 2.1, one base station and 13 end devices or routers will be required for a complete system. This means that the total estimated cost for a remote control and monitoring system for the average farm will amount to around R18 000. This is summarised in Table 5.2.

Table 5.2: Estimated system cost

Item	Qty	Cost	Total
Base station	1	R1 000	R1 000
End nodes	13	R1 300	R16 900
		Total	R17 900

5.2.2 Current cost of farming

In an attempt to accurately estimate the cost involved for physically monitoring all the remote sites around a farm, three methods were used. Firstly, information of all the physical parameters were gathered and certain assumptions were made.

This involved adding the distances between remote sites and multiplying these with a calculated cost for travelling and labour. Secondly, farmers in the area were asked what they would be willing to pay for a system that would enable remote control and monitoring. This was done to understand the willingness of the farmer to implement such a system and also to establish if there is a need for such a system. Lastly, financial statements were evaluated. This provided accurate insight into actual costs for labour, vehicle cost and fuel expenses as discussed in Section 2.1. The three estimation methods are summarised in Table 5.3.

Table 5.3: Cost of physically monitoring sites (budget)

Item	Amount per annum
Survey (calculated from survey results)	R63 380
Survey (farmer willing to pay)	R8 700
Financial statements of farmers	R255 000

Table 5.3 shows that the total cost for physically monitoring the points of interest on the farm is theoretically calculated from the survey results in Section 3 and amounts to R63 380 per annum.

The cost for the remote monitoring and controlling system for an average farm as described for this project is estimated at R18 000. Therefore, the estimations gathered as part of this study show that a remote monitoring and controlling system can be implemented with a payback period of less than a year.

Furthermore, the financial figures obtained from financial auditors indicated that the actual cost for monitoring these points could even be more significant, as shown in Section 2.1. This section indicated the relevant labour and travel cost to be 34% of total the cost of farming (R255 000 per annum), which is significantly more than R63 380, or 10% of the actual financial figures.

This discrepancy could be due to a significant part of travel and labour cost that is not necessarily spent on monitoring points on the farm. Traveling and other farming activities are therefore included in the R255 000. Also, the information acquired via the survey is estimations and opinions that could have an impact on the financial result.

In Section 2.3 a budget of R76 500 was discussed by which a remote monitoring and controlling system will be proven feasible. In conclusion to the system development, the cost of implementing such a system was found to be far less than R76 500. The budget is therefore feasible and should be accepted by these farmers.

5.3 QFD: User requirements

From the product planning phase discussed in Section 2.8, the customer needs have been converted into product specifications. These specifications are summarised in Table 5.4 as requirements. To ensure the success of this product, these requirements had to be met. Therefore, the main aim during the development stages was to overcome these challenges inherent to this project. Table 5.4 indicates whether or not the needs of the customer were successfully addressed during the development stages.

Table 5.4: QFD results

Requirement	Expected (from research)	Actual result (after development)	Success
Long range	Excellent	<6 000 m	Yes
Low cost	Excellent	<R1 300 per node	Yes
Low power consumption	Moderate	<6 Wh/day	Yes
No licence/running cost	Excellent	TX power <10 dbm	Yes
Effective/easy to use	Excellent	Used by semi-skilled staff	Yes
Plug and play	Excellent	Not self-configuring/healing	No
Bi-directional	Excellent	Control and monitoring	Yes
Mesh networking	Excellent	Maximum range achieved	Yes

Table 5.4 shows that most of the critical design requirements were met, with the exception of only one. The requirements that were not met could easily be catered for with further development and should not cause the solution to fail the initial product requirements.

In conclusion, the development of this telemetry system has been very successful in accurately selecting the appropriate technology as well as meeting the initial specifications for the design and market purposes of this project.

CHAPTER SIX

Conclusions and Recommendations

In Chapter 1, four research parameters were stated that guided the research and development for this project. This section summarises the results, indicating that the individual goals have been met. Recommendations for future work will also be discussed.

6.1 Conclusions

Initially, using standard “off-the-shelf” technologies seemed to be a challenge in terms of meeting all the user requirements. However, Table 5.4 summarises the results and concluded that most of the critical design parameters had been satisfied by using these standard technologies. The main design limitations for this project were long range with limited output power; both range and transmitted power requirements, however, had been met. Furthermore, the outcome of this project is a low cost solution. In addition, low power consumption and no licensing fees reduced the running cost of this system. The conclusion is therefore that this system meets the specific user requirements and is technically achievable.

The information obtained from the user survey as discussed in Section 2.1.2 provides a good estimate of time and resources spend for physically monitoring events or equipment. This information aided in estimating the increase in farming efficiency by calculating the labourer hours and cost that theoretically could be reduced. Table 2.2 and Section 2.3.1 indicates that an estimated 715 hours and up to R63 380 per year is spent on monitoring equipment and events.

In Section 5.2, a financial break down was discussed. The discussion showed that the system cost is relatively low when a breakeven analysis is done. To summarise, the total system hardware cost is estimated at R18 000 even though a budget of up to R76 500 was set out to be feasible. The total estimated savings that could be achieved is R63 380 because of the reduced travel costs and labour hours.

This indicates a payback period of 3 months with an annual saving equal to 10% of total expenses for these extensive, rural farming enterprises. The conclusion is that such a system should be financially feasible and hence economically viable for the specific farming industry when the practical and financial benefits are considered.

From the research conducted, components were identified that will be able to meet all the required needs for the application. This included a specific hardware platform, the Zigbit 900 from Atmel. This provided the physical wireless communication interface for a range of up to 6 km and a Zigbee PRO protocol stack that provided the embedded application for the management and interfacing of the Zigbee network.

The user interface was based on a hardware and software solution provided by Microchip and proved to be adequate for the purposes of this project. However, to be able to integrate all the different hardware and software platforms into one working system required custom embedded software to be developed. This software was necessary in order to handle the communication, process data, and to customise the presentation of the data. This was also done on a standard Microchip hardware platform. This resulted in a simple, yet effective user interface platform that could provide alarm and logging functionality while providing external access through a computer or smart phone connected locally or over the internet. The achieved functionality meets the user requirements as set out in Section 3.5.

6.2 Recommendations

In conclusion, this research project proves that the designed system is technically and financially feasible, with the potential to benefit from continued research. The following is recommended to ensure that the system meets the needs of the market completely, resulting in a stable and reliable solution that will be trusted by the user.

Pilot project implementation

A working prototype together with various bench tests have been constructed that yielded very good results. Testing the basic functionality proved the technology feasible in the field and in the lab.

Since no extra nodes could be acquired for this project and also because of time and financial constraints, the system was never implemented in a large scale, real life application. The recommendation is therefore to install a full scale and fully functional system on a farm in the specific region in order to test the system's functionality fully. While this recommendation will require significant costs to be incurred, such an installation will be able to prove the technology effective by introducing significant savings to the day-to-day activities on the farm. This will also provide real-life technical feedback that could be used for improvements of the technical aspects of the system. Also, the system will be implemented in a real-life environment, which will provide important user feedback and aid in the understanding of user patterns and therefore further refine the user specifications.

User interface

The user interface that was developed for this system only provided basic functionality for testing purposes. Also, as a text-based interface, it is not as intuitive as it should be, especially when considering that GUI's are very common these days. Therefore, the recommendation is to develop a fully functional and stable graphical user interface for the control and monitoring of nodes.

Zigbee network functionality and hardware platform

Table 5.4 shows that the most limited parameter is "plug and play". For the purposes of this project, each node was initially configured manually to be able to join the Zigbee network. This function is essential for easy installation. Further development is therefore needed to ensure that when the system is implemented for the first time or when new nodes are added, the network will be self-configuring. The nodes should also be able to slot into the network without any technical complications. Also, for testing purposes, most of the components of the wireless controlling and monitoring system have been developed and implemented on a development or evaluation platform. Using this hardware has many advantages but also introduces various limitations. To be able to increase functionality and reliability, the manufacturing of specialised hardware is recommended. Customised hardware will enable the implementation of any number of new nodes for testing purposes since this was a major constraint during the development and testing stages of this project.

REFERENCES

AA Travel, 2012. www.aa.co.za. <http://www.aa.co.za/on-the-road/calculator-tools/vehicle-operating-costs.html>

[Accessed 18 April 2012].

Alta Pacific Technology Solutions Group, inc, 2010. Alta Pacific Technology Solutions Group, inc. <https://altapacific.com/solutions-telemetry.html>

[Accessed 16 April 2012].

Atmel Corporation, 2009. ZigBit™ 700/800/900 MHz Wireless Modules – Datasheet. <http://www.atmel.com/Images/doc8227.pdf>

[Accessed 28 May 2012].

Atmel Corporation, 2011. BitCloud SerialNet User Guide.

<http://www.atmel.com/Images/doc8389.pdf>

[Accessed 01 May 2012].

Atmel, 2009. ZigBit™ 700/800/900 MHz Wireless Modules.

http://www.atmel.com/dyn/resources/prod_documents/doc8227.pdf

[Accessed 16 December 2011].

Bestbier, Mr. 2008. Interview with employee on 16 June 2008. Tramirloc, Stellenbosch.

Botma, Mr. 2008. Interview with employee on 0.July 2008. Spectrum Communication, Cape Town.

CellC, 2012. Cell C Coverage Map. C the power is in your hands, Autumn, Issue 6, p. 37.

Cirronet, Inc, 2005. www.cirronet.com.

http://www.cirronet.com/pdf/wp_zigbee_engineering_options.pdf

[Accessed 23 February 2012].

Cirronet, 2002. White paper on Wireless communications for industrial applications.
www.industrialnetworking.com/pdf/cirronet_wireless.pdf

[Accessed 6 August 2009].

Coleman, C., 2004. An Introduction to Radio Frequency Engineering. Cambridge:
Cambridge University Press.

Crow, K. A., 2002. <http://www.npd-solutions.com/>. <http://www.npd-solutions.com/qfd.html>

[Accessed 28 July 2012].

