Design considerations for wildlife monitoring networks

by

Jan Pieter Meijers

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (MEng) (Research) in the Faculty of Engineering at Stellenbosch University

Department of Electrical and Electronic Engineering,
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Dr. R. Wolhuter

March 2015

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 2015/02/23 .....

Abstract

Design considerations for wildlife monitoring networks

J.P. Meijers

Department of Electrical and Electronic Engineering,
Stellenbosch University,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng E&E (Research)

March 2015

Keywords: Tracking, Wildlife, VHF, Coverage

Studying wild animals in nature is a complex and tedious task for biologists and ecologists. The rugged terrain in which the animals live and hide makes it for researchers difficult to follow and observe the animals. Technology is used to facilitate studies on animals, usually in the form of radio tracking devices.

This thesis discusses the problems and shortfalls surrounding the current technology and looks at various ways of improving it. Expanding tracking collars to capture the needed data, as well as improving the radio links to continuously provide data, are seen as the most important improvements.

The proposed solution makes use of a wireless network, through which a tracking collar can have an unbroken connection with the outside world. The study focuses on the radio links themselves and topics including antennas, radio frequency bands and propagation effects of the surroundings. An existing routing protocol to use on top of the physical radio links is discussed.

Hardware for all parts of the network is designed, built and tested. Proper ways of capturing data in a power efficient way are stated and recommended. Measurements were taken with the hardware prototypes in a location comparable to where the system will ultimately be used. The results obtained proved the feasibility of such a radio network.

Radio coverage simulations were set up to predict the coverage that is to be expected by the chosen radios in the location where the tests were done. The simulation parameters were changed until the simulated results most closely matched the real life measurements. Vegetation caused an extra attenuation of around 20dB in the radio signal. Using the tuned parameters for the simulation a network can be designed and costs calculated before huge investments are made.

Uittreksel

Ontwerpskeuses vir wildwaarnemingstelsels

("Design considerations for wildlife monitoring networks")

J.P. Meijers

Departement Elektriese en Elektroniese Ingenieurswese,
Universiteit Stellenbosch,
Privaatsak X1, Matieland 7602, Suid-Afrika.

Tesis: MIng E&E (Navorsing)

Maart 2015

Trefwoorde: Wildbewaring, Wildbeskerming, Radioplekbepaling, Radiodekking

Om wilde diere in die natuur te bestudeer is vir wetenskaplikes 'n uitdagende onderneming. Die ongerepte natuur waarin die diere bly, maak dit vir die dierkundiges moeilik om die diere op te spoor en te agtervolg. Tegnologie word gewoonlik ingespan in hierdie studies en meestal word radiosporingstoestelle gebruik.

Hierdie tesis bespreek die probleme en tekortkominge in die bestaande tegnologie. Verskeie moontlike verbeteringe word bekyk. Radiosporingshalsbande wat uitgebrei word om metings te neem, en verbetering van die radioverbindings met die halsbande is twee van die belangrikste punte waarop verbeter kan word.

In die voorgestelde verbeterings word 'n radionetwerk gebruik om op 'n deurlopende basis kontak tussen die halsbande en die buitewêreld te verseker. Die studies lê klem op die radioverbindings en onderwerpe soos antennas en radiofrekwensiebande. Radioseine se voortplantingskenmerke in die betrokke omgewing word bespreek. 'n Bestaande netwerkprotokol om op die radioverbindings te implementeer word ook bepsreek.

Apparatuur vir alle dele van die netwerk is ontwerp, gebou en getoets. Metodes van datavaslegging op 'n energiedoeltreffende wyse word beskryf en aanbevelings daarvoor gemaak. Metings is geneem met die prototipe van die apparate in 'n plek wat vergelykbaar is met waar die uiteindelike stelsel gebruik behoort te word. Die resulte wat verkry is, het die nut van die radionetwerk bewys.

Simulasies van die radiodekking is opgestel om die verwagte dekkings van die betrokke radios in die veld te bepaal. Die simulasieparameters is geleidelik verander totdat die simulasieresultate met die werklike metings ooreenstem. Plantegroei het 'n ekstra verswakking van ongeveer 20dB in die radiosein veroorsaak. Die vasgestelde parameters kan gebruik word in 'n simulasie vir die ontwerp van 'n grootskaalse netwerk. Resultate verkry vanuit die simulasie kan gebruik word vir kosteberekings voordat grootskaalse beleggings gemaak word.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

Dr Riaan Wolhuter, my study leader for this project.

Prof Nathalie Mitton from the **INRIA** research institute in Lille, France for lessons in routing protocols.

Contents

List of Figures

List of Tables

List of Graphs

List of Listings

Nomenclature

Radio Abbreviations

AFSK Audio Frequency Shift Keying. A audio baseband form of frequency shift keying, allowing the use of a voice channel for digital communication.

ASK Amplitude-Shift Keying

BER Bit Error Rate

FSK Frequency-Shift Keying

GFSK Gaussian Frequency-Shift Keying

HF High Frequency. A radio frequency band betweem 3MHz and 30MHz.

IQ modulation A form of quadrature amplitude modulation.

ISM Industrial, Scientific and Medical radio band. A frequency band that can be used without a license while keeping to certain limits and regulations.

NBFM Narrow Band Frequency Modulation, usually referring to 25kHz bandwidth channels.

PSK Phase-Shift Keying

RF Radio Frequency

RSSI Received Signal Strength Indicator

RX Receive

SNR Signal-to-Noise Ratio

SSB Single-sideband modulation

TX Transmit

UHF Ultra High Frequency. A radio frequency band between 300MHz and 3GHz.

VHF Very High Frequency. A radio frequency band between 30MHz and 300MHz.

Computer and Network Abbreviations

API Application Program Interface.

APRS Automatic Packet Reporting System, a tracking system used by amateur radio operators.

BeRTOS The name of an open source Real Time Operating System meant for use on 8-bit microcontrollers.

CCN Content Centric Networking.

CPU Central Processing Unit

CRC Cyclic Redundancy Check, a type of error detection.

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

FEC Forward Error Correction

GSM Global System for Mobile Communications, the standard for second generation cellular networks.

IPv4 The fourth version of the Internet Protocol, as is still in use today.

IPv6 The sixth version of the Internet Protocol, as is started to be used today.

ISP Internet Service Provider (or In-System Programming)

KISS Keep It Simple, Stupid. A protocol commonly used between a computer and a AX.25 modem.

LAMP Linux, Apache, MySQL, PHP. A server software stack for web based services.

OS Operating System. The software that runs on a computer to control hardware on the lower level and provide a generic interface to software on the higher level.

OSI Open Systems Interconnection model, a standard model to abstract communication protocols.

PC Personal Computer

PHP A server-side scripting language designed for web development.

RAM Random-Access Memory

REST Representational State Transfer is communication specification for web services.

ROM Read-Only Memory

SD-card Secure Digital memory card

SSID Secondary Station Identifier, as used in the AX.25 protocol.

TCP The Transmission Control Protocol, as is commonly used on networks and the internet.

USB Universal Serial Bus

WAMP The same as LAMP, but running on the Windows operating system.

Electronic Circuit Abbreviations

ADC Analog-to-digital converter. A hardware device that converts continuous analog signals to discreet digital signals.

AVR Rumoured to stand for Advanced Virtual RISC or Alf and Vegard's RISC. "*The AVR is a modified Harvard architecture 8-bit RISC single chip microcontroller which was developed by Atmel in 1996.*" - Wikipedia.org.

BJT Bipolar Junction Transistor

DAC Digital-to-analog converter. Converts digital signals to an analog equivalent.

DIP Dual In-line Package, a common electronic component form factor.

EEPROM Electrically Erasable Programmable Read-Only Memory

GPIO General Purpose Input/Output, the digital IO pins on a microcontroller.

IC Integrated circuit

IO Input/Output

ISP In-System Programming (or Internet Service Provider)

i^2c Inter-Integrated Circuit, a serial communication bus.

LED Light-Emitting Diode

PCB Printed Circuit Board

PLL Phase-Locked Loop.

PWM Pulse Width Modulator

RTC Real Time Clock

SPI Serial Peripheral Interface, a serial communication bus.

UART Universal Asynchronous Receiver/Transmitter. A hardware communication standard using serial transmission of data.

WDT WatchDog Timer

Other Abbreviations

INRIA The French Institute for Research in Computer Science and Automation.

RS compoenents The name of a large electronic component store.

SRTM (Space-)Shuttle Radar Topography Mission

STC Standard Testing Conditions. The conditions at which a photovoltaic cell is rated.

§ 1. Introduction

1.1 Problem Description

> *Humans have encroached on natural areas as towns expand. With urban sprawl comes animal conflict as the latter are displaced and make for uneasy neighbours.*
>
> — CapeNature, *Understanding Baboons*

South Africa has vast areas of unspoilt nature. Mountains close to towns are inhabited by baboons, a scavenger species that is always on the lookout for easy prey and food. For the people living here it can quickly become a nightmare. Luckily baboons and apes are approachable species, making research into their social system possible. These type of studies helped considerably in our understanding of wildlife and its behaviour and gave us guidelines to what our behaviour towards it should be like.

Except for cities expanding into the wildlife's territory, the rest of the natural areas are mostly included in farms. Lesser researched species have a similar habit for discovering the easily accessible food — farm animals. Jackals and caracals are blamed for loss in livestock across the country. Farmers' approach of "killing the enemy" sometimes cause a cascade effect on the ecosystem. Not knowing the social interaction in for example a pack of jackals, a much worse effect may occur than anticipated.

Further away from urban civilisation a different problem crops up. Nature reserves are specially set up in rural areas to give shelter to wildlife. Lately, but as has always been, poaching of animals in wildlife reserves is a problem that is escalating. Due to the sheer size of these reserves, protecting individual animals is in practice almost impossible and securing entire reserves is not much easier.

Using technology to gather information where it is impractical or impossible for humans to do it, is a common practice. Building on current technology many of the known problems can be alleviated. Monitoring a pack of jackals may provide us with the insight needed to keep them away from farm animals, without upsetting the delicate balances that are at play in the wild. Monitoring individual animals suddenly became reality in the past decade with relatively easy localisation by GPS and easy communication via the mobile telephone network.

The earth as we know it is being endangered by the extreme growth in the human population. To feed all these people, farms are forced to overproduce. The consequence is that wildlife suffers. But by using the technology at our disposal, we can make a significant difference.

Getting measurements from the field is a task more difficult than normally anticipated. Using technology in this field is an old practice, but many problems still exist with the devices and their application. This document looks at some of these problems and ways in which they could possibly be solved, or at least alleviated. This research investigates the known technical problems and possible associated improvements.

1.2 Objectives

The research presented in this document identified the following objectives:

- Investigate ways to implement animal borne data acquisition in a power efficient way. Both hardware and software techniques are to be investigated.
- Find a suitable method to transport acquired data from the field, to researchers with internet access. Specifically the frequency band that should be used for wireless communication, as well as the network protocol used on the wireless links, are to be investigated.
- Radio propagation simulations do not accurately predict propagation through low level clutter which is typical for wildlife conservation areas. An approach to quantify the attenuation of a radio signal by this type of vegetation is to be investigated. Guidelines to follow for proper conservative simulations, are to be sought.
- Radio communication is heavily dependant on the antenna used on both sides of the radio link. This document will also investigate the negative effects of an animal body close to an antenna. Research into directional antennas to assist in the collection of data with a wireless network is discussed as well.

Even though the objectives for the research appear to be independent, they all play an important part in wildlife monitoring. Improving any one of these areas will hugely benefit the entire system.

1.3 Proposed Methodology

A network of stations, equipped with lower power radios and solar panels will be deployed in a typical area to be monitored. Animal borne sensors will transmit radio messages containing measured parameters. These messages will be received by the network of deployed radios and relayed to a location where an internet connection is available. With this architecture a live or semi-live feed of data from the field should be possible.

In the course of this process, the various aspects listed under the perceived objectives, will be considered. Where feasible they will be incorporated.

1.4 Summary of Outcomes

- A sensor module was designed and tested to do animal borne or field based low power measurements. The data is then transmitted via a radio link. The module was later modified to do constant sample rate logging of accelerometer data and storing the data on a SD card.
- Another hardware module was designed and built to act as a bridge or a repeater to forward the collected data through a wireless network. A base station was set up to receive data from the radio network and forward the data to the internet. A network protocol was implemented for data link management on top of the bridge, repeater and gateway nodes.

- Field measurements with the developed hardware provided data for comparison with radio coverage simulations. A number of simulations were done using different simulation parameters. This resulted in finding the best match between the real life radio coverage and the simulated coverage.

- Simulations of radio coverage at different frequencies, as well as simulations of antenna radiation patterns, when attached to the body of an animal, were done. Results indicate that a compact helical whip antenna at VHF frequencies is suitable and a quarter wavelength monopole antenna at UHF frequencies has an acceptable radiation pattern. Performance of the monopole in the UHF band is worse than the helical whip at VHF frequencies. The outcome from these simulations was that the VHF frequency band was the best choice for use in a wildlife monitoring network. The radio coverage range at lower frequencies is larger and the antennas are still compact enough.

- Further antenna design was done by developing, simulating and measuring an electronic steerable antenna. This antenna is meant for optimisation of the radio links in the wireless network.

1.5 Outline of this Document

In Chapter 2 previous research and existing technologies are analysed and discussed. In particular their limitations and how they can be improved are looked at. This will start to form a picture of where this research fits in and how a contribution can be made. Specific things that are discussed are the working of existing tracking collar designs, how communication with these collars function and how this communication can be improved.

Chapter 3 discusses how certain questions and requirements of biologists can be answered by using electronic sensors. This includes detection of motion and logging of the behaviour of the animal, as well as localisation of animals. Ways of implementing these in practice are described and especially how it can be done in an energy efficient way.

An outline of the proposed solution is given in chapter 4. The overall system design is sketched after which every hardware module in this design is discussed. A look is also given at the communication between these hardware modules.

To determine which radio frequency band will be optimal for our use, simulations are done and discussed in chapter 5. Radio coverage over a known terrain is simulated at different frequencies, determining which one will provide the best range. Antennas close to the body of an animal are affected by the animal's body. Simulations are done to provide an idea of how much this effect will be at different frequencies. Different types of antennas are analysed to determine if compact antennas are just as efficient as larger ones. At the end of the chapter a conclusion is drawn from the simulations providing the optimal frequency band to use.

Chapter 6 discusses what components, including radios and microcontrollers, are available on the market and why certain ones are chosen. Different methods of modulating and demodulating digital data using the chosen radio and microcontroller are looked at and compared. Calculation and programming of the radio's settings are examined at the end of the chapter.

In chapter 7 the discussion of the hardware and software is focussed more on the specific modules that are built and how the components are grouped per module and connected together. Every hardware sub-circuit is discussed providing the reason for its inclusion. Software specifications are given and suggestions are made for proper use and implementation. A deeper look into the network protocol, the layout of its frames and a proper addressing scheme for our use is given.

Tests are done on and with the hardware in chapter 8. This includes tests to prove the viability of the hardware for digital communication and the overall coverage that is to be expected. The results of the coverage measurements are compared to results from a terrain propagation model. The simulation is then adjusted to match the measured results. A look into the power efficiency of the hardware components and a suggestion on how they should be powered are given.

A summary of results and observations are given in chapter 9. This is done for every hardware module, followed by the network and then the entire system. Proposals on how to rectify any problems and improve the entire system are finally given.