Cunningham, B., 2008. Radio Theory De-Mystified.

http://www.cooperindustries.com/content/dam/public/bussmann/Wireless/Resources/White%20Papers/BUS_WIR_WHITE%20PAPER_OM_RadioTheoryDemystifiedWINA.pdf

[Accessed 16 December 2011].

Department of Primary Industries, 2009. Department of Primary Industries.

<http://www.dpi.vic.gov.au/agriculture/about-agriculture/projects-and-activities/farm-monitor-projects/monitor-project/07-08>

[Accessed 16 December 2011].

Dreyer, S. A., 2001. Survey of Instrumentation and Measurement. s.l.:John Wiley & Sons.

Elpro Technologies, 2008. White paper on performance of different frequency bands.
www.elprotech.com/UserFiles/File/PDFs/white_paper_-_frequencies_1.3.pdf

[Accessed 7 August 2008].

Farahani, S., 2008. Zigbee Wireless Networks and Transceivers. Burlington: Elsevier.

G. Vellidis, V. G. S. P. C. P. C. K. M. T., 2007. <http://www.nespal.org>.

<http://www.nespal.org/tapac/docs/vellidis.how.wireless.will.change.agriculture.6ecpa.pdf>

[Accessed 03 August 2012].

Galitz, W. O., 2007. The Essential Guide to User Interface Design. 3 ed. Indianapolis: Wiley
Publishing, Inc..

Ghosh, A. k., 2005. Introduction to Instrumentation and Control. New Delphi: Prentice-Hall of India.

Gnome Technology, 2008. White Paper: Why GPRS is ideal for telemetry.
<http://www.gnome-technologies.com/downloads/GPRS%20Telemetry%20Whitepaper.pdf>
[Accessed 14 April 2012].

Google, 2012. Google Earth. <http://www.google.com/earth/index.html>
[Accessed 01 June 2011].

Hawkins, D. H.-J., 2010. <http://www.agrisa.co.za>.
<http://www.agrisa.co.za/Dokumente/Hulpbron/GreenChoice.pdf>
[Accessed 28 July 2012].

Haykin, S., 1988. In: Digital Communications. Canada: John Wiley, p. 273.

Haykin, S., 1988. Digital Communications. s.l.:John Wiley & Sons.

Haykin, S., 1994. Communication Systems. Canada: John Wiley.

Ipatov, V., 2005. Spread spectrum and CDM: principles and applications. Chichester: John Wiley.

Le, K. T., 2005. EETimes.
<http://www.eetimes.com/design/industrial-control/4012593/ZigBee-SoCs-provide-cost-effective-solutions>
[Accessed 16 April 2012].

Microsoft Encarta online encyclopaedia, 2008. Electromagnetic Radiation.
http://encarta.msn.com/encyclopedia_761578834/Electromagnetic_Radiation.html
[Accessed 6 August 2008].

Murty, D., 2004. Transducers And Instrumentation. New Delphi: PHI Learning Pvt. Ltd..

Patranabis, D., 1999. Telemetry Principles. 10 ed. s.l.:Tata McGraw-Hill.

Perlegos, H., 2009. Atmel Acquires MeshNetics ZigBee Intellectual Properties, San Jose, CA: Atmel Corporation.

South African Government, 2000. Department of environmental affairs.
<http://www.environment.gov.za/enviro-info/nat/images/biomes.jpg>
[Accessed 01 May 2012].

SouthAfrica.info, 2008. <http://southafrica.info>.
<http://www.southafrica.info/business/economy/sectors/542547.htm>
[Accessed 16 December 2011].

SouthAfrica.info, 2008. South African agriculture.
<http://www.southafrica.info/business/economy/sectors/agricultural-sector.htm>
[Accessed 16 December 2011].

Stewart, D., 2007. EEPublishers.
http://www.eepublishers.co.za/images/upload/teltech_zigbee.pdf
[Accessed 16 December 2011].

Technology Training Limited, 2012. Technology Training Limited.
http://www.technology-training.co.uk/understandingtheiso7layermodel_10.php
[Accessed 16 April 2012].

Texas Instruments, 2005. <http://www.ti.com>. <http://www.ti.com/lit/an/swra048/swra048.pdf>
[Accessed 28 July 2012].

The Zigbee Alliance, n.d. Zigbee Alliance. <http://www.zigbee.org/About/FAQ.aspx>
[Accessed 16 April 2012].

van Zyl, W., 2008. The New Product Development Process: Small Firm Success by Studying Larger Firms. Stellenbosch: Stellenbosch University.

Vena, V., 2010. Anatomy of a farm murder. <http://mg.co.za/article/2010-04-08-anatomy-of-a-farm-murder>.

[Accessed 16 December 2011].

Vengalil, J. A., 1985. Economic Geography - Factor influencing the location of economic activity. New Delhi, India: Concept Publishing Company

Vodacom, 2011. Vodacom Coverage maps.

<http://www.vodacom.co.za/personal/main/coveragemaps>

[Accessed 16 December 2011].

APPENDICES

Appendix 1

Questionnaire used for market survey

Navorsing: Kaapse Universiteit van Tegnologie (CPUT) - Julie 2008

1. Hoe groot is die gemiddelde plaas in u streek en hoeveel plase word per boer besit?

2. Hoeveel belangrike punte is daar op u plaas wat u gereeld moet nagaan? (bv. Waterpunte, pompe, damme, hekke, ens.)

Vir 5 punte wat u gereeld nagaan:	Punt 1	Punt 2	Punt 3	Punt 4	Punt 5
Spesifiseer die punt.					
Hoeveel keer 'n week word dit nagegaan?					
Hoe ver is dit vanaf u huis?					

3. Wat sou u graag nog wou monitor op u plaas? Enige iets wat u tans moeilik vind om te monitor?

4. Wat dink u, sou boere bereid wees om te betaal vir 'n sisteem wat punte outomaties kon monitor?

Bv. u kan vanaf u plaashuis watervlakke, pompe, krane, ens monitor.

5. Sou dit wenslik wees om sekere van bogenoemde punte te kan beheer vanaf 'n sentrale punt? (Spesifiseer asb.)

6. Hoeveel is u bereid om te betaal vir 'n sisteem wat die punte kan beheer? (bv. pompe aan of af skakel)

7. Oor watter van die volgende kommunikasie middele beskik u oor op u plaas?

Telkom outomaties Selfoon "Wireless" Internet Tweerigting radios

Ander: _____

8. Is u bewus van ander moniteringsoplossings soos die wat "wireless" tegnologie bied? Spesifiseer.

Appendix 2

Wireless telemetry technology comparison chart

	WiFi	Zigbee	Proprietary WSN	SCADA	Wireless HART	Generic Digital Communication	Satellite communication	Simple analogue communication	CB Radio
Implementation cost	2	1	1	3	3	1	3	1	2
Hardware complexity	3	2	1	3	2	2	3	1	2
Functionality	3	2	2	2	2	2	3	1	2
Software Licence fees	No	Yes	Free	No	No	No	No	No	No
Subscription fees	No	No	No	No	No	No	Yes	No	No
ICASA Licence fees	No	No	No	No	Yes	Range Dependant	No	Range dependant	Yes
Maintenance	2	1	1	2	2	1	2	2	2
Power supply	2	1	1	2	2	2	2	2	2
Range	2/3	2/3	2/3	na	2	1-3	1	1-3	1
Implementation (end user)	2	1	1	2	2	1	2	2	2
Architecture	Mesh	Mesh	Mesh	Infrastructure	Infrastructure	Design dependant	Point to multi point	Point to multi point	Point to Point
Procurement	1	2	2	2	2	2	2	2	2
Product Development	1	2	2	2	2	3	3	3	2

1- advantage, 2 - medium, 3 - disadvantage

Appendix 3

Zigbee comparison chart

Manufacturer	Module	Frequency	Sensitivity	Max TX Power
Atmel	Zigbit 2.4GHz	2.4GHz	-101dBm	3dBm
	Zigbit 2.4GHz Amplified	2.4GHz	-104dBm	20dBm
	Zigbit 700/800/900MHz	700-900MHz	-110 dBm	11dBm
Jennic	JN5139-xxx-M00/1/3	2.4GHz	-96dBm	2.5dBm
	JN5139-xxx-M02/4	2.4GHz	-100dBm	19dBm
ST Microelectronics	SPZB250	2.4GHz	-92dBm	3dBm
	SPZB260		-95dBm	
RF Monolithics	ZMN2405/30	2.4GHz	-92 dBm	0dBm
	ZMN2430A		-90dBm	-2dBm
	ZMN2405HP			17dBm
	ZMN2405HPA		-95dBm	15dBm
	ZMN2430HP			17dBm
	ZMN2430HPA			15dBm
Crossbow Technology	XM2110CA	2.4GHz	-101dBm	3dBm
	MICAz (MPR2400)		-94dBm	0dBm
MeshNetics	MNZB-24-B0	2.4GHz	-101dBm	3dBm
	MNZB-24-A2			
	MNZB-A24-U0			
	MNZB-A24-UFL	-104dBm	20dBm	
	MNZB-900-B0	700-900MHz	-110 dBm	11dBm
Digi International Inc.	Series 1 XBee manual	2.4GHz	-92dBm	0dBm
	Series 1 XBee- PRO manual		-100dBm	18dBm, 10dBm
	Series 2 XBee ZB manual		-96dBm	3dBm
	Series 2 XBee-PRO ZB manual		-102dBm	17dBm, 10dBm
Microchip	MRF24J40MC	2.4GHz	-108dBm	19dBm
	MRF24J40MB		-102dBm	20dBm
	MRF24J40MA		-94dBm	0dBm

Appendix 4

Embedded webserver application

Main Demo.c

```

#define THIS_IS_STACK_APPLICATION
//-----
//----- Includes -----
//-----
#include "TCPIP Stack/TCPIP.h"
#include "MainDemo.h"
#include "HardwareProfile.h"
#include "adriaan.h"
#include "uart2.h"
#include "stdio.h"
//-----
//----- Variables -----
//-----
BYTE machineDesc[33];           // Machine descript string
BYTE NrOfNodes
DWORD displayTimeout;         // When SM_DISPLAY_WAIT times out
BYTE AN0String[8];
BYTE UARTStatus;
static volatile BYTE *ptr;
const char ok[] = "OK";
BYTE Disp8[3];
long int templongint;
char rxBuffer[30],
txBuffer[30],EcoBuffer[30],RemoteTestIO,random;
ini,OutputStatus, InputStatus, Address, ATStatus, Templnt, ATCommsInlt,
ATCommsSuccess,
NumberOfNodes = 3;
char UART2RXBuffer[50],ii;
struct
{
//Main structure containing node info
char UpdateStatus;
char Nr[3];
char Name[16];
char Address[3];
char Role[10];
char Type[10];
char Priority[5];
char State[4];
char Activity;
char Alarm[4];
char AlarmTriggered;
}Node[3];
struct{
char sec[3];
char min[3];
char hour[3];
}Time[3];

```

```

struct{
    char    sec[3];
    char    min[3];
    char    hour[3];
}SystemTime;
struct Status {
    unsigned int LocalChange:1;
    unsigned int NewDateTime:1;
    unsigned int NewNodeData:1;
    unsigned int AlarmTriggered:1;
    unsigned int Priority:1;
    unsigned int RemoteChange:1;
    unsigned int NewUARTData:1;
};
struct Status Statusbits;
static enum
{
    SM_IDLE = 0u,
    SM_DEBOUNCE_DOWN,
    SM_ADD_COIN,
    SM_TRY_VEND,
    SM_PREV,
    SM_NEXT,
    SM_DISPLAY_WAIT,
    SM_RELEASE_WAIT,
    SM_DEBOUNCE_UP
} smVend = SM_DEBOUNCE_DOWN;           // Application state machine

//-----
//----- Defines -----
//-----
APP_CONFIG AppConfig;

//-----
//----- Prototypes -----
//-----
// Vending Machine Function Prototypes
void WriteLCDMenu(void);
static void WritePriceLCD(BYTE price, BYTE position);
// Private helper functions.
// These may or may not be present in all applications.
static void InitAppConfig(void);
static void InitializeBoard(void);
static void ProcessIO(void);

//-----
//----- Interrupt service routines -----
//-----
void __attribute__((interrupt, no_auto_psv)) _U2RXInterrupt(void) {
    UART2RXBuffer[iii] = U2RXREG;

    if(UART2RXBuffer[iii] == 0x0D)
    {
        Statusbits.NewUARTData = 1;
    }
}

```

```

        ii=0;
    }
    else
    {
        ii++;
        IFS1bits.U2RXIF = 0;
    }
}
void __attribute__((interrupt, no_auto_psv)) _U2TXInterrupt(void) {
    IFS1bits.U2TXIF = 0;
}

//-----
//----- Main Program starts here -----
//-----

int main(void)
{
    static TICK t = 0;
    static DWORD dwLastIP = 0;
    // Initialize and display the stack version on the LCD
    InitUART2(); // Initialize UART2 for 9600,8,N,1 TX/RX
    LCDInit();
    strcpypgm2ram((char*)LCDText, "Zigbit WebServer ");
    LCDUpdate();
    TickInit();
    MPFSInit();

    // Initialize Stack and application related NV variables into AppConfig.
    InitAppConfig();
    InitializeBoard();

    XEEBeginWrite(0x0000);
    XEEWrite(0xFF);
    XEEEndWrite();

    StackInit();

    ii=0;
    while(1)
    {
        // Blink LED0 (right most one) every second.
        if(TickGet() - t >= TICK_SECOND/2ul){t = TickGet();LED0_IO ^= 1;}

        StackTask();
        StackApplications();

        if(Statusbits.LocalChange == 1)
        {
            SendUartData();
            Statusbits.LocalChange = 0;
        }
        if(Statusbits.NewUARTData == 1)
        {

```

```

        ProcessUARTData();
        Statusbits.NewUARTData = 0;
        PORTA++;
    }
}

//-----
//----- Functions -----
//-----

void SendUartData()
{
    //Send node info
    if(Statusbits.NewNodeData == 1)
    {
        UART2PutChar(1);
        for(i=0;i<NumberOfNodes;i++)
        {
            if(strcmp(Node[i].Type,"Water")==0){UART2PutChar(1);}
            else if(strcmp(Node[i].Type,"Movement")==0){UART2PutChar(2);}
            else if(strcmp(Node[i].Type,"Input")==0){UART2PutChar(3);}
            else if(strcmp(Node[i].Type,"Output")==0){UART2PutChar(4);}
            else if(strcmp(Node[i].Type,"Analogue")==0){UART2PutChar(5);}
            else(UART2PutChar(0));

            if(strcmp(Node[i].Priority,"Low")==0){UART2PutChar(1);}
            else if(strcmp(Node[i].Priority,"Med")==0){UART2PutChar(2);}
            else if(strcmp(Node[i].Priority,"High")==0){UART2PutChar(3);}
            else(UART2PutChar(0));

            if(strcmp(Node[i].Alarm,"Yes")==0){UART2PutChar(1);}
            else if(strcmp(Node[i].Alarm,"No")==0){UART2PutChar(2);}
            else(UART2PutChar(0));
            Statusbits.NewNodeData = 0;
        }
    }
    if(Statusbits.NewDateTime == 1)
    {
        //      Send time info
        UART2PutChar(2);
        UART2PutChar(SystemTime.sec[0]);
        UART2PutChar(SystemTime.sec[1]);
        UART2PutChar(SystemTime.min[0]);
        UART2PutChar(SystemTime.min[1]);
        UART2PutChar(SystemTime.hour[0]);
        UART2PutChar(SystemTime.hour[1]);
        Statusbits.NewDateTime = 0;
    }
    //      Send termination character
    UART2PutChar(0x0D);
}
void ProcessUARTData(void)

```

```

{
    for(i=0;i<NumberOfNodes;i++)
    {
        Node[i].State[0] = UART2RXBuffer[(i*12)+1];
        Node[i].State[1] = UART2RXBuffer[(i*12)+2];
        Node[i].State[2] = UART2RXBuffer[(i*12)+3];

        if(UART2RXBuffer[(i*12)+4]==0){}
        else if(UART2RXBuffer[(i*12)+4]== 1 ){ strcpypgm2ram((char*)Node[i].Type,
(ROM char*)"Water");}
        else if(UART2RXBuffer[(i*12)+4]== 2 ){ strcpypgm2ram((char*)Node[i].Type,
(ROM char*)"Movement");}
        else if(UART2RXBuffer[(i*12)+4]== 3 ){ strcpypgm2ram((char*)Node[i].Type,
(ROM char*)"Input");}
        else if(UART2RXBuffer[(i*12)+4]== 4 ){ strcpypgm2ram((char*)Node[i].Type,
(ROM char*)"Output");}
        else if(UART2RXBuffer[(i*12)+4]== 5 ){ strcpypgm2ram((char*)Node[i].Type,
(ROM char*)"Analoque");}

        if(UART2RXBuffer[(i*12)+5]== 1 ){ strcpypgm2ram((char*)Node[i].Priority,
(ROM char*)"Low");}
        else if(UART2RXBuffer[(i*12)+5]== 1 ){
strcpypgm2ram((char*)Node[i].Priority, (ROM char*)"Med");}
        else if(UART2RXBuffer[(i*12)+5]== 3 ){
strcpypgm2ram((char*)Node[i].Priority, (ROM char*)"High");}

        if(UART2RXBuffer[(i*12)+6]== 0 ){ strcpypgm2ram((char*)Node[i].Alarm,
(ROM char*)"No");}
        else if(UART2RXBuffer[(i*12)+6]== 1 ){ strcpypgm2ram((char*)Node[i].Alarm,
(ROM char*)"Yes");}

        Time[i].sec[0]          = UART2RXBuffer[(i*12)+7];
        Time[i].sec[1]          = UART2RXBuffer[(i*12)+8];
        Time[i].min[0]          = UART2RXBuffer[(i*12)+9];
        Time[i].min[1]          = UART2RXBuffer[(i*12)+10];
        Time[i].hour[0]         = UART2RXBuffer[(i*12)+11];
        Time[i].hour[1]         = UART2RXBuffer[(i*12)+12];
    }
    Statusbits.NewUARTData = 0;
    ii=0;
}

static void InitializeBoard(void)
{
    CLKDIVbits.RCDIV = 0;          // Set 1:1 8MHz FRC postscalar

    XEEInit();
    LED0_TRIS = 0;
    LED0_IO = 1;
    TRISA = 0;
}

void UART2PutChar( char ch )
{