Some specifics about the hardware design are stated in the appendices and not in the main document. This is to improve proper flow in the main document. The appendices are however to be noted as they contain important information about:

- low power and constant sample rate data logging
- electronic steerable antenna design
- radio coverage simulations

§ 2. Review of Existing Technology

There are quite a number of products on the market that provide tracking capabilities. This chapter does a review of a number of the available technologies, pointing out the shortfalls and strong points of each.

2.1 Previous research on this topic

Dirk Warnich of the Stellenbosch University did preliminary research on this topic for his masters thesis[1], titled *Tracking collar and infrastructure for leopard research*. An important conclusion he made and which we should keep in mind is that satellite communication is energy intensive. His conclusion is that a better approach would be to use a network of terrestrial VHF radio links.

Andrew Markham of the University of Cape Town researched and described a similar system in his doctoral thesis[2], titled *On a Wildlife Tracking and Telemetry System: A Wireless Network Approach*. He focused on different tag sizes for different sizes of animals, providing more or less abilities, depending on how much weight and power is available. These tags are then put into a network trying to learn the social interactions of the animals by analysing the topology of the network. Lastly the subject of using a power hungry GPS receiver in such a way that periodic sampling can be triggered by accelerometer data, rather than at fixed intervals, is touched on. The result is location samples taken at uniform distances, rather than uniformly distributed in time.

2.2 Tracking collars

Many wildlife studies have been conducted using tracking collars. Much of the research and development of this technology is commercial and not much information regarding actual product design is available in the open domain.

Radio tracking collars normally function by transmitting a radio pulse (or ping) on a certain VHF frequency. Using a directional antenna, this allows the user to track the animal down by locating from what direction the signal is the strongest. The range of this ping is not more than a few kilometres. So the user needs to have a rough idea where the animal is hiding[3].

If the radio collar is also equipped with a data logger, the data can be downloaded from the collar when the user is in close proximity of the collar, and thus the animal. The VHF ping is used to find the animal, and when the user is close to the animal a second low power UHF radio link is enabled to download the data[4].

A number of companies exist that manufacture radio collars. A few of them are:

Africa Wildlife Tracking `http://www.awt.co.za/`

Biotrack `http://www.biotrack.co.uk/`

Marshall `http://www.marshallradio.com/sporting-dog-products/sporting-dog-collars`

Telonics `http://www.telonics.com/wildlife.php`

2.3 Radio direction finding

Using directional antennas and signal strength to find the location of a radio transmitter, or collar in our case, is not a very accurate method of locating an animal. More accuracy can be achieved by using the phase of the radio signal, received by two out of phase dipoles (Time Direction of Arrival). At the point where the signals' phases cancel out, the two dipoles and the transmitter line up. Using trigonometry the animal's location can be calculated when a second direction has been acquired. This process is used in the sport called Fox Hunting, practised by amateur radio operators[5].

Because the animal might have moved during the time two directions have been found, this is an iterative process until the animal has been found. Only then the UHF downlink for data retrieval can be used. If the animal starts to run away the UHF link will fail before all data has been collected.

This process can be made easier by setting up a network of receivers listening for the ping. If these receiver stations have reconfigurable directional antennas, they can at once report the direction from where the ping is heard back to a central point where the location of the animal can be computed. A reconfigurable directional antenna similar to what will be needed for these receiver stations has been designed and tested. The design is shown in Appendix C. It was observed that the size of this antenna at VHF frequencies will be too large to be practical.

In an alternative scenario, if the VHF ping includes GPS co-ordinates, finding the animal will be easier. Having a GPS receiver constantly powered up for this, will require a huge amount of extra energy. Even in this case every animal needs to be found and their data manually downloaded. If many animals have collars, this becomes an almost impossible task.

2.4 Drones

A few studies focus on the use of drones, including quadcopters, to assist in wildlife monitoring[6]. This is also a topic being mentioned in the news lately[7], [8].

One specific study focuses on extending the existing VHF collar system with drones equipped with receivers[6, *Matthias Tobler, San Diego Zoo Institute for Conservation Research in collaboration with University of California San Diego*]. With appropriate software, the drones can then pinpoint the location of the animal, track it down and download the data on the UHF link. The idea is to allow tracking of animals in areas where tracking by foot is impractical. The goal is to locate animals to a precision of $50m$ in an area of $50km^2$.

As the height above the ground has a considerable influence on the radio link quality, a simpler solution than using drones could be to use a helium gas filled balloon carrying the receiver. The range can thus be extended and the location of the animal be tracked easier. It could even be used for the data download process using the UHF radio.

2.5 Other communication possibilities

The biggest issue with the existing tracking collars is the communication. If we can find a way to be constantly connected to the tracking collar, we know at all times the location of the animal, and can download all the data from the collar as it happens.

For constant communication with the animal three major communication possibilities exist: by satellite, by cellphone (GSM) or by long range terrestrial radio. We will now look into some of these.

2.5.1 Satellite

Using satellite radios on tracking collars has already been researched and it was found that the energy requirement is too high for use on these devices.[1, 11.3.3 Revert to Terrestrial Communications Infrastructure]. This of course does not apply where the weight of the collar is less important, as with bigger animals. In such a case batteries can be bigger to support the satellite links.

No further research into this topic was undertaken as the previous research found that more focus was needed on terrestrial VHF communication. A more versatile wildlife monitoring system should be usable on smaller animals too, and satellite communication is therefore impractical.

2.5.2 GSM and other cellphone systems

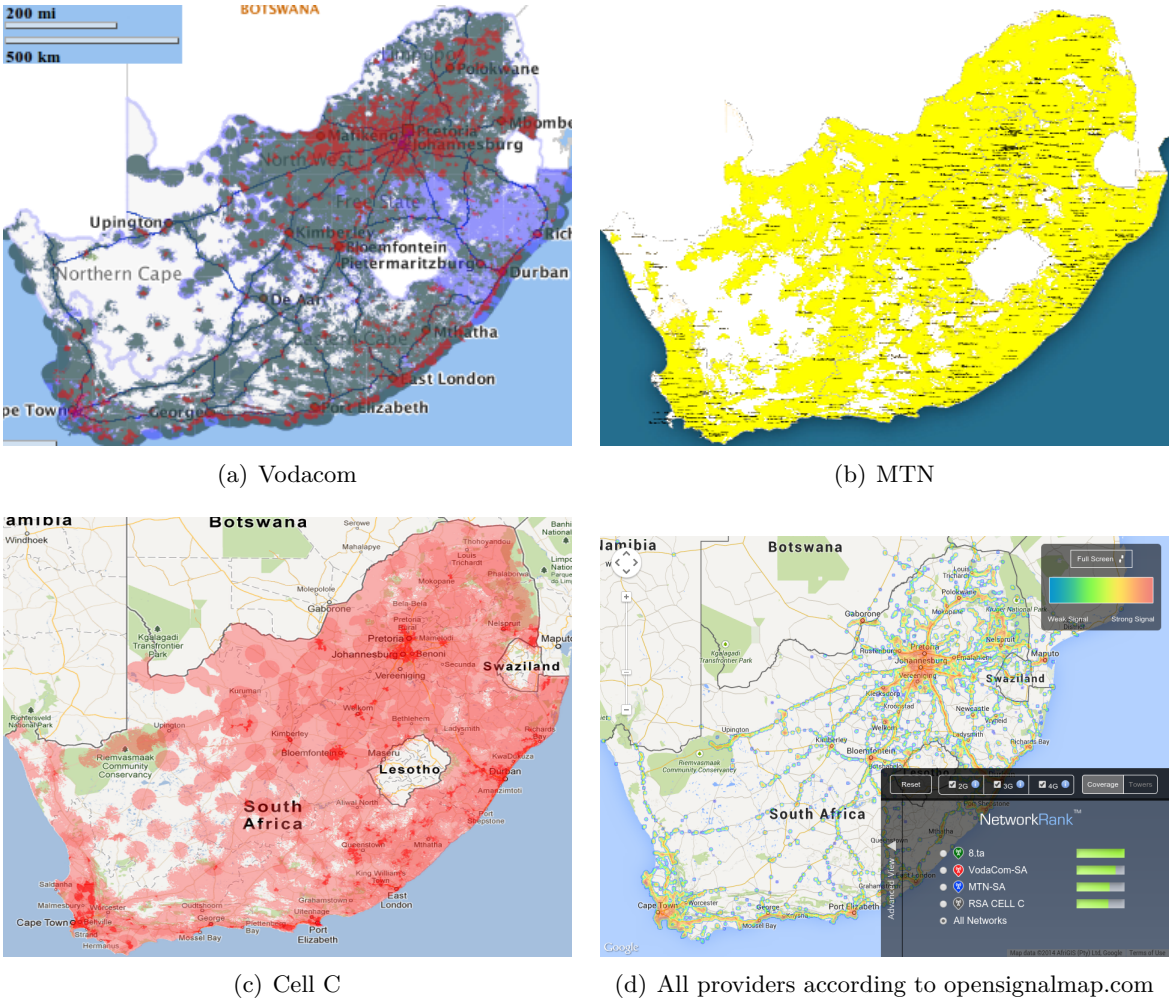
Cellphone radio technology has developed extensively in the past few years and the energy requirements of this technology is relatively low. This is due to the effort put into the development of smartphones and increasing their battery life. GSM radios will work well in areas close to cities and towns, but unfortunately the most wildlife parks are situated in very rural areas. GSM coverage in these remote areas is very limited and in some cases non-existent.

The coverage supplied by the major South African cellphone service providers can be seen in figure 2.1. The theoretical coverage as seen in 2.1(a), 2.1(b) and 2.1(c) differs considerably from the actual measured coverage in 2.1(d). All of the service providers exclude the Karoo and Kalahari area in the central west of the country. And this is in fact exactly an area we need to target.

It is possible to increase the coverage of GSM by erecting picocells, but this is not financially viable when the sparse distribution of the animals and collars are taken into account.

2.5.3 Wifi, Bluetooth, ZigBee

WiFi uses either or both the 2.4GHz and 5GHz bands. Bluetooth and ZigBee use 2.4GHz. As will be seen in chapter 5, these high frequencies do not have the coverage range we are looking for, due to higher free space loss at higher frequencies. Regulations also limit the radio transmission power to very low levels at these frequencies.



(a) Vodacom (b) MTN

(c) Cell C (d) All providers according to opensignalmap.com

Figure 2.1: Signal coverage for the three major South African cellphone service providers.

2.5.4 Radio

Long range terrestrial radio links is the last communication media we look at. The maximum distance a radio signal can reach is proportional to the square of the power being transmitted (inverse square law). The power requirements for a radio on a tracking collar to reach a location with internet access, where the data can be relayed anywhere, are normally too high. A better solution would thus be to use a network of radio links, decreasing the path lengths and thus exponentially decreasing the power required for every radio link.

The frequency used for the radio links also has an influence on the maximum distance the link will reach (free space loss). It can be calculated using the Friis transmission equation when the power output is known. This topic will be looked at in more detail in chapter 5.

2.6 Networks

As the animals are constantly moving around, a fixed network will not work. Ad-hoc principles should thus be looked at to configure the network dynamically.

In an ad-hoc network all nodes have the same level of authority. There is no central controller defining how everyone should communicate. Every node has a view of its neighbours with whom

it can communicate, and he can decide himself where to send and forward packets. In the most simplest case flooded routing is used to propagate a message through the network, ensuring that it reaches its destination. In literature ARPAnet and the amateur radio packet network (using AX.25) are the first examples of ad-hoc networks.

2.6.1 Wifi, Bluetooth, ZigBee

Both WiFi and ZigBee have ad-hoc capabilities that would work well for us, but they are restricted to the hardware and thus frequencies used for the two standards. One extended protocol on WiFi which seems to be very good is called Batman (better approach to mobile ad-hoc networking)[9].

Bluetooth Low Energy is quite a new standard which can provide short range radio links at very low power[10]. This could be used for short range communication between sensors on an animal, but does not provide us with multi-hop network capabilities.

2.6.2 Zebranet[11]

As the name indicates, ZebraNet is an ad-hoc mobile wireless network designed for tracking herds of zebra. The entire protocol is written in such a way that the tracking collars themselves form an ad-hoc network. Losing one collar won't influence the measurement much as there are many other zebras in the same herd providing similar data.

Zebranet is designed to track herds. The design kept in mind that a herd has many tracking collars distributed through it. A researcher will be able to quite easily find the herd and get close enough to make radio contact with at least one of the collars. As the collars are in a network, the researcher can thus obtain data from all the collars in the herd, even if only one can be reached by radio.

In our case we want to monitor solitary animals like leopards and caracals, and ones that live in small groups like jackals. We specifically do not want to use the tracking collars as part of the network. Using collars as routers in the network will shorten the lifetime of the collars, and compromise the integrity of the data it captures. We only have one animal providing us with data, not an entire herd.

2.6.3 Geo-routing algorithms

[12, Chapter 15: Position-Based Routing in Wireless Ad Hoc and Sensor Networks]

A large number of geo-routing (or position-based routing) protocols exist. Some of them use real coordinates from GPS and others use landmarks and virtual coordinate systems. The most of these protocols suffer from one or more of the following issues:

- Dead-ends while routing a packet, thus not guaranteeing delivery.
- Address spaces that grow too big to be implemented on devices with limited memory.
- Mathematical simplification of routing graphs that are too complex for an embedded microcontroller to compute.

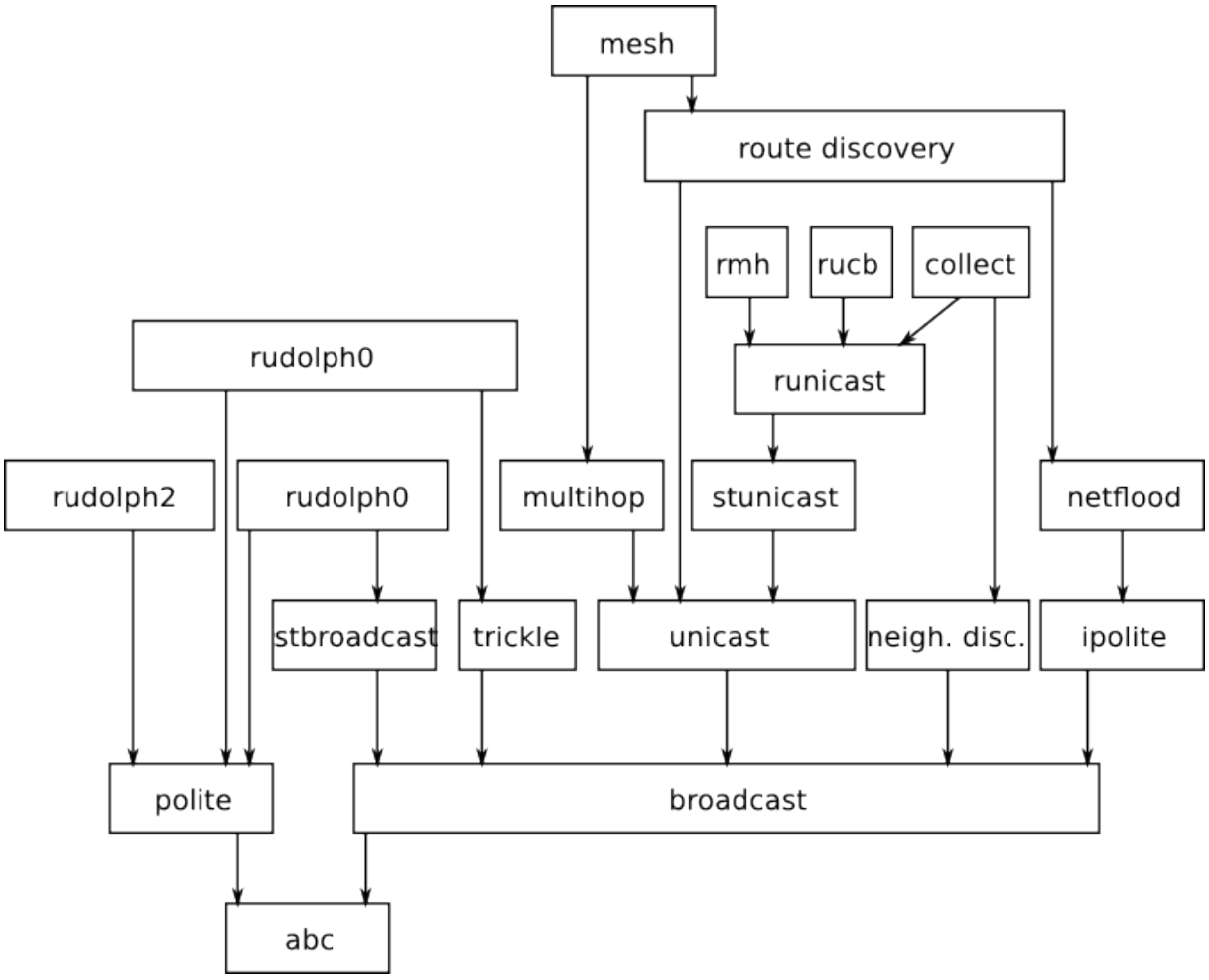


Figure 2.2: The lower levels of the Contiki Rime protocol stack with mesh networking at the top, above which IPv4 can be implemented. At the lowest level is anonymous broadcast (abc) which is synonym to a simple radio transmission.