```

```

    U2TXREG = ch;
    while(U2STAbits.TRMT == 0);
}

static ROM BYTE SerializedMACAddress[6] = {MY_DEFAULT_MAC_BYTE1,
MY_DEFAULT_MAC_BYTE2, MY_DEFAULT_MAC_BYTE3, MY_DEFAULT_MAC_BYTE4,
MY_DEFAULT_MAC_BYTE5, MY_DEFAULT_MAC_BYTE6};
#pragma romdata
static void InitAppConfig(void)
{
    AppConfig.Flags.bIsDHCPEnabled = TRUE;
    AppConfig.Flags.bInConfigMode = TRUE;
    memcpy_pgm2ram((void*)&AppConfig.MyMACAddr, (ROM
void*)SerializedMACAddress, sizeof(AppConfig.MyMACAddr));
//    {
//        _prog_addressT MACAddressAddress;
//        MACAddressAddress.next = 0x157F8;
//        _memcpy_p2d24((char*)&AppConfig.MyMACAddr, MACAddressAddress,
sizeof(AppConfig.MyMACAddr));
//    }
    AppConfig.MyIPAddr.Val = MY_DEFAULT_IP_ADDR_BYTE1 |
MY_DEFAULT_IP_ADDR_BYTE2<<8ul | MY_DEFAULT_IP_ADDR_BYTE3<<16ul |
MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
    AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
    AppConfig.MyMask.Val = MY_DEFAULT_MASK_BYTE1 |
MY_DEFAULT_MASK_BYTE2<<8ul | MY_DEFAULT_MASK_BYTE3<<16ul |
MY_DEFAULT_MASK_BYTE4<<24ul;
    AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;
    AppConfig.MyGateway.Val = MY_DEFAULT_GATE_BYTE1 |
MY_DEFAULT_GATE_BYTE2<<8ul | MY_DEFAULT_GATE_BYTE3<<16ul |
MY_DEFAULT_GATE_BYTE4<<24ul;
    AppConfig.PrimaryDNSServer.Val = MY_DEFAULT_PRIMARY_DNS_BYTE1 |
MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul |
MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul |
MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
    AppConfig.SecondaryDNSServer.Val = MY_DEFAULT_SECONDARY_DNS_BYTE1
| MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul |
MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul |
MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;

// SNMP Community String configuration
#if defined(STACK_USE_SNMP_SERVER)
{
    BYTE i;
    static ROM char * ROM cReadCommunities[] =
SNMP_READ_COMMUNITIES;
    static ROM char * ROM cWriteCommunities[] =
SNMP_WRITE_COMMUNITIES;
    ROM char * strCommunity;

    for(i = 0; i < SNMP_MAX_COMMUNITY_SUPPORT; i++)
    {
        // Get a pointer to the next community string
        strCommunity = cReadCommunities[i];
        if(i >= sizeof(cReadCommunities)/sizeof(cReadCommunities[0]))

```

```

        strCommunity = "";

        // Ensure we don't buffer overflow. If your code gets stuck here,
        // it means your SNMP_COMMUNITY_MAX_LEN definition in
TCPIPConfig.h
        // is either too small or one of your community string lengths
        // (SNMP_READ_COMMUNITIES) are too large. Fix either.
        if(strlenpgm(strCommunity) >= sizeof(AppConfig.readCommunity[0]))
            while(1);

        // Copy string into AppConfig
        strcpypgm2ram((char*)AppConfig.readCommunity[i], strCommunity);

        // Get a pointer to the next community string
        strCommunity = cWriteCommunities[i];
        if(i >= sizeof(cWriteCommunities)/sizeof(cWriteCommunities[0]))
            strCommunity = "";

        // Ensure we don't buffer overflow. If your code gets stuck here,
        // it means your SNMP_COMMUNITY_MAX_LEN definition in
TCPIPConfig.h
        // is either too small or one of your community string lengths
        // (SNMP_WRITE_COMMUNITIES) are too large. Fix either.
        if(strlenpgm(strCommunity) >= sizeof(AppConfig.writeCommunity[0]))
            while(1);

        // Copy string into AppConfig
        strcpypgm2ram((char*)AppConfig.writeCommunity[i], strCommunity);
    }
}
#endif

strcpypgm2ram((char*)machineDesc, (ROM char*)"Adriaan dev board");
machineDesc[18] = '\0';

// Load the default NetBIOS Host Name
memcpypgm2ram(AppConfig.NetBIOSName, (ROM
void*)MY_DEFAULT_HOST_NAME, 16);
FormatNetBIOSName(AppConfig.NetBIOSName);
}

void InitUART2(void)
{
// Initialize UART2 for 9600,8,N,1 TX/RX
U2MODEbits.UARTEN = 0; // Bit15 TX, RX DISABLED, ENABLE at end of func
U2MODEbits.USIDL = 0; // Bit13 Continue in Idle
U2MODEbits.IREN = 0; // Bit12 No IR translation
U2MODEbits.RTSMD = 0; // Bit11 Simplex Mode
U2MODEbits.UEN = 0; // Bits8,9 TX,RX enabled, CTS,RTS not
U2MODEbits.WAKE = 0; // Bit7 No Wake up (since we don't sleep here)
U2MODEbits.LPBACK = 0; // Bit6 No Loop Back
U2MODEbits.ABAUD = 0; // Bit5 No Autobaud (would require sending '55')
U2MODEbits.RXINV = 0; // Bit4 IdleState = 1
U2MODEbits.BRGH = 0; // Bit3 16 clocks per bit period
U2MODEbits.PDSEL = 0; // Bits1,2 8bit, No Parity

```

```

U2MODEbits.STSEL = 0;      // Bit0 One Stop Bit

U2BRG = BAUDRATEREG2;    // baud rate

// Load all values in for U1STA SFR
U2STAbits.UTXISEL1 = 0;    //Bit15 Int when Char is transferred (1/2 config!)
U2STAbits.UTXINV = 0;    //Bit14 N/A, IRDA config
U2STAbits.UTXISEL0 = 0;    //Bit13 Other half of Bit15
U2STAbits.UTXBRK = 0;    //Bit11 Disabled
U2STAbits.UTXEN = 0;    //Bit10 TX pins controlled by periph
U2STAbits.UTXBF = 0;    //Bit9 *Read Only Bit*
U2STAbits.TRMT = 0;    //Bit8 *Read Only bit*
U2STAbits.URXISEL = 0;    //Bits6,7 Int. on character recieved
U2STAbits.ADDEN = 0;    //Bit5 Address Detect Disabled

IFS1bits.U2TXIF = 0;    // Clear the Transmit Interrupt Flag
IEC1bits.U2TXIE = 0;    // Disable Transmit Interrupts
IFS1bits.U2RXIF = 0;    // Clear the Recieve Interrupt Flag
IEC1bits.U2RXIE = 1;    // Enable Recieve Interrupts

U2MODEbits.UARTEN = 1;    // And turn the peripheral on

U2STAbits.UTXEN = 1;
}

```

CustomHTTPApp.c

```

#define __CUSTOMHTTPAPP_C
#include "TCPIP Stack/TCPIP.h"
#include "HardwareProfile.h"
#if defined(STACK_USE_HTTP2_SERVER)
extern HTTP_CONN curHTTP;
extern HTTP_STUB httpStubs[MAX_HTTP_CONNECTIONS];
extern BYTE curHTTPID;
// Access to MainDemo.c functions and variables
#if defined(MPFS_USE_EEPROM)
extern void SaveAppConfig(void);
#endif
void WriteLCDMenu(void);
extern APP_CONFIG AppConfig;
#include "adriaan.h"
extern DEVICE Device[MAX_DEVICES];
extern BYTE machineDesc[33];

extern struct
{
//Main structure containing node info
char      UpdateStatus;
char      Nr[3];
char      Name[16];
char      Address[3];
char      Role[10];
char      Type[10];

```



```
        char        Priority[5];
        char        State[4];
        char        Activity;
        char        Alarm[4];
        char        AlarmTriggered;
}Node[3];

extern struct Status {
    unsigned int LocalChange:1;
    unsigned int NewDateTime:1;
    unsigned int NewNodeData:1;
    unsigned int AlarmTriggered:1;
    unsigned int Priority:1;
    unsigned int RemoteChange:1;
    unsigned int NewUARTData:1;
};
extern struct Status Statusbits;
extern struct{
    char    sec[3];
    char    min[3];
    char    hour[3];
}Time[3];

extern struct{
    char    sec[3];
    char    min[3];
    char    hour[3];
}SystemTime;

#if defined(HTTP_USE_AUTHENTICATION)
BYTE HTTPNeedsAuth(BYTE* cFile)
{
    // No authentication is defined yet.

    return 0x80;
}
#endif
#if defined(HTTP_USE_AUTHENTICATION)
BYTE HTTPCheckAuth(BYTE* cUser, BYTE* cPass)
{
    // No authentication is defined yet.

    return 0x80;
}
#endif
HTTP_IO_RESULT HTTPExecuteGet(void)
{
    BYTE *ptr, name[20];
    char  TempChar[10],Temp;

    // Load the file name
    // Make sure BYTE filename[] above is large enough for your longest name
    MPFSGetFilename(curHTTP.file, name, 20);
```

```

// Make sure it's the correct page
if(strcmp(pgm2ram((char*)name, (ROM char*)"index.htm") != 0)
    return HTTP_IO_DONE;

//Check What Date The page submitted

ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"EditNode");
if(ptr)
{
    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"Nr");
    if(ptr)

        strcpy(pgm2ram((char*)TempChar, (ROM char*)ptr);
        TempI = atoi(TempChar);
        strcpy(pgm2ram((char*)Node[TempI].Nr, (ROM char*)ptr);
        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"Discription");
        if(ptr && *ptr!=0 )
            strcpy(pgm2ram((char*)Node[TempI].Name, (ROM char*)ptr);

        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"Type");
        if(ptr)
            strcpy(pgm2ram((char*)Node[TempI].Type, (ROM char*)ptr);

        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"Priority");
        if(ptr)
            strcpy(pgm2ram((char*)Node[TempI].Priority, (ROM char*)ptr);

        ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"Alarm");
        if(ptr)
            strcpy(pgm2ram((char*)Node[TempI].Alarm, (ROM char*)ptr);

        Statusbits.NewNodeData = 1; //Tell main loop te send new node info via uart

Statusbits.LocalChange = 1;
}

//Date and time section
ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"DateTime");
if(ptr)
{

    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"hour");
    if(ptr)
        strcpy(pgm2ram((char*)SystemTime.hour, (ROM char*)ptr);

    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"min");
    if(ptr)
        strcpy(pgm2ram((char*)SystemTime.min, (ROM char*)ptr);

    ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE *)"sec");
    if(ptr)
        strcpy(pgm2ram((char*)SystemTime.sec, (ROM char*)ptr);
    Statusbits.NewDateTime = 1; //Tell main loop te send new node info via uart

    Statusbits.LocalChange = 1;
}