Even though the most of the existing protocols have problems, it is possible to combine some of them and create a simple but effective protocol for our application.

2.6.4 Contiki OS and Rime

[13] Contiki is an operating system designed for low power embedded microcontrollers. This is the same type of hardware used for tracking collars. Contiki was designed with the "Internet of Things" in mind. The idea is to connect anything, even how simple or small, to the internet. To reach this goal Contiki has built in ad-hoc networking capabilities. It consists mainly of two protocol stacks: Rime and IPV6RPL. Rime is the older of the two and is built up by a number of protocol layers. The lowest protocol layer, the physical radio broadcast, and the higher one, the mesh network, are worth mentioning. On top of the mesh network layer a number of other protocols can be used, including IPv4 low-pan (low power personal area networks) providing UDP and TCP capabilities for communication via the internet.

2.6.5 Contiki OS and IPv6RPL

The second and newer protocol stack used by Contiki is IPv6RPL[14]. It is a IPv6 implementation specifically designed for low power and lossy networks (LLNs). The IETF draft for this protocol also specifies that LLNs "typically operate with constraints on processing power, memory, and energy (battery power)". It also discusses the ad-hoc capabilities of the protocol, supporting route discovery and the ability of more than one sink node as long as they are connected to the same back-bone network.

This protocol is compatible with the internet, has an official protocol specification, has the ad-hoc capabilities, and complies to the power constraints required in wildlife monitoring. It therefore seems like the best candidate to use.

Contiki has (as far as the literature survey has shown) the only fully working IPv6RPL implementation available. Contiki is a widely used (e.g. by researchers at INRIA, France[15]) and relatively user friendly operating system. It however only supports a few embedded systems and the IPv6RPL protocol stack only supports very specific radios.

2.6.6 The problem with Contiki

It is possible to buy the hardware that is supported off the shelf (`http://www.contiki-os.org/hardware.html`), but the radios and frequency spectrum and power output used (mostly 2.4GHz at 100mW) means a very small coverage area. The number of nodes in the ad-hoc network would be too large to build an affordable tracking network in our situation.

Further research is therefore done into lower frequency radios and network, using affordable, off-the-shelf radios operating in the VHF and UHF bands.

2.6.7 APRS

Automatic Packet Reporting System (http://www.aprs.org/) is a network protocol, normally operating in the VHF and UHF bands, that was designed by amateur radio operators[16]. It is mainly used for broadcasting geographic coordinates, but also has functionality for weather reports and peer-to-peer messaging. APRS is built on the AX.25 data link protocol[17], giving the ability to create connections for reliable transmission of data between two nodes, or broadcasting data packets to any receiving station.

AX.25 and APRS together is a simple plaintext protocol with many implementations available. A few of these are meant for an embedded system. It is therefore possible to modify and use this protocol for a wildlife tracking and monitoring network.

One huge benefit of AX.25 and APRS is that it is designed to use existing voice communication radio networks. Off-the-shelf hardware as well as existing repeaters and radio installations can be used with minor to no modification.

2.6.8 CCN as ad-hoc protocol[18]

A new movement in wireless sensor networks is towards content centric networking. The idea behind it is to address the data in a network, rather than addressing the nodes in the network. Rather than asking a node if it has the data you are looking for, you will rather broadcast

a request for certain data. Thus addressing the data. Any node with the data in its store (sensors) or cache, can answer to the request. If a node does not have the data, it will forward the request by sending another broadcast. It can be seen from this that no other ad-hoc protocol is necessary, as the broadcasts will reach all nodes in the network and no explicit node addressing is necessary. A comparison of CCN with the normal address-based approach is seen in table 2.1.

Node Centric	Content Centric
Ask DNS server "Who has google.com?", and get an IP address back.	We broadcast a request: "I want google.com!"
Connect to the host with that IP address and ask for the content google.com.	Anyone with a copy of the data will respond.

Table 2.1: Comparison of node centric and content centric network approaches.

As CCN is broadcast-based, it inherently fits well with networks using radio broadcasts as the physical medium. The Contiki operating system mentioned earlier also has an implementation of CCN[15]. Except for Contiki's implementation, only two other prototype implementations of this protocol exist. The most feature complete implementation is called CCNx, while a more slimmed down version for use on microcontrollers is called CCN-lite. During this research no commercial implementation or use of CCN could be found.

CCN is very much a pull-based data network (or polling network), where a central point will constantly query the nodes in the network. This is handy in a home-monitoring environment where the status of a light or the temperature of a room needs to be requested. In wildlife monitoring, tracking collars might not always be within radio coverage to receive the data request. They will normally gather data and "push" the data upstream towards a central point whenever it has been scheduled, if an event occurs or when the collar has radio coverage.

CCN is thus not the ideal solution for a wildlife monitoring network.

2.7 Summary

This chapter looked at a number of existing technologies that can be used for wildlife monitoring. Their drawbacks and advantages were highlighted to give an idea which of the mentioned ones are the best for continued development.

In the next chapter the general questions asked by wildlife researchers are discussed and preliminary work in answering these questions by using technology is shown.

§ 3. Description of preliminary work

While designing a new system, the usage needs to be kept in mind. In our case, we need to know which questions the biologists have that the technology needs to answer. Where is the animal? What did the animal do? Is the animal still alive?

Keeping this in mind and looking at the current technology, we can specify more guidelines to create an improvement on what exists. We want to prevent the manual tracking down of animals to download its data. Live or semi-live feeding of data to the internet is preferred to prevent researchers to do unnecessary fieldwork.

3.1 Where is the animal?

The first question can be answered by getting the location from a GPS receiver on the tracking collar and transmitting the location at set intervals. To save energy the location can alternatively be logged periodically on the tracking collar and downloaded at a later stage.

3.2 What did the animal do and is it still alive?

Researchers normally use accelerometer data to analyse what an animal does[19]. It can tell us if an animal is running or walking, or if it is lying on its side or back. If no movement is detected it is unlikely that the animal is still alive, as even breathing will cause a measurable movement[20, Activity-mediated mortality-sensing]. Data captured during a test with a motion sensor on a cow's ear indicates that a cow very seldom keeps its ear still. The ear is therefore a very good location to put an activity-based mortality sensor. During the ±40 hours this test was executed (results shown in graph 3.1), the cow only kept its ear still for a maximum period of 136 seconds (about $2\frac{1}{2}$ minutes).

A second test done on the cow tried to measure its heart rate using pulse oxymetry (Photoplethysmography). A light source, which is normally red or infrared, is shone towards the skin, and the intensity of either reflections or light passed through the body is measured. Theoretically this can provide one with the heart rate and blood oxygen level. This principle is commonly used on humans. The sensor did not work as expected on the cow. Most likely the thick layer of hair on the cow's skin does not allow enough light through for the measurement to work. Likely the result will be different if the hair is shaven off at the spot where the measurement is done. More research needs to be done in this area, as heart rate is the best way to determine stress levels and animal health. Doing this with a non-intrusive sensor is preferable.

A third way of determining the health of an animal is by measuring its temperature[20, Temperature-mediated mortality-sensing]. Doing this in a non-intrusive way is difficult, as any sensor on the skin will be affected by sun, rain and wind.

3.3 Measuring and logging animal motion

Two types of sensors exist to measure motion. The most well-known one is an accelerometer, which is an integrated silicone circuit using a capacitor to measure acceleration. The capacitor has three electrodes, of which the outer two are fixed while the middle one is floating. When acceleration occurs, the middle electrode will move due to inertia (Newton's First Law of motion)[21, p. 8]. Figure 3.1 shows the basic circuit layout of this sensor. This capacitor can measure acceleration in one axis. Three of them are therefore needed for a 3 dimensional acceleration measurement.

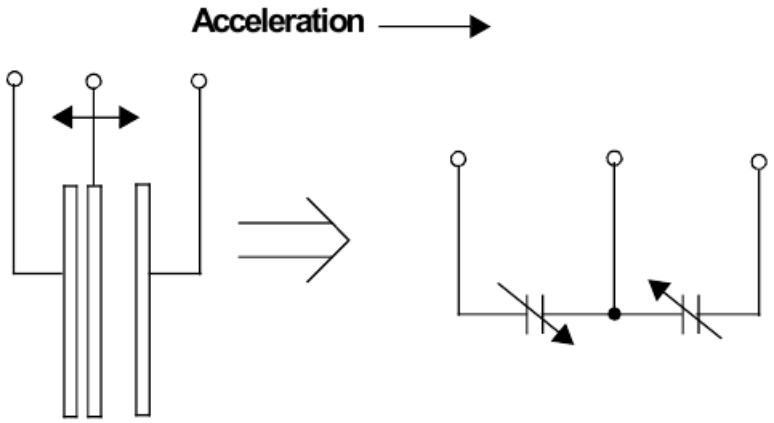


Figure 3.1: Internal working of an accelerometer.

A circuit was built to log acceleration at a constant sample rate. Data is stored on an SD-card. Timing of the SD-card write cycle is not guaranteed as the firmware of the SD-card performs multiple background tasks, like wear levelling. To guarantee a constant sample rate, even though the timing of storing the data is not guaranteed, an algorithm was devised. More details on the design of this circuit and software can be seen in appendix A. The sensor was tested at multiple sample rates and it was seen that logging up to 250Hz was quite reliable. After discussions with a signal processing engineer it was decided that a sample rate of 100Hz will be more than sufficient and provide extra time between samples to write to the SD-card.

Using only an accelerometer means the sensor needs to be powered up and sampled constantly. This means power usage of both the sensor and the microcontroller. An alternative way is to use a passive motion sensor (mercury switch). This sensor acts like a switch, either being an open circuit or a closed circuit. When it is in the closed circuit state a current flows, but by using a very high value resistor in series this current can be extremely low.

When the microcontroller is in sleep mode, motion will cause the passive motion sensor to switch, which will trigger an interrupt on the microcontroller which will wake it up. The accelerometer can now be powered up, it can be sampled for a set period of time (depending on the use), and powered down again. The captured data will be analysed. If something is detected worth noting, the data can be logged onto an SD-card or an alarm message can be transmitted by radio. After this process the microcontroller can go back to sleep mode where it will stay until it is woken up again, either by a periodic wakeup from the watchdog timer, or the passive

motion sensor. Power usage of the microcontroller in sleep mode is orders of magnitude lower than in normal powered mode. The values can be seen in section 8.3. The algorithm that was just described can also be seen in figure 3.2. A system like this was used to monitor motion on a cow's ear. It did not include the accelerometer, but counted the number of passive motion triggers per 8 second interval for a total duration of two days. The battery unfortunately ran out after 40 hours. The results can be seen in graph 3.1.

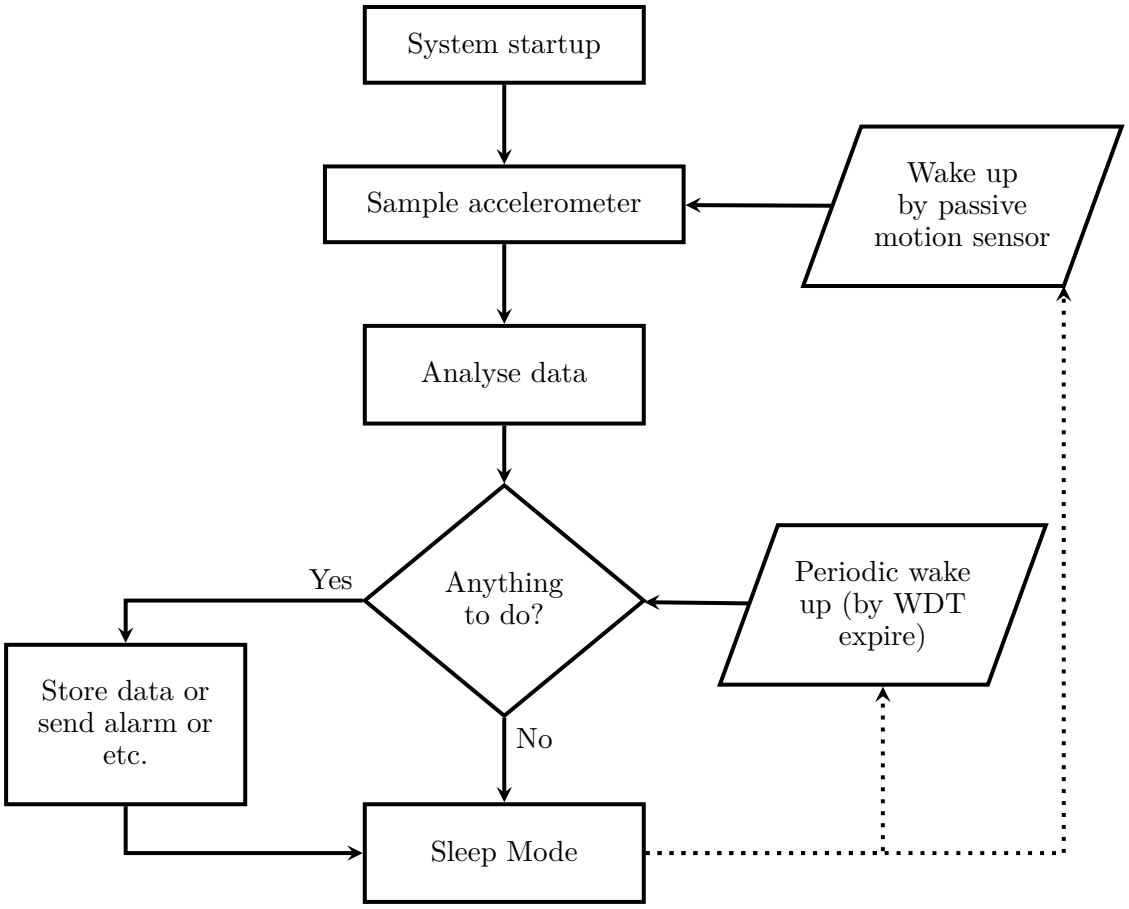
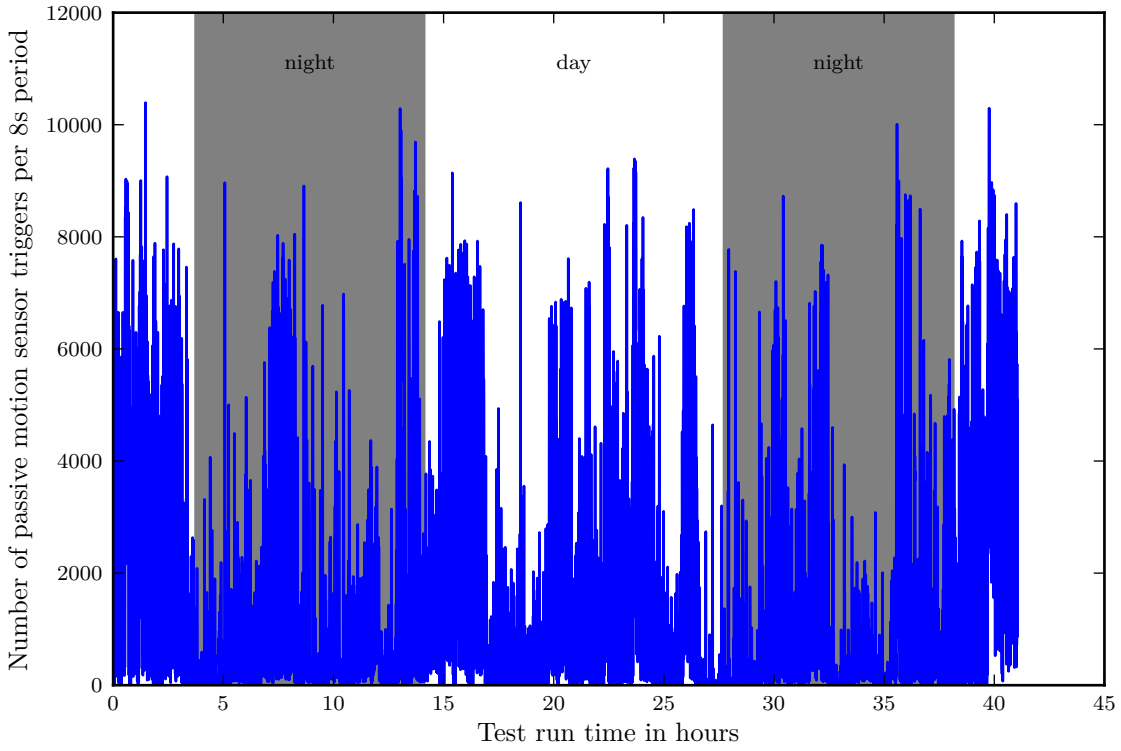


Figure 3.2: Flow diagram of a system using both a passive motion sensor and an accelerometer.

Timestamping of data is normally done by a real time clock (RTC), which is a separate hardware component. These components are designed to be very power efficient, so to add one will not make a noticable difference in the total power consumption. If a GPS is included, it can be periodically switched on to synchronise the RTC.

An even more efficient way to do this is by using the already available watchdog timer (WDT) as a clock. In our case the WDT was set up to cause an interrupt every 8 seconds. If the WDT is never reset by any software, we know that every 8 seconds the interrupt function will execute. We can use this to increment our real time variable. Again, this can be periodically synchronised using the GPS receiver to correct any drift in time that may occur. Using this method we save on component costs and total energy requirements.

Graph 3.1: Number of passive motion sensor triggers per 8s interval for a test on a cow's ear.



3.4 Summary

This chapter looked at the three main questions wildlife researchers have.

- Where is the animal?
- What did the animal do?
- Is the animal still alive?

Ways of answering these question using technology were discussed. Focus was laid on how to do the measurements in a reliable, but extremely power efficient way.

The next chapter will look at ways to incorporate these sensing techniques into a system consisting of a radio network.

§ 4. Proposed methodology

This chapter discusses the layout of a complete wildlife monitoring system as proposed in this research. An overview of the system layout is shown, followed by specifics for every type of node in the wireless network that is used.

4.1 System Layout

As an animal walks through the bush, repeater nodes in the area should be able to receive the data the collar transmits. The data should then be relayed to a central base station (or multiple base stations). A base station will be connected to the internet (likely by GSM), over which it can forward the data to a central database. Figure 4.1 indicates this layout.

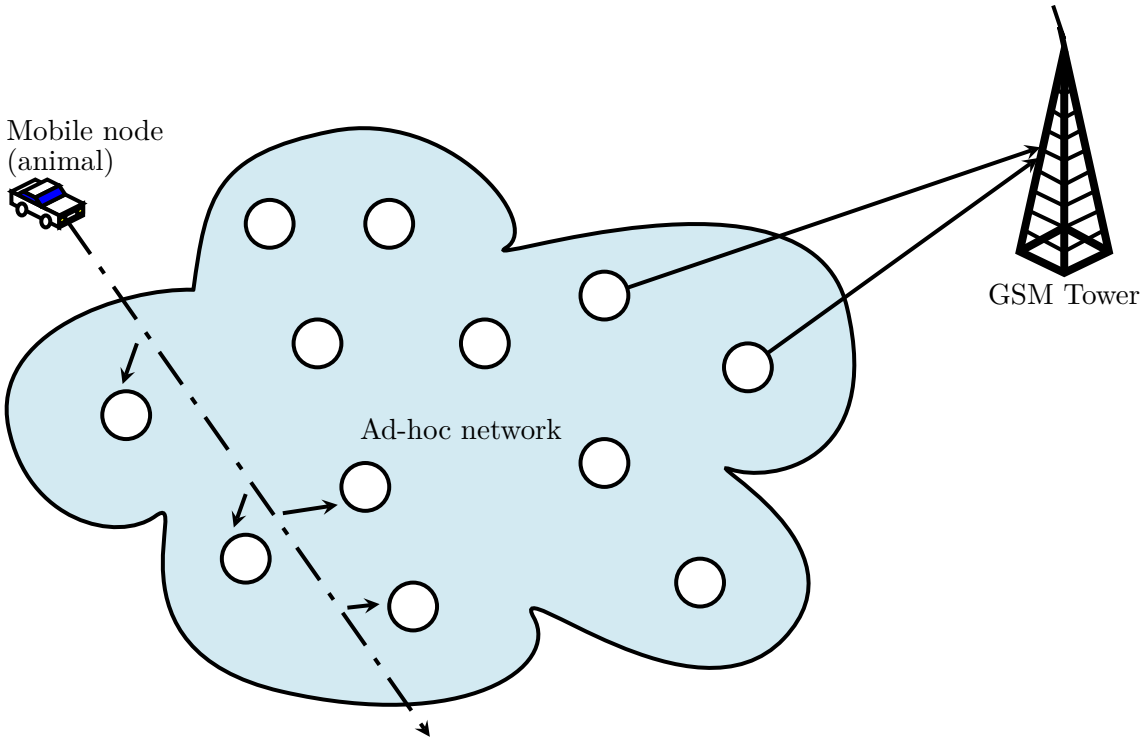


Figure 4.1: General layout of the system, with the mobile node (animal) travelling through a network of relaying (repeater) nodes.

If more than one base station received a message the central database will receive the message multiple times. This redundancy is good and using a specific metric like signal strength or hop counts, the best copy can be chosen and acknowledged via the received path.

Looking at the payload on the animal, we have a choice of two architectures. The network behind the system is not affected by which architecture is chosen. The logical layout of the system can be seen in figure 4.2.

1. The payload on the animal is only a sensor node, transmitting status and alarm messages. To have full coverage, this needs to be the same frequency and power output than what is used for the relaying network.

   Positive Only one module on the animal, which makes it lighter.

   Negative Power usage of the sensor is high, as a long range radio needs to be used.

2. Two separate modules on the animal. The one is an ultra low power sensor node, transmitting alarm and status messages over a short range low power radio link to a second animal-borne module. The second module is a bridge node, receiving data on the low power, short range radio link, and transmitting it on a higher power, longer range radio link.

   Positive Power usage of the sensor node is very low, needing smaller batteries and therefore making it very light. Because it is lighter more locations on the animal's body can be used for monitoring, like the ear. The weight that is saved is offset by a second module which can be fastened at a more solid location on the animal, like around the neck or foot. This second module can therefore have a slightly larger battery, and also host the bigger and power hungry components like the GPS receiver.

   Negative Multiple modules need to be fastened to the animal.

For the hardware design the focus was on the second option, to have a separate sensor module and separate bridge module. If a smaller animal is being monitored and only one module is to be installed on the animal, the bridge node and sensor node can be integrated, also saving some components.

4.2 Sensor node

To make the sensor node as light and power efficient as possible, only the components necessary to do a measurement and relaying the data are included. This can include the following:

- small battery (small solar cell)
- passive motion sensor
- accelerometer
- thermometer
- heart rate sensor
- low power, short range radio

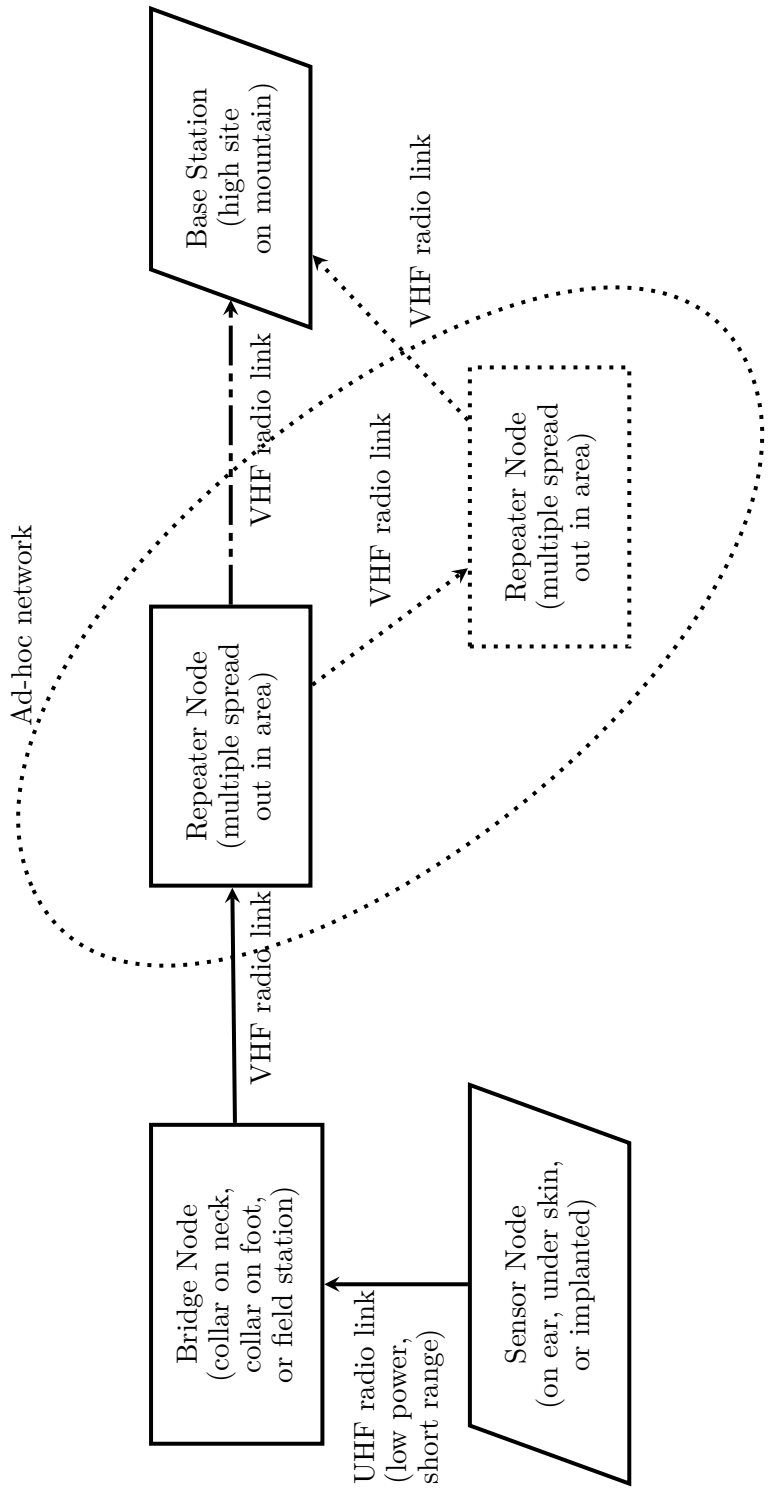


Figure 4.2: Network systems diagram. This diagram displays the logical layout of the system and the locations of all the nodes in the network.

4.3 Bridge node

All other necessary components that need to be on the animal can be included on the bridge node. The likely scenario is that no meaningful measurements of the animal can be taken where the bridge node is installed, but in some cases it might be useful. For example, if the bridge node is fastened to the foot of an animal, the number of paces given per day can be obtained by including a pedometer. Other components that may be necessary are:

- Short range radio to receive sensor data.
- Long range radio to forward any data to the network.
- SD card for keeping a buffer of data if it can not be delivered immediately, or for debugging purposes.
- GPS receiver for determining the location of the animal, as well as the real time.
- Larger battery
- Power generator by means of solar or motion energy.

4.4 Repeater node

The repeater node is just a simple message forwarder, performing flooded routing. It needs to listen permanently for messages on the radio, when a message is received it will be transmitted again. As these nodes will be spread out in the bush and being static by nature, powering them is a little easier. For this we need the following components:

- long range radio
- large battery
- large solar cell
- omnidirectional antenna

The abilities of the repeater node can be augmented with an electronic steerable antenna. This will improve the radio coverage and the reliability of the radio links. Such an antenna can also be used to determine the location of a transmitter by using trigonometry. An electronic steerable antenna for the 2.4GHz band was developed and is described in Appendix C. The results indicates good performance. Physical size of the antenna at lower frequencies, especially in the VHF band, is however too large to be practical.

4.5 Base station

The assumption is that a base station will be installed at an already existing high site. High sites normally already have electricity supplied, and the chance of cellphone reception, or internet by means of long range wireless links is almost certain.

A high site can cover a greater area than a repeater node, as the height and line-of-sight characteristics of high sites are very favourable to radio propagation. Using a good radio and modem combination at the base station will guarantee a greater coverage area and reliability. The following components are advised for the base station:

- Commercial two-way radio with good sensitivity characteristics.
- Omnidirection (azimuth) antenna, maybe with more gain towards the horizon (elevation). Well matched for the used frequency.
- Embedded Linux computer to do the processing of the data, and perhaps also do demodulation of the signal. A Raspberry Pi will suffice.
- Modem to do modulation/demodulation of the signal, or a soundcard for the computer in case soundcard modem software is used.
- An internet link by means of ethernet, 3G or other means. For 3G a USB modem, router and perhaps even a direction antenna will be needed.

4.6 Network

A simple flooded routing protocol is sufficient, as the total number of transmissions is very limited. It will also keep complexity of the system down. This will allow the use of simpler and cheaper hardware. The animal-borne module will do a broadcast, wait for an acknowledgement and either stop when one is received, or retry after a set time out period, for a certain number of times.

When a message is received by a repeater, the repeater should append its address to the message and broadcast the message again. If a message already contains the repeater's address, do not forward it to prevent loops in the network. Alternatively if no addressing is used, messages should contain a unique number. A list of a number of previously forwarded messages should be kept. If a message is received that is in this list, it should not be forwarded again.