```

```
        // Indicate that we're finished
        return HTTP_IO_DONE;
        while(1);
    }
    #if defined(HTTP_USE_POST)

    HTTP_IO_RESULT HTTPExecutePost(void)
    {
    }

    #endif //(use_post)

    void HTTPPrint_machineDesc(void)
    {
        // Make sure enough space exists
        if(strlen((char*)machineDesc) > TCPIsPutReady(sktHTTP))
        {
            curHTTP.callbackPos = 0x01;
            return;
        }

        // Write the string and make sure we continue
        TCPPutString(sktHTTP, machineDesc);
        curHTTP.callbackPos = 0x00;
        return;
    }

    void HTTPPrint_Sec(WORD NodeNr)
    {
        if(strlen((char*)Time[NodeNr].sec) > TCPIsPutReady(sktHTTP))
        {
            curHTTP.callbackPos = 0x01;
            return;
        }

        TCPPutString(sktHTTP, Time[NodeNr].sec);
        curHTTP.callbackPos = 0x00;
        return;
    }

    void HTTPPrint_Min(WORD NodeNr)
    {
        if(strlen((char*)Time[NodeNr].min) > TCPIsPutReady(sktHTTP))
        {
            curHTTP.callbackPos = 0x01;
            return;
        }

        TCPPutString(sktHTTP, Time[NodeNr].min);
        curHTTP.callbackPos = 0x00;
        return;
    }

    void HTTPPrint_Hour(WORD NodeNr)
```

```
{
    if(strlen((char*)Time[NodeNr].hour) > TCPIsPutReady(sktHTTP))
    {
        curHTTP.callbackPos = 0x01;
        return;
    }

    TCPPutString(sktHTTP, Time[NodeNr].hour);
    curHTTP.callbackPos = 0x00;
    return;
}
void HTTPPrint_version(void)
{
    TCPPutROMString(sktHTTP,(ROM void*)VERSION);
    return;
}

void HTTPPrint_builddate(void)
{
    TCPPutROMString(sktHTTP,(ROM void*)__DATE__ " " __TIME__);
    return;
}

void HTTPPrint_hostname(void)
{
    TCPPutString(sktHTTP, AppConfig.NetBIOSName);
    return;
}

void HTTPPrint_nr(WORD NodeNr)
{
    TCPPutString(sktHTTP, Node[NodeNr].Nr);
    return;
}

void HTTPPrint_name(WORD NodeNr)
{
    if(strlen((char*)Node[NodeNr].Name) > TCPIsPutReady(sktHTTP))
    {
        curHTTP.callbackPos = 0x01;
        return;
    }

    // Write the string and make sure we continue
    TCPPutString(sktHTTP, Node[NodeNr].Name);
    curHTTP.callbackPos = 0x00;
    return;
}
void HTTPPrint_Type(WORD NodeNr)
{
    TCPPutString(sktHTTP, Node[NodeNr].Type);
    return;
}
void HTTPPrint_Wrole(WORD NodeNr)
{

```

```

        TCPPutString(sktHTTP, Node[NodeNr].Role);
        return;
    }
    void HTTPPrint_Priority(WORD NodeNr)
    {
        TCPPutString(sktHTTP, Node[NodeNr].Priority);
        return;
    }
    void HTTPPrint_State(WORD NodeNr)
    {
        if(strlen((char*)Node[NodeNr].State) > TCPIsPutReady(sktHTTP))
        {
            curHTTP.callbackPos = 0x01;
            return;
        }
        // Write the string and make sure we continue
        TCPPutString(sktHTTP, Node[NodeNr].State);
        curHTTP.callbackPos = 0x00;
        return;
    }
}

#endif

```

HTTPPrint.h

```

/*****
 * HTTPPrint.h
 * Provides callback headers and resolution for user's custom
 * HTTP Application.
 *
 * This file is automatically generated by the MPFS Utility
 * ALL MODIFICATIONS WILL BE OVERWRITTEN BY THE MPFS GENERATOR
 *****/

#ifndef __HTTPPRINT_H
#define __HTTPPRINT_H
#include "TCPIP Stack/TCPIP.h"
#if defined(STACK_USE_HTTP2_SERVER)
extern HTTP_STUB httpStubs[MAX_HTTP_CONNECTIONS];
extern BYTE curHTTPIPID;
void HTTPPrint(DWORD callbackID);
void HTTPPrint_nr(WORD);
void HTTPPrint_name(WORD);
void HTTPPrint_Type(WORD);
void HTTPPrint_Priority(WORD);
void HTTPPrint_Hour(WORD);
void HTTPPrint_Min(WORD);
void HTTPPrint_Sec(WORD);
void HTTPPrint_State(WORD);
void HTTPPrint(DWORD callbackID)
{
    switch(callbackID)
    {
        case 0x0000001d:
            HTTPPrint_nr(0);
            break;
        case 0x0000001e:

```

```
        HTTPPrint_name(0);
        break;
case 0x0000001f:
        HTTPPrint_Type(0);
        break;
case 0x00000021:
        HTTPPrint_Priority(0);
        break;
case 0x00000022:
        HTTPPrint_Hour(0);
        break;
case 0x00000023:
        HTTPPrint_Min(0);
        break;
case 0x00000024:
        HTTPPrint_Sec(0);
        break;
case 0x00000025:
        HTTPPrint_State(0);
        break;
case 0x00000026:
        HTTPPrint_nr(1);
        break;
case 0x00000027:
        HTTPPrint_name(1);
        break;
case 0x00000028:
        HTTPPrint_Type(1);
        break;
case 0x0000002a:
        HTTPPrint_Priority(1);
        break;
case 0x0000002b:
        HTTPPrint_Hour(1);
        break;
case 0x0000002c:
        HTTPPrint_Min(1);
        break;
case 0x0000002d:
        HTTPPrint_Sec(1);
        break;
case 0x0000002e:
        HTTPPrint_State(1);
        break;
case 0x0000002f:
        HTTPPrint_nr(2);
        break;
case 0x00000030:
        HTTPPrint_name(2);
        break;
case 0x00000031:
        HTTPPrint_Type(2);
        break;
case 0x00000033:
        HTTPPrint_Priority(2);
```

```
        break;
    case 0x00000034:
        HTTPPrint_Hour(2);
        break;
    case 0x00000035:
        HTTPPrint_Min(2);
        break;
    case 0x00000036:
        HTTPPrint_Sec(2);
        break;
    case 0x00000037:
        HTTPPrint_State(2);
        break;
    default:
        // Output notification for undefined values
        TCPputROMArray(sktHTTP, (ROM BYTE*)"!DEF", 4);
    }

    return;
}
void HTTPPrint_(void)
{
    TCPput(sktHTTP, '~');
    return;
}
#endif
#endif
```

Appendix 5

Sensor network server embedded code

```
//-----
//----- Includes -----
//-----
#include <p24fxxx.h>
#include <libpic30.h>
#include "system.h"
#include "lcdPmp.h"
#include "uart2.h"
#include "stdio.h"
#include "Compiler.h"
#include <libq.h>
#include "spieeprom.h"
#include "spi2.h"

//-----
//----- Congig bits -----
//-----
_CONFIG2(IESO_OFF & FNOSC_PRIPLL & FCKSM_CSDCMD & OSCIOFNC_OFF &
POSCMOD_HS)
_CONFIG1(JTAGEN_OFF & GCP_OFF & GWRP_OFF & COE_OFF & ICS_PGx2 &
FWDTEN_OFF)

//-----
//----- Variables -----
//-----
//Main structure containing node info
struct Node{
    char        UpdateStatus;
    char        Nr[3];
    char        Address[3];
    char        Role;
    char        Type;
    char        Priority;
    char        State[4];
    char        Activity;
    char        AlarmSet;
    char        AlarmTriggered;
    char        Status;
}Node[3];
struct Status {
    unsigned int LocalChange:1;
    unsigned int AlarmTriggered:1;
    unsigned int Priority:1;
    unsigned int RemoteChange:1;
    unsigned int NewUARTData:1;
    unsigned int FirstADCRun:1;
};
};
```



```

struct Status Statusbits;
struct Error {
    unsigned int BaseStation:1;
    unsigned int Node:1;
};
struct Error Errorbits;
struct TransmitControl {
    unsigned int DataType:1;
};
struct TransmitControl TransmitControlbits;
struct{
    char    sec[3];
    char    min[3];
    char    hour[3];

}Time[3];
struct{
    char    sec[2];
    char    min[2];
    char    hour[2];
}SystemTime;

//Variable which values can be updated by the user and then used as constants
char  NumberOfNodes = 3;
char  PLow = 5, Pmed = 60, Phigh = 720;
//Random use variable
unsigned char    testchar,led1 =0,led2=0;
long int        RTCArmCount=0,i,rx;
unsigned char    rxBuffer[30], txBuffer[30], EcoBuffer[30];
unsigned char    *ptr, ii, Result,y;
unsigned char    UART1RXBuffer[50];
unsigned int txData, rxData;

//-----
//----- Defines -----
//-----
#define DeviceA PORTDbits.RD6
#define LedA PORTAbits.RA4
#define DeviceB PORTDbits.RD7
#define LedB PORTAbits.RA5
#define DeviceC PORTDbits.RD13
#define LedC PORTAbits.RA6
#define DeviceD PORTDbits.RD5
#define LedD PORTAbits.RA7
char LCDInitBuffer[16] = "Zigbit Uart v.1 ";
#define TRUE 1
#define FALSE 0

//-----
//----- Prototypes -----
//-----
void LocalLCDInit(void);
void GetNodeStatus(void);