By doing flooded routing, messages can arrive at more than one base station. If all of them acknowledge the same message, the reliability of receiving and acknowledging a message is higher. The base stations all send the received message to a central database on the internet where duplicates are filtered and the best data kept. Reverse communication towards the animal is also now possible as the base station which received the message the best (according to a metric like signal strength, or hop count) is known. A message can therefore be sent to the animal from the best base station.

All of the characteristics mentioned above are provided by the APRS system[16], built on the AX.25 protocol[17]. It uses a simple audio-frequency modulation scheme which can be modulated and demodulated by low power embedded systems. Because this system already exists for many years and has been proven to work well, it is the best choice to use. There is no point in inventing a new protocol if one already exists that can provide all the features we need.

4.7 Central Database server

A server running the LAMP or WAMP server software stack would be suitable as the central database for the network. LAMP stands for Linux, Apache, MySQL and PHP. WAMP is the same, but with Windows as the server operating system. These server software stacks provide website capabilities, backed by a MySQL database and is programmed using the PHP server side language.

A PHP program can be set up to act as a REST-like API, listening for communication from any one of the base stations. Alternatively the official APRS-IS (APRS internet system) can be used, or cloned for the wildlife tracking purposes. The latter option allows the use of existing base station software known under the term "IGate" (`http://info.aprs.net/index.php?title=IGate`).

The website capabilities can be used to provide a user interface to the ecologists, biologists and game rangers. With the availability of smartphones it can even be used by these people while working in the field.

The work needed to set up a server for tracking and telemetry purposes was already discussed in a previous project[22]. It is therefore not mentioned again in this thesis.

4.8 Summary

In this chapter a proposed layout for the wildlife monitoring network was discussed. Focus was given to every part of the system as it is seen as nodes in the network. Requirements for the network protocol that should be used were discussed. A proposal was made for an existing protocol that suits the requirements.

Next, a closer look will be given to the actual radio links, specifically which frequency and antennas should be used.

§ 5. Simulation of radio coverage and choice of frequency band

Radio coverage simulations are done to predict which frequency band will provide the best coverage in our environment. Antenna size is inversely proportional to the frequency used. As our animal-borne equipment should be as small and light as possible, this also dictates the usable frequency bands to some degree.

5.1 Radio coverage of an appropriate area using terrain analysis software

The software package Radio Mobile is used for these simulations. Radio Mobile is a freeware application that is available to download online. It is developed by Roger Coudé, a radio amateur with callsign VE2DBE from Quebec in Canada[23]. The software is mainly meant for amateur radio and humanitarian use, but commercial usage is allowed without any responsibility or guarantee from the author [24].

Radio Mobile makes use of elevation data obtained from the Space Shuttle Radar Terrain Mapping Mission (SRTM) to calculate line of sight, and predict radio path loss using the Longley-Rice propagation model [25].

The following coverage simulations are done in an area that is representative of the general location where wildlife monitoring would be used for monitoring of jackals and caracals. A location of 30.446827°S, 20.421613°E is used, a relatively clear area just west of the town of Brandvlei in the Northern Cape of South Africa.

Simulation settings mostly used the defaults in Radio Mobile, with minor changes mostly to the frequency and power output. All simulation parameters are given for ease of reproducibility.

5.1.1 Simulation parameters

Frequency as per each simulation below

Polarisation Vertical

Mode of variability Spot, 70% of situations

Additional loss none

Surface refractivity 301 N-Units

Ground conductivity 0.005 S/m

Relative ground permittivity 15

Climate Continental temperature

5.1.2 Topology

Visible yes

Data net star topology (Master/Slave)

Transmit power 0.1W (20dBm)

Receiver threshold $1\mu V$ (-107dBm)

Line loss 0.5dB

Antenna omnidirectional with 2dBi gain to the horizon

Antenna height Both static base station and the mobile station are set to use antenna heights of 2 meter.

5.1.3 Simulation results

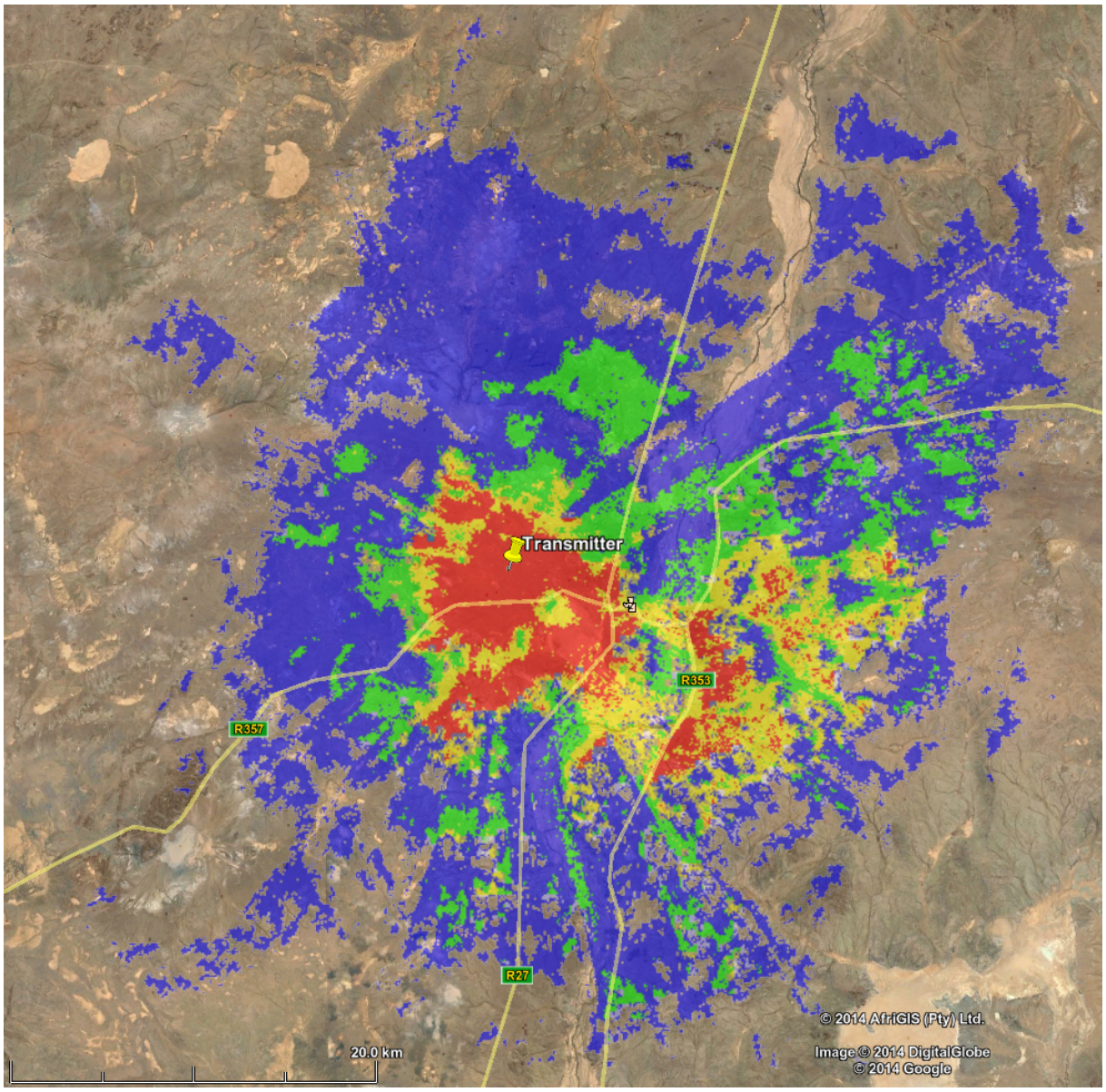
The results from the simulation can be seen in figure 5.1. To someone with experience in the field the simulation results will look very optimistic. It is true as the sensitivity of the receiver is set quite low, and the effect of vegetation and reflections are not taken into account. Any radio noise and interference will also affect the real world situation. For the purpose of comparing the propagation at different frequencies this simulation is good enough, as all parameters are equal between all simulation runs, except the frequency. Only the frequency can therefore influence the simulation results.

5.2 Simulation of an antenna on the foot of a large animal using FEKO

FEKO is an electromagnetic simulation package meant for identifying radio frequency currents on materials by analysing them with the Maxwell equations[26]. Doing this provides us mainly with the reflection coefficient and radiation pattern of an antenna. As all conducting material in the vicinity of an antenna affects its radiation properties, it is important to do a simulation to determine the effect.

The shape of a large animal of about the size of a small elephant is approximated by cylinders and oval spheres, as can be seen in figure 5.3. A large animal was chosen as the volume and mass of the body will have a great effect on radio propagation. The model also includes a plane on which the animal stands, approximating the ground. The permittivity used for the animal is roughly the same as human fat, and the permittivity of the ground layer is about the same as ceramic. The area we are interested in is quite dry and therefore a dry material like ceramic is a good match for a simulation. FEKO has built-in models for many materials, including the two we use.

In figure 5.3(b) the antenna can be seen just left of the right back foot. The simulation used a compact helical whip antenna for VHF frequencies (144MHz), as is also commonly used for walkie-talkie type portable two way radios. On UHF (433MHz) a quarter wavelength monopole



Colour	Frequency band
Red	2.4GHz ISM band used by WiFi and ZigBee
Yellow	868MHz ISM band
Green	433MHz ISM band
Blue	145MHz amateur radio band

Figure 5.1: Simulated radio coverage for four different frequency bands.

antenna was used at the same location. The results of the simulation at 144MHz and 433MHz can be seen side by side in figure 5.4 and figure 5.5.

A helical whip antenna has about the same radiation pattern as a quarter wavelength monopole antenna. Even though the physical length of the helical antenna is much shorter, the electrical length is about the same. For this kind of helical antenna the radius of turns is much shorter (less than a tenth) than a wavelength of the operating frequency. It operates in what is called the “normal mode” of a helical antenna and radiates sideways like the

monopole[27]. To make sure the radiation patterns of the helical antenna in normal mode match the radiation pattern of the monopole, they were both simulated. The results can be seen in figure 5.2.

The physical parameters used for the helical antenna is based on a real antenna commercially available for the VHF band. It has 70 turns of radius $4mm$, and a total antenna length of $16cm$.

5.3 Frequency and antenna choice

From the simulation in 5.1 it can be seen that a lower frequency gives a better range. The 144MHz band will therefore be the most effective one to use.

Section 5.2 looks at how different frequencies radiate from an antenna in close proximity to a large animal. It can be seen that a lower frequency is attenuated less by the body of the animal, giving a slightly more uniform radiation pattern with less directions of no radiation. Therefore it will be better to use a lower frequency.

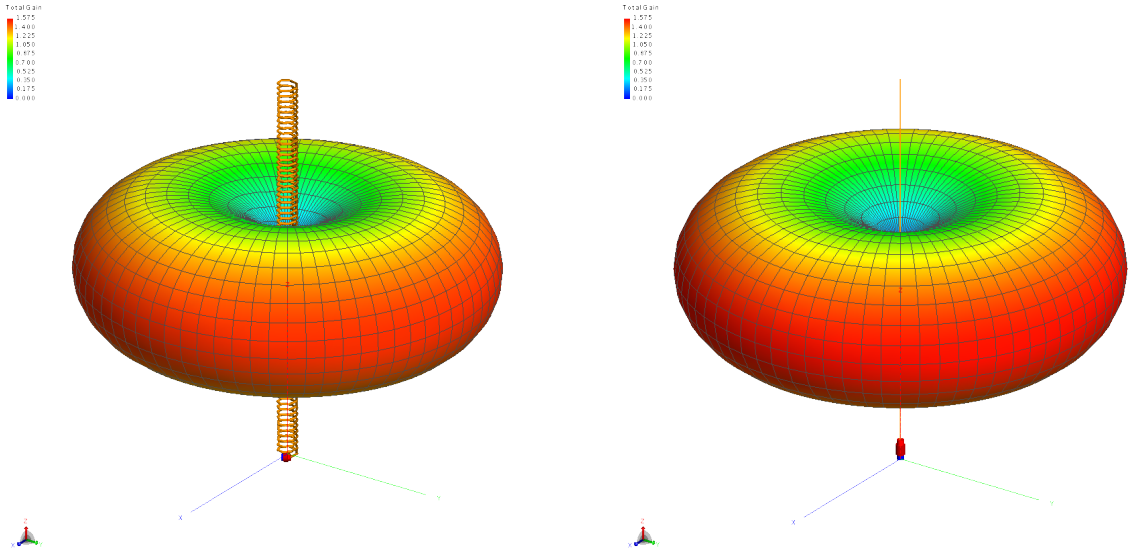
A lower frequency translates to a bigger or longer antenna. As can be seen in 5.2 and 5.6 a helical antenna can be used to limit the physical size, but still maintains good radiation properties. It is possible that a $16cm$ long VHF antenna might be too big for smaller animals, in which case a $5cm$ long helical antenna for the 433MHz band would work better. This will however compromise the range of the radio link.

There is therefore no single answer to which frequency to use. A good rule of thumb will be to opt for the lowest frequency for which the antenna will not be too large for the specific animal you are working with. This is the same conclusion that was made by D. Warnich in his thesis[1]. For the purpose of this project it was decided to use the 144MHz band.

5.4 Summary

In this chapter the reasons were discussed why the VHF frequency band is a suitable, and likely the best choice for radio communication in a wildlife monitoring network. A lower frequency has a wider coverage range, but the antennas are larger. A higher frequency requires smaller antennas, but their range is restricted. Using simulations this chapter has shown that the VHF band has the best coverage range while using antennas that are still relatively small. An extra result obtained is that animal bodies attenuate lower frequencies less than at higher frequencies.

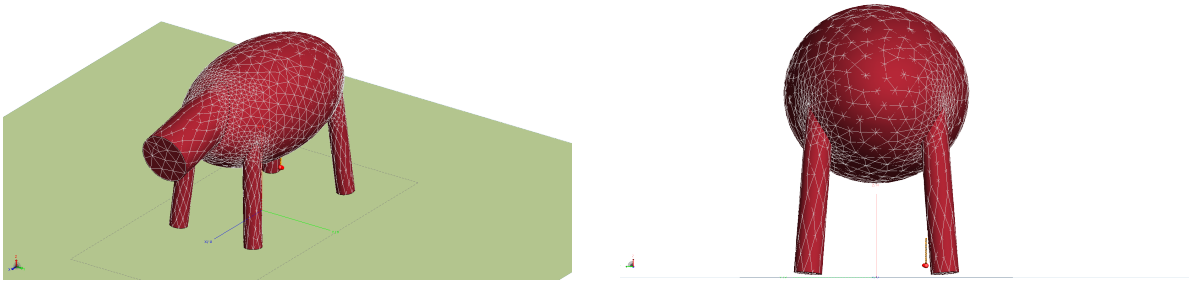
In the next chapter the focus will shift to the hardware, specifically what hardware is available for the implementation of a radio network.