```

```

char ReadRemotelIO(int Address, int IO1, int IO2);
void InitAT(void);
void ReadSensorData(void);
void ProcessUARTData(void);
void WriteRemotelIO(char Address, char Command, char Output, char state);
void UpdateSystemDateTime(void);
void RTCCInit(void);
void SendSensorData(void);

//-----
//----- Interrupt routines -----
//-----
void __attribute__((interrupt,no_auto_psv)) _RTCCInterrupt (void) {

    RTCAAlarmCount++;
    PORTAbits.RA0 ^= 1;
    ALCFGRPTbits.ALRMEN = 1;
    IFS3bits.RTCIF = 0;
}
void __attribute__((interrupt, no_auto_psv)) _U2RXInterrupt(void)
{
    IFS1bits.U2RXIF = 0;
}
void __attribute__((interrupt, no_auto_psv)) _U2TXInterrupt(void)
{
    IFS1bits.U2TXIF = 0;
}
void __attribute__((interrupt, no_auto_psv)) _U1RXInterrupt(void) {

    UART1RXBuffer[ij] = U1RXREG;

    if(UART1RXBuffer[ij] == 0x0D)
    {
        Statusbits.NewUARTData = 1;
        ii=0;
        IFS0bits.U1RXIF = 0;
    }
    else
    {
        ii++;
        IFS0bits.U1RXIF = 0;
    }
}
void __attribute__((interrupt, no_auto_psv)) _U1TXInterrupt(void) {

    IFS0bits.U1TXIF = 0;
}

//-----
//----- Main -----
//-----
int main (void)
{
    //App inits

```

```
TRISA = 0x0080;;
TRISD = 0xFFFF;
PORTAbits.RA3 = 1;
RCONbits.SWDTEN = 0;// Disable Watch Dog Timer
InitUART2();
SPI2INTInit();
EEPROMInit();
RTCCInit();
//Node[1].Type = 1;
//Node[2].Type = 5;
if(PORTDbits.RD7 == 0)
{ClearEEPROM();}
ReadEEPROMData();
SendSensorData();
Node[0].Type = 0;
InitAT();
//Main app loop
Statusbits.LocalChange == 1;
while(1)
{
    if(RTCAlarmCount>=5) //Board 1
    {
        ReadSensorData();
        PORTAbits.RA1 ^= 1;
        RTCAlarmCount=0;
    }

    if(Statusbits.LocalChange == 1)
    {
        SendSensorData();//Transmit Data to Webapp
        Statusbits.LocalChange = 0;
        SaveEEPROMData();
        PORTAbits.RA2 ^= 1;
    }

    if(Statusbits.RemoteChange == 1)
    {
        UpdateNodes();
    }

    if(Statusbits.AlarmTriggered == 1)
    {
        //Trigger alarm
    }

    if(Statusbits.NewUARTData == 1)
    {
        ProcessUARTData();
        Statusbits.NewUARTData = 0;
        SaveEEPROMData();
        // InitUART2();
    }

    if(PORTDbits.RD13 == 0)
    {
        while(PORTDbits.RD13 == 0)
        {
            PORTA = 0xFF;
        }
    }
}
```

```

        PORTA = 0;
    }
    if(PORTDbits.RD7 == 0)
    {ClearEEPROM();}
}

//-----
//----- Functions -----
//-----

void ProcessUARTData(void)
{
    if(UART1RXBuffer[0]==1)
    {
        for(i=0;i<NumberOfNodes;i++)
        {
            Node[i].Type      = UART1RXBuffer[(i*3)+1];
            Node[i].Priority   = UART1RXBuffer[(i*3)+2];
            Node[i].AlarmSet   = UART1RXBuffer[(i*3)+3];
        }
    }
    else if(UART1RXBuffer[0]==2)
    {
        SystemTime.sec[1] = UART1RXBuffer[1];
        SystemTime.sec[0] = UART1RXBuffer[2];
        SystemTime.min[1] = UART1RXBuffer[3];
        SystemTime.min[0] = UART1RXBuffer[4];
        SystemTime.hour[1] = UART1RXBuffer[5];
        SystemTime.hour[0] = UART1RXBuffer[6];

        UpdateSystemDateTime();
    }
    ii=0;
}

void ReadSensorData(void)
{
    char    TempValue1[4],TempValue2[4];
    int     TempIntRead;
    for(y=0;y<NumberOfNodes;y++)
    {

        strcpy(TempValue1, Node[y].State);
        if(Node[y].Type == 1)//Level Sensor - Add all three float switches to get value of 0-3
        {
            TempIntRead = 0;
            TempIntRead = (ReadRemotelO(y, 3,3)-48);
            TempIntRead = TempIntRead + (ReadRemotelO(y,3, 4)-48);
            TempIntRead = TempIntRead + (ReadRemotelO(y,3, 5)-48);
        }
    }
}

```

```

        if(Errorbits.Node == 0){sprintf(Node[y].State,"%#3i", TempIntRead);}
    }
    else if(Node[y].Type == 2)//Movement
    {
        //See if sensor was trigger since last sweep
        //Reset sensor (555 timer)
        TempIntRead = ReadRemotelO(y, 3,3);

        if(TempIntRead==1)
        {
            sprintf(Node[y].State,"%#3i", TempIntRead);
            WriteRemotelO(y,2,4,3);
            __delay32(0xFFFF);
            WriteRemotelO(y,3,4,1);
            __delay32(0xFFFF);
            WriteRemotelO(y,3,4,0);
        }
        else
        if(TempIntRead==0){sprintf(Node[y].State,"%#3i", TempIntRead);}
    }

}
else if(Node[y].Type == 3)//Input
{
//See what the current state of sensor is
}
else if(Node[y].Type == 4)//Output
{
//See what the current state of sensor is
}
else if(Node[y].Type == 5)//ADC
{
    if(Statusbits.FirstADCRun == 0)
    {
        WriteRemotelO(y,2,6,3);
        __delay32(0xFFFF);
        WriteRemotelO(y,2,7,3);
        __delay32(0xFFFF);
        WriteRemotelO(y,2,8,3);
        __delay32(0xFFFF);

        WriteRemotelO(y,3,6,0);
        __delay32(0xFFFF);
        WriteRemotelO(y,3,7,1);
        __delay32(0xFFFF);
        WriteRemotelO(y,3,8,1);
        __delay32(0xFFFF);

        WriteRemotelO(y,0,0,9);
        __delay32(0xFFFF);

        Statusbits.FirstADCRun = 1;
    }

    ReadRemotelO(y,0,4);
    Node[y].State[0] = " ";
}

```

```

        Node[y].State[1] = " ";
        Node[y].State[2] = " ";
        if(rxBuffer[2] != 0x0D)
        {
            Node[y].State[0] = rxBuffer[2];
            if(rxBuffer[3] != 0x0D)
            {
                Node[y].State[1] = rxBuffer[3];
                if(rxBuffer[4] != 0x0D)
                {
                    Node[y].State[2] = rxBuffer[4];
                }
            }
        }

        //Get the latest ADC value
        //Do conversion calculation
    }
    if(strcmp(TempValue1, Node[y].State) != 0)
    {
        Statusbits.LocalChange = 1;
        GetEventTime();
    }
}

void UpdateNodes()
{
}

void SendSensorData()
{
    //    InitUART2();
    Nop();
    UART1PutChar(0);
    for(i=0;i<NumberOfNodes;i++)
    {
        UART1PutChar(Node[i].State[0]);
        UART1PutChar(Node[i].State[1]);
        UART1PutChar(Node[i].State[2]);
        UART1PutChar(Node[i].Type);
        UART1PutChar(Node[i].Priority);
        UART1PutChar(Node[i].AlarmSet);

        UART1PutChar(Time[i].sec[0]);
        UART1PutChar(Time[i].sec[1]);
        UART1PutChar(Time[i].min[0]);
        UART1PutChar(Time[i].min[1]);
        UART1PutChar(Time[i].hour[0]);
        UART1PutChar(Time[i].hour[1]);
    }
}

```

```
    }
        UART1PutChar(0x0D);
    }

void SaveEEPROMData()
{
    int EEPROMAddress = 0x0010,x;
    //Node structure
    ptr = &Node;
    for(x=0;x<=sizeof(Node);x++)
    {
        EEPROMWriteByte((*ptr), EEPROMAddress);
        ptr++;
        EEPROMAddress++;
    }

    //Time structure

    ptr = &Time;
    EEPROMAddress = 0x0020 + sizeof(Node);
    for(x=0;x<=sizeof(Time);x++)
    {
        EEPROMWriteByte((*ptr), EEPROMAddress);
        ptr++;
        EEPROMAddress++;
    }
}

void ReadEEPROMData()
{
    int EEPROMAddress = 0x0010,x;
    ptr = &Node;
    for(x=0;x<=sizeof(Node);x++)
    {
        *ptr = EEPROMReadByte(EEPROMAddress);
        ptr++;
        EEPROMAddress++;
    }

    ptr = &Time;
    EEPROMAddress = 0x0020 + sizeof(Node);
    for(x=0;x<=sizeof(Time);x++)
    {
        *ptr = EEPROMReadByte(EEPROMAddress);
        ptr++;
        EEPROMAddress++;
    }
}

void ClearEEPROM(void)
{
```

```

int EEPROMAddress = 0x0010,x;

for(x=0;x<=sizeof(Node);x++)
{
    EEPROMWriteByte((0), EEPROMAddress);
    EEPROMAddress++;
}

ptr = &Time;
EEPROMAddress = 0x0020 + sizeof(Node);
for(x=0;x<=sizeof(Time);x++)
{
    EEPROMWriteByte((0), EEPROMAddress);
    EEPROMAddress++;
}
}

void InitAT()
{
    int TempInt1;
    txBuffer[0] = 'A';
    txBuffer[1] = 'T';
    txBuffer[2] = 0x0D;
    i=0;
    while(1)
    {
        U2TXREG = txBuffer[i];
        while(U2STAbits.TRMT == 0);
        TempInt1 = RTCArmCount+2;
        while(U2STAbits.URXDA == 0 || TempInt1 == RTCArmCount);
        if(RTCArmCount >= TempInt1)
        {
            Errorbits.BaseStation = 1;
            break;
        }
        else
        {
            Errorbits.BaseStation = 0;
            EcoBuffer[i] = U2RXREG;
            if(EcoBuffer[i] == 0x0D){break;}
            i++;
        }
    }
    i=0;
    if(Errorbits.BaseStation == 0)
    {
        while(1)
        {
            TempInt1 = RTCArmCount+2;
            while(U2STAbits.URXDA == 0 || TempInt1 == RTCArmCount);
            if(RTCArmCount >= TempInt1)
            {
                Errorbits.BaseStation = 1;
                break;
            }
        }
    }
}

```



```

    }
    else
    {
        rxBuffer[i] = U2RXREG;
        if(rxBuffer[i] == 0x0A && i>4){break;}
        i++;
        Errorbits.BaseStation = 0;
        PORTAbits.RA3 = 0;
    }
}
}
}
char ReadRemotelIO(int Address, int IO1, int IO2)
{
    InitUART2();
    char TempChar1[3], TempChar2[1],TempChar3[1];
    char io,index;
    int TempInt1, w;

    //io=IO;
    index = Address;
    sprintf(TempChar1,"%#3i", Address);
    sprintf(TempChar2,"%#1i", IO1);
    sprintf(TempChar3,"%#1i", IO2);

    txBuffer[0] = 'A';
    txBuffer[1] = 'T';
    txBuffer[2] = 'R';
    txBuffer[3] = TempChar1[0];
    txBuffer[4] = TempChar1[1];
    txBuffer[5] = TempChar1[2];
    txBuffer[6]= ',';
    txBuffer[7]= '0';
    txBuffer[8]= ',';
    txBuffer[9]= 'S';
    txBuffer[10]= '1';
    txBuffer[11]= TempChar2[0];
    txBuffer[12]= TempChar3[0];
    txBuffer[13]= '?';
    txBuffer[14]= 0x0D;

    i=0;
    while(1)
    {
        U2TXREG = txBuffer[i];
        while(U2STAbits.TRMT == 0);
        TempInt1 = RTCAAlarmCount+2;
        while(U2STAbits.URXDA == 0 || TempInt1 == RTCAAlarmCount);
        if(RTCAAlarmCount >= TempInt1)
        {
            Errorbits.BaseStation = 1;
            Errorbits.Node = 1;
            w=0;

```

```

    }
    else
    {
        Errorbits.BaseStation = 0;
        while(U2STAbits.URXDA == 0);
        EcoBuffer[i] = U2RXREG;
        if(EcoBuffer[i] == 0x0D){break;}
        i++;
    }
}
i=0;

if(Errorbits.BaseStation == 0)
{
    while(1)
    {
        TempInt1 = RTCArmCount+4;
        while(U2STAbits.URXDA == 0 || TempInt1 == RTCArmCount);
        if(RTCArmCount >= TempInt1)
        {
            Errorbits.Node = 1;
            break;
        }
        else
        {
            while(U2STAbits.URXDA == 0);
            rxBuffer[i] = U2RXREG;
            if(rxBuffer[i] == 0x0A && i>4){break;}
            i++;
            Errorbits.Node = 0;
        }
    }
    for(i=0;i<5;i++){__delay32(0xFFFF);}
    U2STAbits.OERR = 0;
    i=0;
    return (rxBuffer[2]);
}

}

void WriteRemotelO(char Address, char Command, char Output, char state)
{
    char TempChar1[3], TempChar2[1], TempChar3[1], TempChar4[2];
    InitUART2();
    sprintf(TempChar1,"%#3i", Address);
    sprintf(TempChar2,"%#1i", Command);
    sprintf(TempChar3,"%#1i", Output);
    sprintf(TempChar4,"%#2i", state);
    i=0;
    txBuffer[0] = 'A';
    txBuffer[1] = 'T';
    txBuffer[2] = 'R';
    txBuffer[3] = TempChar1[0];
    txBuffer[4] = TempChar1[1];
    txBuffer[5] = TempChar1[2];

```

```

txBuffer[6]= ',';
txBuffer[7]= '0';
txBuffer[8]= ',';
txBuffer[9]= 'S';
txBuffer[10]= '1';
txBuffer[11]= TempChar2[0];
txBuffer[12]= TempChar3[0];
txBuffer[13]= '=';
txBuffer[14]= TempChar4[0];
txBuffer[15]= TempChar4[1];
txBuffer[16]= 0x0D;

i=0;
while(1)
{
    U2TXREG = txBuffer[i];
while(U2STAbits.TRMT == 0);
    while(U2STAbits.URXDA == 0);
    EcoBuffer[i] = U2RXREG;
    if(EcoBuffer[i] == 0x0D&& i>4){break;}
    i++;
}
i=0;
while(1)
{
    while(U2STAbits.URXDA==0);
    rxBuffer[i] = U2RXREG;
    if(rxBuffer[i] == 0x0A && i>4){break;}
    i++;
}
i=0;
}

```

```

void PingNode(char Address)
{
    InitUART2();
    char TempChar1[3];
    char index;
    int TempInt;
    index = Address;
    sprintf(TempChar1,"%#3i", Address);

    txBuffer[0] = 'A';
    txBuffer[1] = 'T';
    txBuffer[2] = '+';
    txBuffer[2] = 'w';
    txBuffer[2] = 'p';
    txBuffer[2] = 'i';
    txBuffer[2] = 'n';
    txBuffer[2] = 'g';
    txBuffer[2] = ' ';
    txBuffer[2] = TempChar1[0];
    txBuffer[2] = TempChar1[1];
    txBuffer[2] = TempChar1[2];
}

```

```

txBuffer[14]= 0x0D;

i=0;
while(1)
{
    U2TXREG = txBuffer[i];
    while(U2STAbits.TRMT == 0);
    TempInt = RTCArmCount+2;
    while(U2STAbits.URXDA == 0 || TempInt == RTCArmCount);
    if(U2STAbits.URXDA != 0)
    {
        Errorbits.BaseStation = 1;
        Errorbits.Node = 1;
    }
    else
    {
        Errorbits.BaseStation = 0;
        Errorbits.Node = 0;
        while(U2STAbits.URXDA == 0);
        EcoBuffer[i] = U2RXREG;
        if(EcoBuffer[i] == 0x0D){break;}
        i++;
    }
}
i=0;

if(Errorbits.BaseStation == 0)
{
    while(1)
    {
        TempInt = RTCArmCount+5;
        while(U2STAbits.URXDA == 0 || TempInt == RTCArmCount);
        if(U2STAbits.URXDA != 0)
        {
            Errorbits.Node = 1;
            break;
        }
        else
        {
            Errorbits.Node = 0;
            while(U2STAbits.URXDA == 0);
            rxBuffer[i] = U2RXREG;
            if(rxBuffer[i] == 0x0A && i>4){break;}
            i++;
        }
    }
    for(i=0;i<5;i++){__delay32(0xFFFF);}
    U2STAbits.OERR = 0;
    i=0;

    Node[index].Status = (strcmp( rxBuffer , "OK" ));
}
}
void UART2PrintString( char *str )

```

```

{
    unsigned char c;
    while(c = *str++)
    {
        UART2PutChar(c);
        if(c==0x0D){break;}
    }
}
void UART2PutChar( char ch )
{
    U2TXREG = ch;
    while(U2STAbits.TRMT == 0);
}
void UART1PrintString( char *str )
{
    unsigned char c;
    while(c = *str++)
    {
        UART1PutChar(c);
        if(c==0x0D){break;}
    }
}
void UART1PutChar( char ch )
{
    U1TXREG = ch;
    while(U1STAbits.TRMT == 0);
}
void InitUART2(void)
{
    U2MODEbits.UARTEN = 0;    // Bit15 TX, RX DISABLED, ENABLE at end of func
    U2MODEbits.USIDL = 0;    // Bit13 Continue in Idle
    U2MODEbits.IREN = 0; // Bit12 No IR translation
    U2MODEbits.RTSMD = 0;    // Bit11 Simplex Mode
    U2MODEbits.UEN = 0;      // Bits8,9 TX,RX enabled, CTS,RTS not
    U2MODEbits.WAKE = 0;    // Bit7 No Wake up (since we don't sleep here)
    U2MODEbits.LPBACK = 0;  // Bit6 No Loop Back
    U2MODEbits.ABAUD = 0;   // Bit5 No Autobaud (would require sending '55')
    U2MODEbits.RXINV = 0;   // Bit4 IdleState = 1
    U2MODEbits.BRGH = 0;    // Bit3 16 clocks per bit period
    U2MODEbits.PDSEL = 0;   // Bits1,2 8bit, No Parity
    U2MODEbits.STSEL = 0;   // Bit0 One Stop Bit
    U2BRG = BAUDRATEREG2;  // baud rate
    // Load all values in for U1STA SFR
    U2STAbits.UTXISEL1 = 0;  //Bit15 Int when Char is transferred (1/2 config!)
    U2STAbits.UTXINV = 0; //Bit14 N/A, IRDA config
    U2STAbits.UTXISEL0 = 0;  //Bit13 Other half of Bit15
    U2STAbits.UTXBRK = 0;    //Bit11 Disabled
    U2STAbits.UTXEN = 0; //Bit10 TX pins controlled by periph
    U2STAbits.UTXBF = 0; //Bit9 *Read Only Bit*
    U2STAbits.TRMT = 0;     //Bit8 *Read Only bit*
    U2STAbits.URXISEL = 0;  //Bits6,7 Int. on character recieved
    U2STAbits.ADDEN = 0; //Bit5 Address Detect Disabled
    U2STAbits.RIDLE = 0; //Bit4 *Read Only Bit*
    U2STAbits.PERR = 0;     //Bit3 *Read Only Bit*
    U2STAbits.FERR = 0;     //Bit2 *Read Only Bit*
}