(a) Helical whip antenna

(b) Quarter wavelength monopole antenna

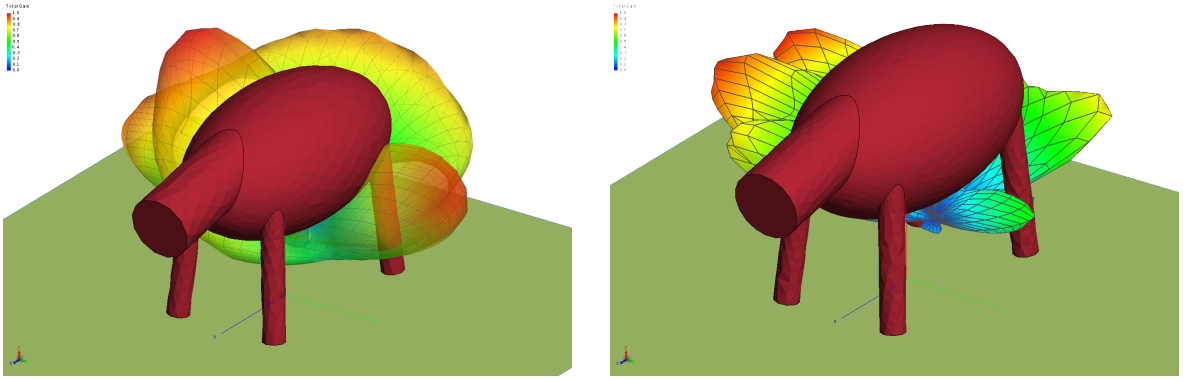
Figure 5.2: Comparison of the radiation patterns of a helical and a monopole antenna.



(a) Isometric view of animal model

(b) Model of animal viewed from the rear

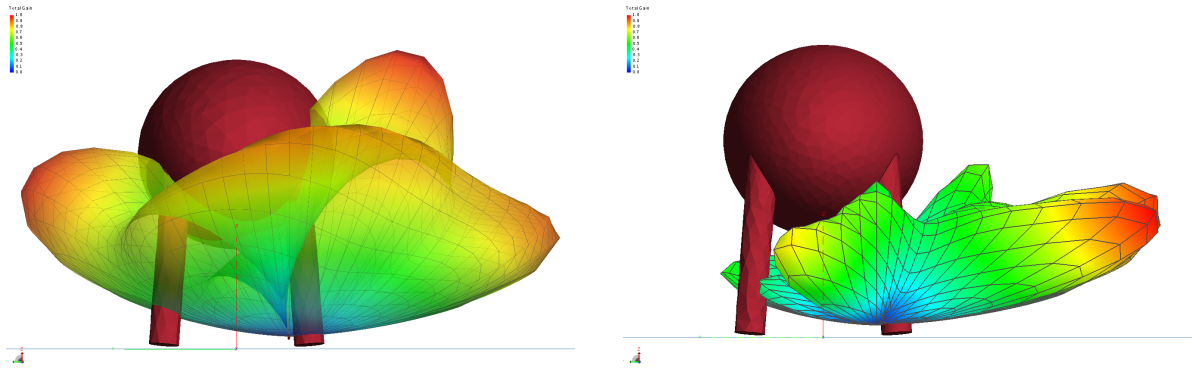
Figure 5.3: Model used in simulation to approximate an animal.



(a) 144MHz with helical whip

(b) 433MHz with monopole antenna

Figure 5.4: Isometric view of simulation results.



(a) 144MHz with helical whip

(b) 433MHz with monopole antenna

Figure 5.5: Rear view of simulation results.

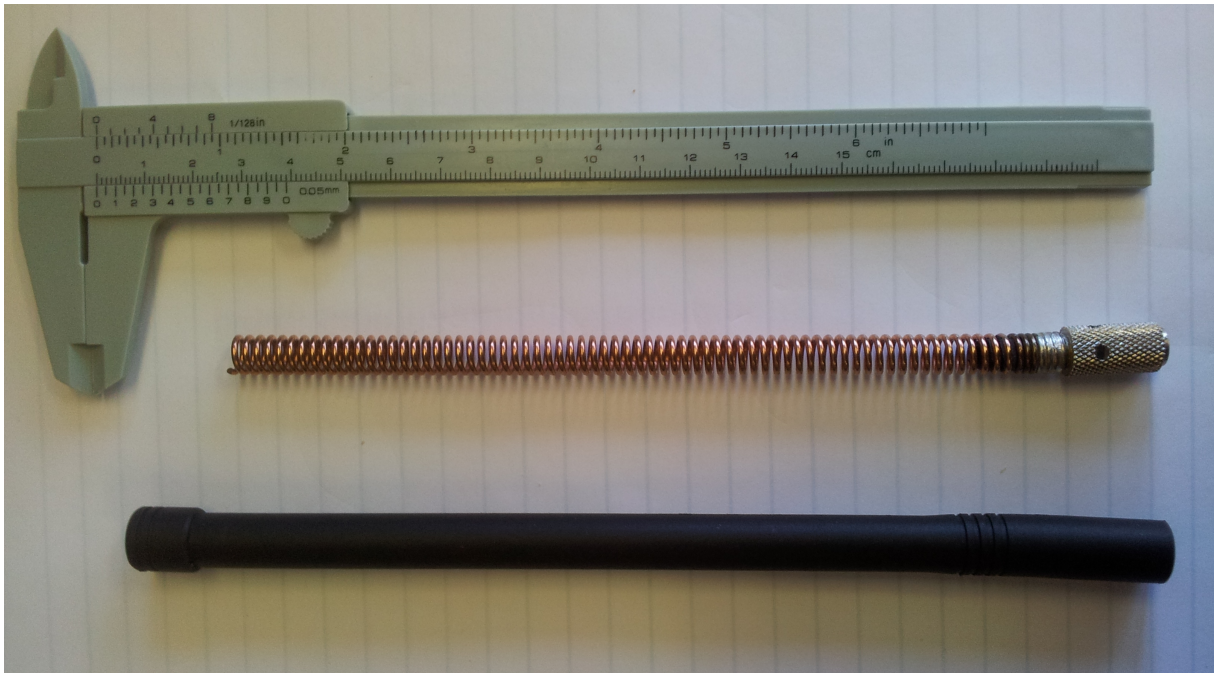


Figure 5.6: A helical antenna for VHF frequencies in comparison to a vernier caliper. Top: vernier caliper, middle: helical antenna without protective insulation, bottom: helical antenna with rubber insulation.

§ 6. General Hardware and software details

This chapter lists radios and microcontrollers that are available on the market and are suitable for our use. It states which devices are preferred. A description is given how the chosen hardware is used for transmission of data across a radio link.

6.1 Radios available on the market

6.1.1 Nordic Semiconductor nRF905[28]

This radio IC is available on a user friendly carrier board manufactured by Montar Manufacturing in South Africa. It can be obtained for R190 from RF Design (`http://www.rfdesign.co.za/`).

The radio operates in either the $433MHz$, $868MHz$ or $915MHz$ bands, using Gaussian Frequency Shift Keying (GFSK) modulation. Maximum power output is $10dBm$ and the maximum data rate obtainable is 100kbps at a deviation of $\pm 50kHz$. The data rate is halved due to internal manchester encoding. Internally an address, preampble and CRC are added to the transmissions.

Because of the internally added extras, this radio is not a low level physical layer on which a custom protocol can be implemented. Keeping to their protocol and addressing scheme one can however create a low power, short range and low cost radio link.

6.1.2 AeroComm AC4868-250[29]

With a power output of 250mW (24dBm), this radio seems like a good alternative as it will be able to provide greater range. The frequency it operates on is 868MHz, and can transmit data up to a rate of 57.6kbps using frequency shift keying (FSK). The radio is available from RF Design for R1007.

The radio has a protocol stack embedded in it, making the use of a custom protocol impossible. It does however provide a number of network topologies in which these radios can operate (server/client or peer-to-peer). This can be a useful tool. The frequency band the radio operates in is not ideal, but the power output is good. Pricewise it is a bit expensive.

6.1.3 Radiometrix BiM3A[30]

This Radiometrix radio operates in the 868MHz band with a power output of $2.5dBm$ using FM modulation. It is available from RF Design for R440. This is a "dumb" radio, providing a baseband input line and a baseband output line, which can be either used for audio or digital data up to 64kbps. In addition to these, separate RX enable and TX enable lines are provided,

allowing one to manually put the radio in the desired mode. The RSSI is also available on an output line.

With the supplied inputs and outputs one has the freedom to implement a custom low level protocol on top of the FM modulation. The output power of this radio is unfortunately extremely low and the frequency band it operates in is not ideal.

6.1.4 Axsem AX5042 and AX50424[31]

These two radio IC's are around R50 each from RF Design. Both operate in the 433MHz and 868MHz ISM bands. The AX5042 can output 14dBm in the 433MHz band. Using FSK modulation it can communicate at 200kbps, or using ASK or PSK it can do up to 600kbps. Some other modulation schemes are also possible. The sensitivity of these radios seem quite good at -122dBm, but can even be improved to -127dBm using the built-in soft Viterbi FEC. Power consumption seems very good as well.

As these are radio IC's, a carrier board with antenna matching circuitry need to be developed before they can be used. Buying a development kit for them is quite expensive at R4300 from RF Design.

6.1.5 Radiometrix BiM2EH

Operating at 433.92MHz and providing an output power of 100mW ($20dBm$), this dumb radio provides us with the same specifications and freedom as the Radiometrix BiM3A, but at a higher power output. At R550 from RF Design it is a little more expensive, and the fixed frequency is not ideal, but it is worth to consider.

6.1.6 Radiometrix SHX1[32]

This Radiometrix radio operates on the VHF band with a power output of 500mW. It has a built-in dumb modem. This modem can be bypassed allowing one to use the module as a dumb radio, giving us full control over the lower level radio. The frequency it transmits and receives on can be programmed via a serial port. This is the only high power embedded radio module that could be found, and it operates on the VHF band, which is ideal for our use. The built-in modem could be used to off-load the modem work from the microcontroller. The radio is available from RF Design at R1180, which is relatively expensive.

6.1.7 HopeRF RFM12B

These radios operate in the 433MHz band and can supply an output of up to 7dBm of RF power. At R95 (including VAT, from Netram), they are the cheapest radios locally available that do not need any extra circuitry to function. Driver libraries already exist to use them with ATmega microcontrollers.

6.2 Radios chosen

As VHF is the preferred choice concerning the frequency at which our system will operate, we don't have much of a choice. Luckily the only radio that operates at VHF frequencies, the Radiometrix SHX1, is also the one with the highest RF power output at 500mW. Even though the price is quite high, it seems to be the best choice in our case. This radio can be used in the repeater node, as well as the bridge node. Interfacing with the radio is done by audio and control lines, just as one would interface with a normal two-way radio.

For the shorter low power link between the sensor node and the bridge node, any low power radio can be used, as it is a point-to-point link, and no custom protocol needs to be implemented. For this project the HopeRF RFM12B radio module was chosen, due to low price and availability. The availability of drivers for this radio also contributed in making this decision.

6.3 Low power embedded microcontrollers

There exist many microcontrollers manufactured by a wide array of companies. The choice in microcontroller is normally made according to the preference of the engineers. They choose the one they have experience with.

A quick review of current embedded projects online will show that the Atmel AVR has a wide community of developers behind it. It is mostly due to the Arduino development board that also uses Atmel AVR's. From the range of AVR's available, one can choose a specific one by number of pins, amount of RAM, amount of ROM, clock frequency and which peripherals are provided.

One specific Atmel AVR that is popular is the ATmega 328P. It is the microcontroller that is used on the Arduino UNO development board. It is an 8bit processor with 32KB of flash memory and part of the pico-power range of low-power AVR's (the reason for the P at the end of the name).

For prototyping the choice fell on the ATmega 328P, as library and driver support for it is very good, and its low power modes can be useful. For initial prototyping the Arduino UNO can be used, shortening development time.

6.4 How AX.25 is modulated and demodulated on embedded microcontrollers

6.4.1 General technical specifications of AX.25 and Bell 202

For 1200baud AX.25 communication, Bell 202 tones should be used for audio-frequency-shift-keyed modulation on the physical layer[33]. Depending on the frequency band used, the physical audio channel should either be FM modulated (for VHF and UHF) or SSB modulated (for HF). In our case the Radiometrix SHX1 radio module does FM modulation, so we need to provide it with a Bell 202 modulated audio stream.

In the article *Clarifying the Amateur Bell 202 Modem* by Finnegan and Benson, the Bell 202 modulation, as used in amateur radio, is discussed in detail. It serves no point to reiterate this

discussion. The most important part to take from the article at this point is that Bell 202, as it is used for amateur radio, uses a 1200Hz and a 2200Hz audio tone to indicate marks and spaces. A transition between a mark and a space indicates a binary 1, and no transition indicates a binary 0. The most common baud rate of communication is 1200 at VHF frequencies.

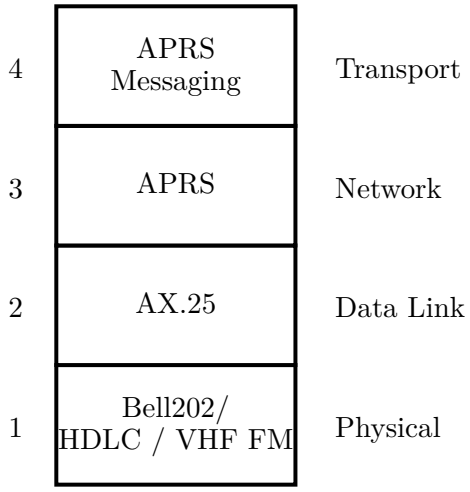


Figure 6.1: The OSI layers for a typical APRS application. The entire layer 1 is seen as "Amateur Bell 202". Source: [33, Figure 1]

6.4.2 Modulating Bell 202

To generate an analog signal using a digital microcontroller one needs to use a component called a Digital-to-analog converter (DAC). This is also what is used in a computer's soundcard to generate audio. Some microcontrollers have a DAC peripheral built into them, but in our case the ATmega 328P does not. It does however have a pulse-width-modulator (PWM) that can be used to generate an analog signal. Another option is to implement our own DAC using digital GPIO pins and a resistor ladder, as can be seen in figure 6.2. When the GPIO pins are logic high (5 volt), resistors R1 to R4 form a voltage divider with R5 to ground. When a GPIO pin is logic low (0 volt), it is in parallel to R5, changing the characteristics of the voltage divider, and thus changing the voltage on the input of C1. Setting a GPIO pin to high impedence (unconnected or set as input), a resistor is removed from the voltage divider. Using this technique multiple voltage levels can be generated by changing the state of the GPIO pins. A lookup table is calculated for mapping GPIO pin settings to voltage levels. If only sine waves need to be generated, this lookup table can be simplified to have the values for a sine wave's shape. The sampling rate for generating the sine wave however needs to be a multiple of the frequency of the sine wave for this simplification to work.

Generating sine waves of different frequencies can be done in two ways. A constant sample rate can be used, changing the voltage level for every sample, as found from the lookup table. Alternatively constant voltage levels are used for one sine cycle, changing the sample rate to change the frequency of the generated sine wave. In the latter case the lookup table can be simplified quite a bit, as only a limited number of voltage levels need to be known, saving program memory. Both of these methods were tested and functioned quite well.

The circuit and algorithm proposed in the WhereAVR project[34] was built and tested. The sine wave generated by this project, using constant voltage levels, but a variable sample rate, can be seen in figure 6.3. Measurement was done before the filter capacitor (C1 in figure 6.2). After this capacitor and the audio input circuit of the radio, the sine wave is smoothed out before it is transmitted.

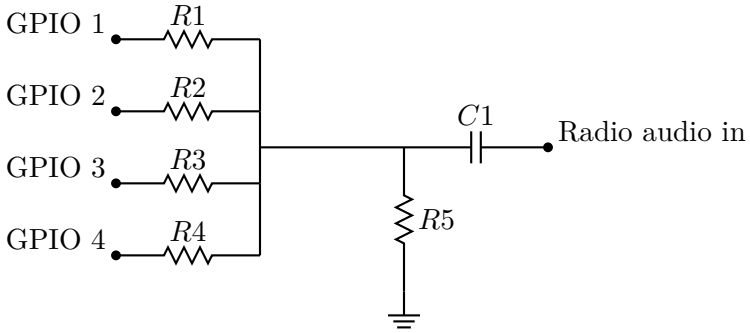


Figure 6.2: Circuit diagram of a resistor ladder used for digital to analog conversion.

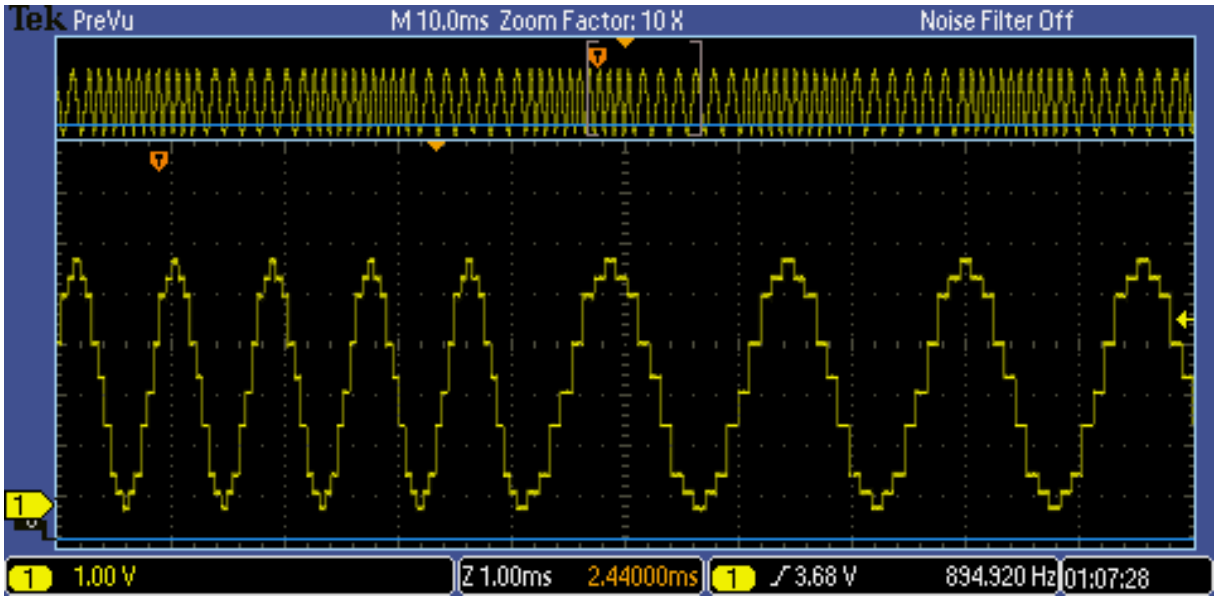


Figure 6.3: Output of the resistor ladder. The image shows the transition from a 2200Hz sine wave to a 1200Hz sine wave. Discreet voltage levels are noticeable which are changed at different rates for different generated frequencies.

6.4.3 Demodulating Bell 202

To demodulate the signal, we need to detect if the audio stream contains a 1200Hz tone, or a 2200Hz tone. There are mainly two ways of doing this in software on a microcontroller.

6.4.3.1 Zero crossing

The easier method of the two is counting the time between two zero crossings. As we know a sine wave crosses the x-axis twice in a cycle. We can calculate what the time difference between the two zero crossings should be for the two frequencies we need to differentiate between.

To detect zero crossings with a microcontroller, we use a peripheral called a comparator inside the microcontroller. The comparator compares an incoming analog signal to a reference voltage. Whenever the analog signal goes higher or lower than the reference voltage, an interrupt routine in the software is executed. The analog inputs can only handle voltages between 0 and 5 volt, so we shift the audio up and amplify it for maximum swing using an operation amplifier. The signal is then compared to a 2.5 volt reference voltage.

The time difference between two successive executions of the interrupt routine is calculated, and in that way the received frequency is determined. Any change in the received frequency will result in the reception of a digital 1, or in the case of no change a 0 will be registered.

Working at a baud rate of 1200, when receiving a 1200Hz signal a full sine wave cycle will be received. In this case we will be able to detect the frequency twice when timing zero crossings. In the case of the 2200Hz signal, it is 0.55 cycles per transmitted bit. We therefore have a transition between 1200Hz and 2200Hz at random times during a cycle of the sine wave. Zero crossings will therefore never be exactly the calculated time apart. Averaging and rounding comes into play when detecting the frequency. During tests with the WhereAVR project, it was seen that the chance of detecting the wrong frequency is quite high. This method of demodulating Bell 202 does work for shorter AX.25 frames, but due to the high chance of a mistake in frequency, the chance of getting a bit error is very high.

6.4.3.2 Filters

A better approach to demodulate AFSK is by using two narrow band-pass filters, one for each of the two frequencies we need to detect. Comparing the output value of the two filters, the filter with the highest output indicates that that specific frequency is being received.

This approach is the most commonly used way of demodulating Bell 202. All soundcard modem software that was inspected used this approach. To implement digital signal filters on a microcontroller is difficult due to the limited processing power that is available. The filters can be mathematically simplified to a point where it is indeed possible to implement on a microcontroller.

The BeRTOS operating system[35] for Atmel AVR's has an implementation of a filter-based Bell 202 demodulator and AX.25 receiver. The demodulator is written to use either a software Butterworth or a software Chebyshev filter. Both of these filers were simplified to be only approximations that still work. For example, in many cases a binary bit shift to the left, which is a division by 2, was used rather than a multiplication by 0.4379 or a multiplication by 0.668. Another example is a bit shift of 2 to the left (divide by 4), rather than a division by 6 or division by 3.55.

With these simplifications the demodulator can execute properly on an ATmega 328P, running at a clock frequency of 16MHz. The audio signal received from the radio is sampled at 9600Hz by an ADC. This translates to 8 samples per symbol, or bit, of the data signal. These

samples are fed through the approximated software filters which detects whether a transition in frequency occurred, or not. The resulting binary data stream (1 for transition, 0 for no transition) is decoded by the AX.25 decoder.

Tests performed with BeRTOS showed that the performance is quite impressive for a microcontroller, but it does consume all the resources available on the microcontroller.

6.4.4 Using a modem IC

Rather than demodulating the signal in software, dedicated hardware exists to perform this task. The MX614 from CML Microcircuits[36] is an integrated circuit that can perform the modulation and demodulation of Bell 202. It is however difficult to obtain and the only ones that could be found during research were (probably fake copies) available from Hong Kong, China. Tests with this integrated circuit were not able to receive or transmit data at all, but trade-offs made in the circuit due to time constraints is probably the reason.

Further research into the use of this integrated circuit is necessary. The overall power usage of the system can be improved by offloading the modulation and demodulation of the signal to a hardware component that was designed for the job.

A modem can be implemented using programmable hardware, like an FPGA. These devices are power intensive and very expensive. This makes them unideal for use in low powered and large scale applications.

6.5 Programming of Radiometrix radios

The Radiometrix SHX1 radio module that was chosen for the VHF communication is able to transmit at various frequencies. To set it up to transmit at our chosen frequency, a couple of register values need to be calculated and then programmed onto the radio. The radio has seven channels which can store a transmit frequency and a receive frequency. During usage a specific channel can be chosen.

For our experimentation the amateur radio frequencies meant for APRS and AX.25 were used[37]. Both the university's license (callsing ZS1SUN) and the author's license (callsign ZS1JPM) allow usage of these frequencies for experimental use, at power output levels of up to a few hundred Watt.

Per channel on the radio there are four registers to program: TXN, RXN, TXR and RXR. TXN and RXN hold a value that indicates the channel spacing for transmission and reception respectively. TXR and RXR hold a value indicating the transmit and receive frequencies, respectively. These register values are used by the PLL circuit to generate the correct carrier frequency.

The channel spacing register is calculated by the formula: $10MHz/R =$ channel spacing. The amateur radio band plans in South Africa[37] mostly use frequencies that are 25kHz apart. The datasheet for the Radiometrix SHX1 also use this value in its example calculation[32, p. 4].

We therefore keep to this value.

$$\begin{aligned} 10MHz \div R &= 25kHz \\ R &= 400 \\ \text{For all channels: } TXR &= 400 \\ \text{and } RXR &= 400 \end{aligned} \tag{6.5.1}$$

To calculate the R-register values one need to use the following two formulas according to the datasheet[32, p. 4]:

$$\text{For transmission: } N = \text{channel frequency} \div (10MHz \div R) \tag{6.5.2}$$

$$\text{For reception: } N = (\text{channel frequency} - 21.4MHz) \div (10MHz \div R) \tag{6.5.3}$$

The results of these calculations are shown in table 6.1 and the set of serial commands to send to the radio to program all these values are listed in listing 6.1.

In normal operation mode the channel can be chosen by three input pins to the radio. Another option is to choose the channel by a serial command. We opted for the latter as it needs to be done only once, and it will save three connections on the prototype. It might be a good idea to connect the channel select lines in a final design. It will also be good to leave the programming serial connection in place, but an extra UART peripheral is needed on the microcontroller. The ATmega 328P only has one UART which is used for communication with the computer in our case. Using a larger ATmega 2560 will solve the issue of too few UART interfaces and GPIO pins.

Channel	Frequency	TXN value	RXN value
0	144.525	5781	4925
1	144.550	5782	4926
2	144.575	5783	4927
3	144.625	5785	4929
4	144.650	5786	4930
5	144.675	5787	4931
6	144.700	5788	4932
7	144.800	5792	4936

Table 6.1: R-register values for all channels to set their frequencies.

6.6 Summary

Radios and microcontrollers which are suitable for wildlife monitoring networks were discussed in this chapter. Specific hardware components were chosen for use in this project. The techniques used to modulate and demodulate digital data on the radio link is described. Programming of radio frequencies into the radio module itself is also mentioned.

Next we will have a look at how these techniques will be integrated into working hardware modules.

Listing 6.1: Commands to send to the radio for programming of its registers.

```
SETSER          ; use the channel specified over serial
GOCHAN 0        ; go to channel 0
SETSER
LDTXN 0 5781    ; load channel 0 transmit N register
LDRXN 0 4925    ; load channel 0 receive N register
LDTXR 0 400     ; load channel 0 transmit R register
LDRXR 0 400     ; load channel 0 receive R register
LDTXN 1 5782
LDRXN 1 4926
LDTXR 1 400
LDRXR 1 400
LDTXN 2 5783
LDRXN 2 4927
LDTXR 2 400
LDRXR 2 400
LDTXN 3 5785
LDRXN 3 4929
LDTXR 3 400
LDRXR 3 400
LDTXN 4 5786
LDRXN 4 4930
LDTXR 4 400
LDRXR 4 400
LDTXN 5 5787
LDRXN 5 4931
LDTXR 5 400
LDRXR 5 400
LDTXN 6 5788
LDRXN 6 4932
LDTXR 6 400
LDRXR 6 400
LDTXN 7 5792
LDRXN 7 4936
LDTXR 7 400
LDRXR 7 400
GOCHAN 3        ; go to channel 3
```

§ 7. Hardware and software details

Two individual hardware modules have been designed and built. The first one is a sensor module with a UHF radio. The second is a module with a VHF radio that communicates using AX.25, which is modulated and demodulated in software.

All nodes in the network are made up of either one of these modules, or a combination of both. A base station was also built, consisting of a Linux computer.

The system layout can be seen in figure 4.2. This chapter describes the hardware and software specifications for every module listed in the network systems layout.

7.1 Sensor module

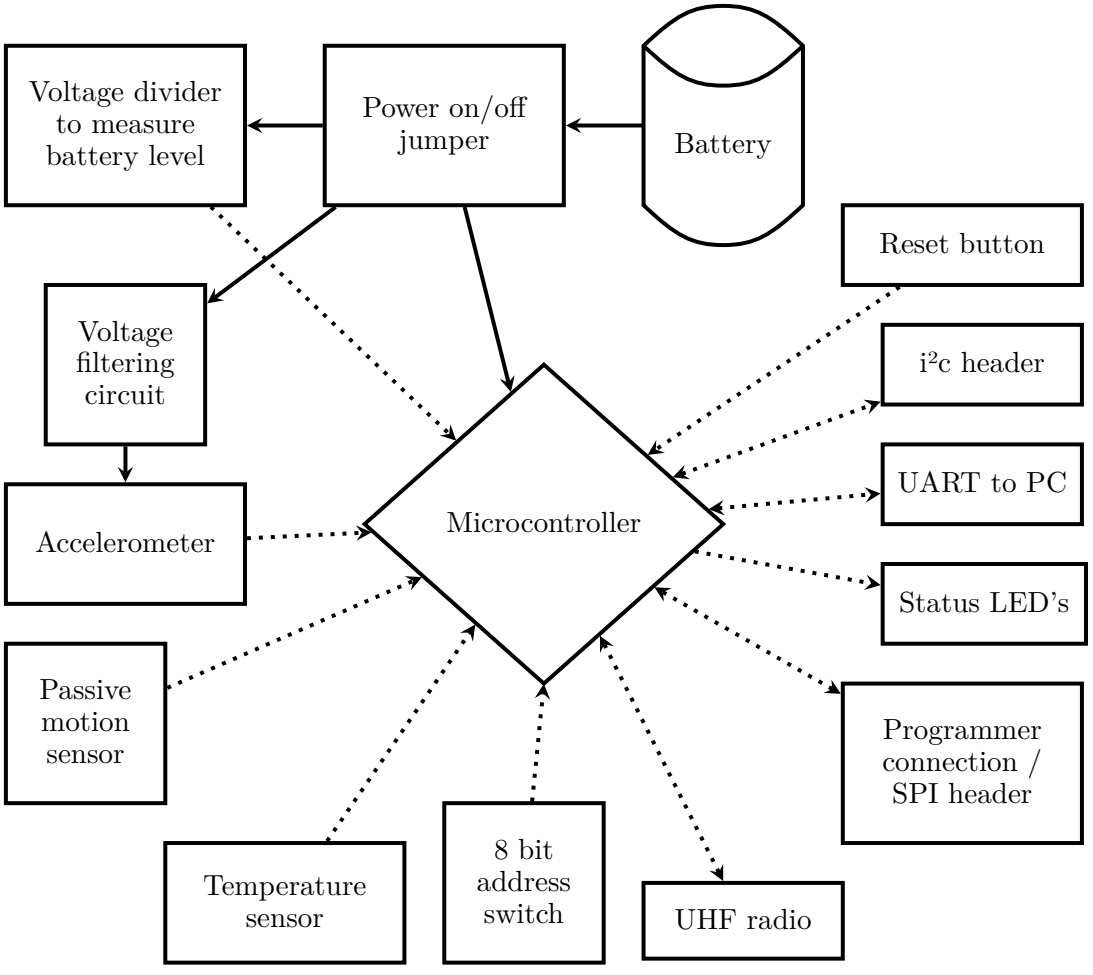


Figure 7.1: System components and layout of the sensor module

The logical layout of the sensor module can be seen in figure 7.1. All parts in this layout will be discussed below. A circuit diagram detailing the electrical layout can be seen in appendix D.

Microcontroller At the heart of our circuit we have the microcontroller. As was already discussed in chapter 6, the ATmega 328P microcontroller is used.