```

```

U2STAbits.OERR = 0; //Bit1 *Read Only Bit*
U2STAbits.URXDA = 0; //Bit0 *Read Only Bit*
IFS1bits.U2TXIF = 0; // Clear the Transmit Interrupt Flag
IEC1bits.U2TXIE = 0; // Disable Transmit Interrupts
IFS1bits.U2RXIF = 0; // Clear the Recieve Interrupt Flag
IEC1bits.U2RXIE = 0; // Disable Recieve Interrupts
U2MODEbits.UARTEN = 1; // And turn the peripheral on
U2STAbits.UTXEN = 1;

//uart1
U1MODEbits.UARTEN = 0; // Bit15 TX, RX DISABLED, ENABLE at end of func
U1MODEbits.USIDL = 0; // Bit13 Continue in Idle
U1MODEbits.IREN = 0; // Bit12 No IR translation
U1MODEbits.RTSMD = 0; // Bit11 Simplex Mode
U1MODEbits.UEN = 0; // Bits8,9 TX,RX enabled, CTS,RTS not
U1MODEbits.WAKE = 0; // Bit7 No Wake up (since we don't sleep here)
U1MODEbits.LPBACK = 0; // Bit6 No Loop Back
U1MODEbits.ABAUD = 0; // Bit5 No Autobaud (would require sending '55')
U1MODEbits.RXINV = 0; // Bit4 IdleState = 1
U1MODEbits.BRGH = 0; // Bit3 16 clocks per bit period
U1MODEbits.PDSEL = 0; // Bits1,2 8bit, No Parity
U1MODEbits.STSEL = 0; // Bit0 One Stop Bit

U1BRG = BAUDRATEREG2; // baud rate

// Load all values in for U1STA SFR
U1STAbits.UTXISEL1 = 0; //Bit15 Int when Char is transferred (1/2 config!)
U1STAbits.UTXINV = 0; //Bit14 N/A, IRDA config
U1STAbits.UTXISEL0 = 0; //Bit13 Other half of Bit15
U1STAbits.UTXBRK = 0; //Bit11 Disabled
U1STAbits.UTXEN = 0; //Bit10 TX pins controlled by periph
U1STAbits.UTXBF = 0; //Bit9 *Read Only Bit*
U1STAbits.TRMT = 0; //Bit8 *Read Only bit*
U1STAbits.URXISEL = 0; //Bits6,7 Int. on character recieved
U1STAbits.ADDEN = 0; //Bit5 Address Detect Disabled
U1STAbits.RIDLE = 0; //Bit4 *Read Only Bit*
U1STAbits.PERR = 0; //Bit3 *Read Only Bit*
U1STAbits.FERR = 0; //Bit2 *Read Only Bit*
U1STAbits.OERR = 0; //Bit1 *Read Only Bit*
U1STAbits.URXDA = 0; //Bit0 *Read Only Bit*

IFS0bits.U1TXIF = 0; // Clear the Transmit Interrupt Flag
IEC0bits.U1TXIE = 0; // Disable Transmit Interrupts
IFS0bits.U1RXIF = 0; // Clear the Recieve Interrupt Flag
IEC0bits.U1RXIE = 1; // Enable Recieve Interrupts

U1MODEbits.UARTEN = 1; // And turn the peripheral on

U1STAbits.UTXEN = 1;

}
void ReadRemotelInfo(char Address, char Info)
{
    //info codes
    //1=wrole

```

```
//2=wpanid
InitUART2();
char TempChar1[3], TempChar2[1], Index;

sprintf(TempChar1,"%#3i", Address);

Index=Address;

txBuffer[0] = 'A';
txBuffer[1] = 'T';
txBuffer[2] = 'R';
txBuffer[3] = TempChar1[0];
txBuffer[4] = TempChar1[1];
txBuffer[5] = TempChar1[2];
txBuffer[6]= ',';
txBuffer[7]= '0';
txBuffer[8]= ',';
txBuffer[9]= '+';
if(Info==1)
{
    txBuffer[10]= 'w';
    txBuffer[11]= 'r';
    txBuffer[12]= 'o';
    txBuffer[13]= 'l';
    txBuffer[14]= 'e';
    txBuffer[15]= '?';
    txBuffer[16]= 0x0D;
}
else if(Info==2)
{
    txBuffer[10]= 'w';
    txBuffer[11]= 'p';
    txBuffer[12]= 'a';
    txBuffer[13]= 'n';
    txBuffer[14]= 'i';
    txBuffer[15]= 'd';
    txBuffer[16]= '?';
    txBuffer[17]= 0x0D;
}

i=0;

while(1)
{
    U2TXREG = txBuffer[i];
    while(U2STAbits.TRMT == 0);
    while(U2STAbits.URXDA == 0);
    EcoBuffer[i] = U2RXREG;
    if(EcoBuffer[i] == 0x0D){break;}
    i++;
}
i=0;

while(1)
{
```

```
        while(U2STAbits.URXDA==0);
        rxBuffer[i] = U2RXREG;
        if(rxBuffer[i] == 0x4B){break;}
        i++;
    }
    for(i=0;i<5;i++){__delay32(0xFFFF);}

    U2STAbits.OERR = 0;

    ptr = strstr(rxBuffer, ":");
    testchar = *(ptr+1);
}
void GetEventTime(void)
{
    int          xx,yy;
    char TempCharTime;

    RCFGCALbits.RTCPTR0 = 0;
    RCFGCALbits.RTCPTR1 = 0;
    xx = RTCVAL;

    //Seconds
    TempCharTime = (xx & 0b0000000000001111);
    yy = xx>>4;
    yy = (yy & 0b0000000000001111);
    yy = yy*10;
    TempCharTime = TempCharTime + yy;
    sprintf(Time[y].sec,"%#2i", TempCharTime);

    // Minutes
    yy = xx>>8;
    TempCharTime = (yy & 0b0000000000001111);
    yy = xx>>12;
    yy = (yy & 0b0000000000001111);
    yy = yy*10;
    TempCharTime = TempCharTime + yy;
    sprintf(Time[y].min,"%#2i", TempCharTime);

    RCFGCALbits.RTCPTR0 = 1;
    RCFGCALbits.RTCPTR1 = 0;
    xx = RTCVAL;
    // Hours
    TempCharTime = (xx & 0b0000000000001111);
    yy = xx>>4;
    yy = (yy & 0b00000000000011);
    yy = yy*10;
    TempCharTime = TempCharTime + yy;
    sprintf(Time[y].hour,"%#2i", TempCharTime);
}
```



```

void UpdateSystemDateTime(void)
{
    int SecInt, MinInt, HourInt, Templnt;
    NVMKEY = 0x55;
    NVMKEY = 0xAA;
    RCFGCALbits.RTCWREN = 1; //So that user can write data into register
    SecInt = SystemTime.sec[1]-48;
    SecInt = SecInt<<4;
    Templnt = SystemTime.sec[0]-48;
    SecInt = SecInt | Templnt;
    MinInt = SystemTime.min[1]-48;
    MinInt = MinInt<<4;
    Templnt = SystemTime.min[0]-48;
    MinInt = MinInt | Templnt;
    HourInt = SystemTime.hour[1]-48;
    HourInt = HourInt <<4;
    Templnt = SystemTime.hour[0]-48;
    HourInt = HourInt | Templnt;
    Templnt = MinInt<<8;
    Templnt = Templnt|SecInt;
    RCFGCALbits.RTCPTR0 = 0;
    RCFGCALbits.RTCPTR1 = 0;
    RTCVAL = Templnt;
    RCFGCALbits.RTCPTR0 = 1;
    RCFGCALbits.RTCPTR1 = 0;
    RTCVAL = HourInt;
    RCFGCALbits.RTCWREN = 0; //So that user can write data into register
}

void RTCCInit(void)
{
    // Enables the LP OSC for RTCC operation
    asm("mov #OSCCON,W1");
    asm("mov.b #0x02, W0");
    asm("mov.b #0x46, W2");
    asm("mov.b #0x57, W3");
    asm("mov.b W2, [W1]");
    asm("mov.b W3, [W1]");
    asm("mov.b W0, [W1]");

    NVMKEY = 0x55;
    NVMKEY = 0xAA;
    RCFGCALbits.RTCWREN = 1; //So that user can write data into register
    RCFGCALbits.RTCEN = 1; //Enable RTCC module
    RCFGCALbits.RTCWREN = 1; //So that user can write data into register
    RCFGCALbits.RTCOE = 0; //Disable output pin
    OSCCONbits.LPOSCEN;

    RTCVAL = 0x0000;
    RCFGCALbits.RTCPTR0 = 0;
    RCFGCALbits.RTCPTR1 = 1;
    RTCVAL = 0x0000;
    RCFGCALbits.RTCPTR0 = 1;
    RCFGCALbits.RTCPTR1 = 0;
    RTCVAL = 0x0000;
}

```

```
//Set Alarm
RCFGCALbits.RTCWREN = 1;
ALCFGRPT = 0b0100010011111111;
RCFGCALbits.RTCWREN = 1;
ALRMVAL = 0;
RCFGCALbits.RTCWREN = 1;
ALCFGRPTbits.ALRMEN = 1; //Enable alarm
IEC3bits.RTCIE = 1;
}

//EOF
```

Appendix 6

AA travel cost certificate

Certificate

Online Vehicle Rates Calculator

Created from www.aa.co.za

Certificate Number	5183429
Issue Date	28 July 2012
Personal Details	
Title	
First Name	Adriaan
Last Name	Rootman
Identity Number	
Email Address	adriaanrootman@gmail.com
Cellphone	0841234567

Vehicle Details

Make
Model
Year
Registration Number
Date Purchased

AA Rates

Purchase Price (ZAR)	125001 - 150000
Annual Distance Travelled (km)	30001 - 35000
Vehicle Type	Light Commercial (Diesel)
Engine Capacity	2501 - 3000
Fuel Price Used (R/lt)	11
Average Fixed Cost (R/km)	1.24
Average Running Cost (R/km)	1.73
Total AA Rate (R/km)	2.97

Print

While every care has been taken to ensure that the AA Rate Schedules accurately reflect the average fixed and running costs of vehicles in the relevant price and engine capacity ranges, Absa Vehicle Management Solutions (Pty) Ltd. and the Automobile Association of South Africa make no express or implied warranty or guarantee in respect of the assumptions, the source data or the calculations used to formulate the published AA rates, and accept no liability whatsoever for any use to which this data is put by the user.