Battery All components on this module can operate around 3 volt. The minimum voltage for the Freescale Semiconductor MMA7455L accelerometer is 2.4V. For an Analog Devices ADXL345 accelerometer the minimum supply is 2V. For the HopeRF RFM12B radio the minimum voltage is 2.2V. All other components are independent of the supply voltage. We can thus safely power this system with two Duracell AA alkaline batteries, operating at 1.5V each[38]. In series it will provide 3V.

Power on/off jumper To be able to switch the sensor on and off without removing the battery, a jumper is added in series with the batteries. For permanent installations the jumper can be omitted and replaced by a wire that is soldered in.

Voltage divider To determine the level of the battery, a connection is made from the battery to an ADC input of the microcontroller. The microcontroller has a built in 1.1 volt reference voltage that is accurate as long as the microcontroller is powered up above 1.8V. We need to measure the battery voltage relative to this built-in reference, and not relative to the input voltage, otherwise a decline in the battery voltage will not be noted. The reference voltage is the maximum value the ADC can measure. We therefore need to divide the battery voltage down to within the 0-1.1V range to be measurable.

Reset button This is necesary to tell the microcontroller to start its program from the beginning.

i^2c header The accelerometer and temperature sensor are both connected to the microcontroller via i^2c (Inter-Integrated Circuit) serial communication lines. This header is included for sniffing of these lines during debugging. More sensors can also be connected later using this header.

UART to PC The UART serial lines are brought out on a header for communication with a computer during development and testing.

Status LEDs These are useful for debugging a program on a microcontroller. A certain point in the code can switch the LED on or off to indicate a certain state.

Programmer connection / SPI header The ATmega range of AVR microcontrollers are programmed using a protocol called ISP (in system programming). It is similar to SPI (serial peripheral interface), and the two also use the same pins on the microcontroller. For the header to be able to program the microcontroller, we also need to connect the supply voltage, ground and the reset pin of the microcontroller to the header. As the SPI lines are connected to this header, it provides us with an ideal place to sniff SPI communication. The radio is connected to the microcontroller via SPI lines, which means the SPI header is used during debugging and testing of the radio.

UHF radio The RFM12B UHF radio module is included for short range, low power wireless communication.

8 bit address switch Eight DIP switches are included for various settings, mainly specifying the address of this sensor. A unique address for every sensor is important to identify where messages originate in the wireless network.

Temperature sensor A temperature sensor may be useful to measure the temperature of an animal.

Passive motion sensor When all the other sensors are switched off and the microcontroller is in sleep mode, any movement will cause this sensor to switch and wake the microcontroller up with a hardware interrupt. This method has already been discussed in section 3.3.

Accelerometer Two accelerometers exist that can be used in this circuit, because they are electrically and physically compatible. The software does need to be changed depending on which one is used. The MMA7455L from Freescale Semiconductor is the cheaper option (R60), providing measurements of acceleration with resolution of 10 bits in a range of either $\pm 2g$, $\pm 4g$ or $\pm 8g$. The other option is an ADXL345 from Analog Devices. It is more expensive (R86 for IC, R648 for development board), has a lower power usage and provides samples with 13 bits of resolution at $\pm 16g$.

Voltage filter circuit The accelerometer has a dedicated filter for the supply voltage. From experimentation is was seen that this sensor is very prone to noise on its measurements. Filtering its supply voltage helped a lot to prevent this noise.

Due to the versatility of this sensor module, its software can be configured to allow different modes of operation. A list of possibilities are:

- Wake up by passive motion sensor. Sample the accelerometer for a short period. Transmit an alarm message when a certain level of movement is exceeded. This is useful when the sensor module is used as a motion alarm.
- Reset a time-out-timer every time movement is detected by the passive motion sensor. If the time-out-timer expires, an alarm message is sent. This is useful to detect when an animal is not moving.
- Periodic temperature measurements can be done, and whenever the temperature is below or above certain levels, an alarm message can be transmitted.
- The accelerometer can be sampled periodically to determine the behaviour of the animal. On board the ATmega 328P is 1 kilobyte of EEPROM memory which can be read and written by the software. This can be used to log samples, which can then be transmitted when the memory is full.

Likely more combinations of the usage of the sensors and available hardware exist. Software was implemented to prove the working of every component.

A photo of a sensor module can be seen in figure 7.2. A double AA battery pack is included for size reference purposes. The radio and antenna can be seen on the right hand side of the PC board, and the microcontroller, 8-bit DIP switch, a button and the power jumper can be

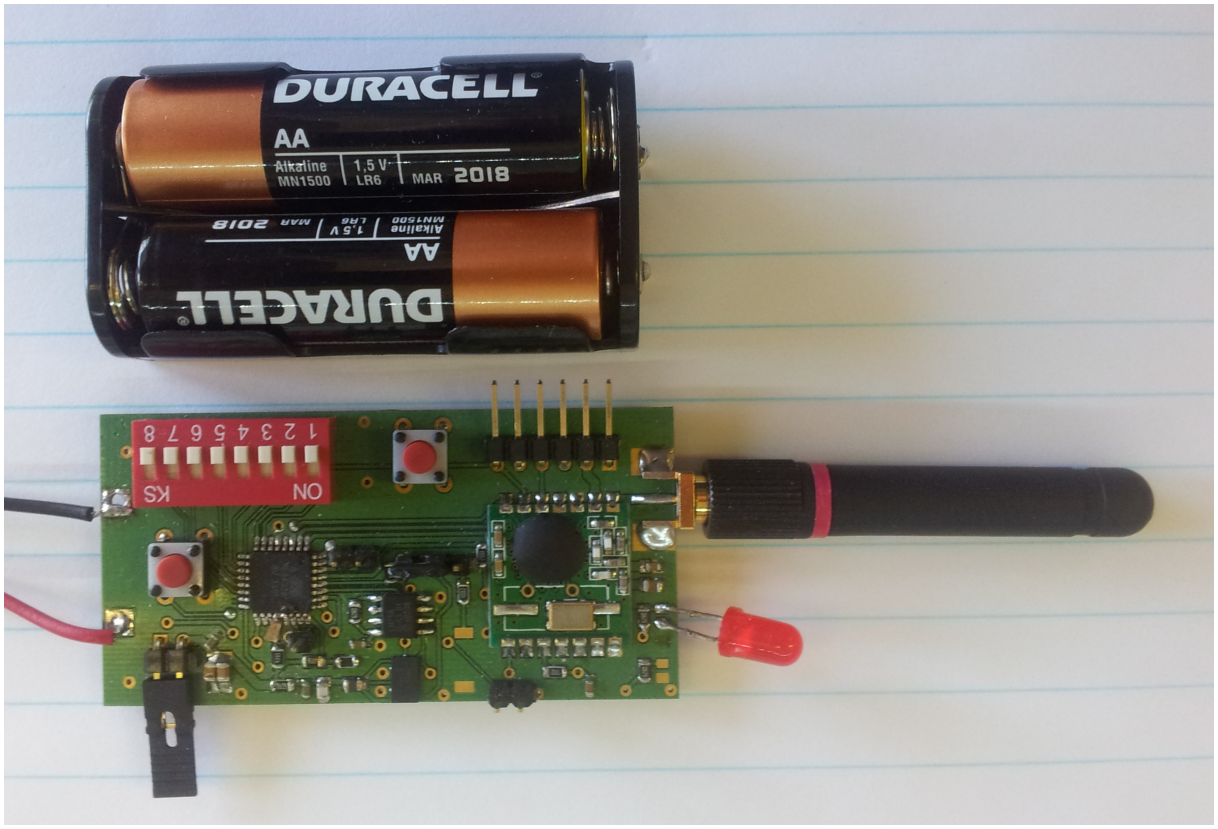


Figure 7.2: A photo of the sensor module.

seen on the left. At the top right is the 6 pin programming header that doubles as the SPI sniffing header. In the middle at the bottom of the PCB is the temperature sensor IC and the accelerometer IC. Right of them are two rectangular copper pads where the passive motion sensor should be soldered onto.

7.2 VHF radio module

Modulating and demodulating AX.25 (and Bell 202) consumes almost all the resources available on the ATmega 328P microcontroller. The decision was therefore made to have a dedicated module only handling the VHF communication. This module can be used on its own as a repeater node in the network. For a bridge node, a sensor module and VHF module can be connected together, offloading much of the logical processing to the sensor node, leaving the VHF module free to handle only the VHF communication. As a bridge node will consist of two separate microcontrollers and circuits, the power consumption will be increased. The VHF radio part can however mostly be powered down, saving energy, and only be switched on when VHF communication needs to be done.

In figure 7.3 the components and layout of a VHF module can be seen. This layout and its parts will now be discussed. A circuit diagram detailing the electrical layout can be seen in appendix D. Figure 7.4 shows a VHF module configured to be a repeater, complete with a sealed lead acid battery, voltage regulating circuit and helical whip antenna connected. This is the second prototype and was constructed on Veroboard. The first prototype was built on a breadboard and a PCB was manufactured for the third prototype.

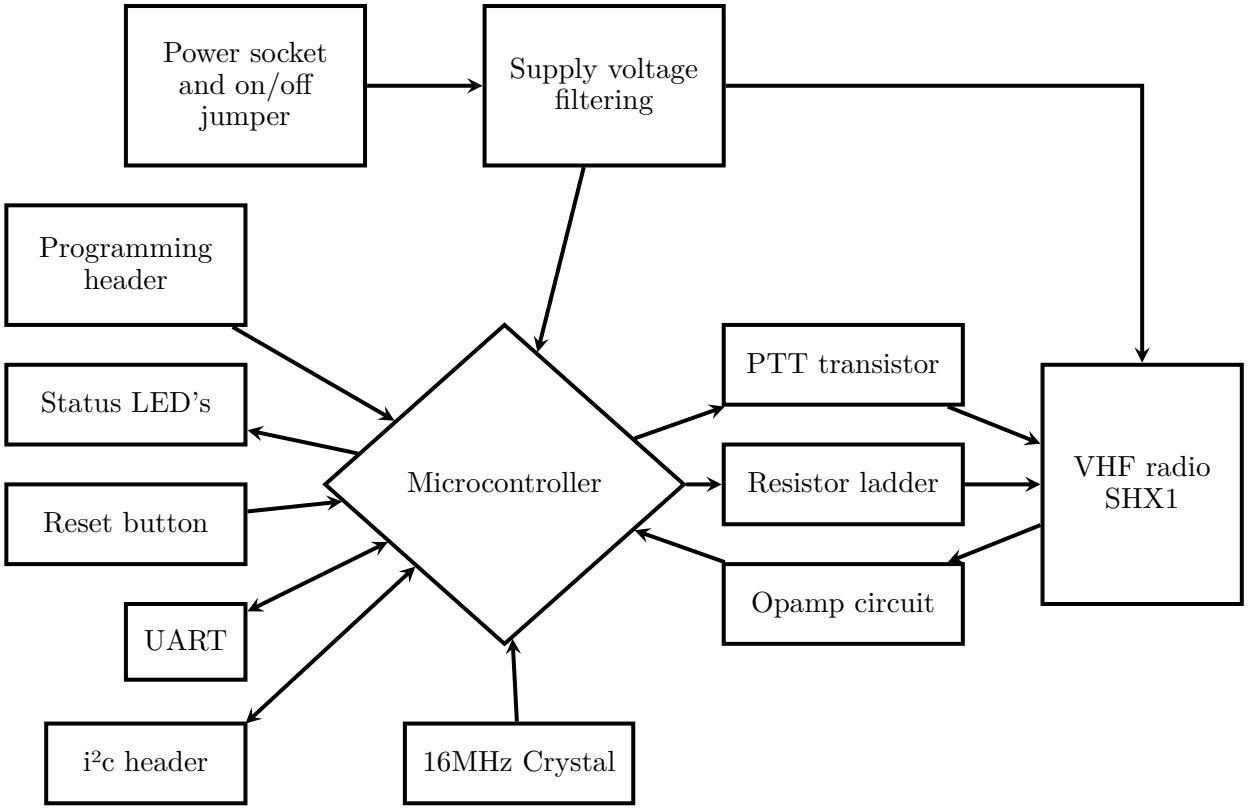


Figure 7.3: System components and layout of the VHF radio module

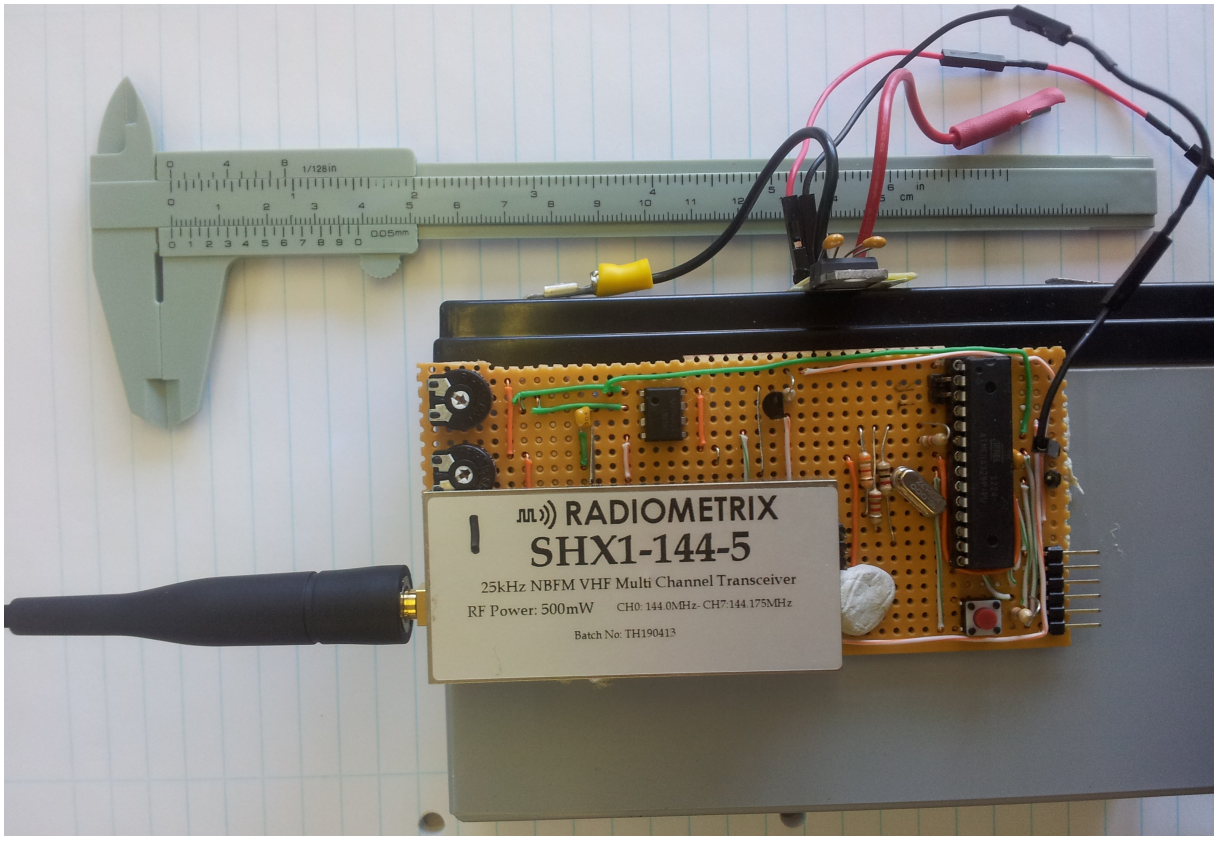


Figure 7.4: The second prototype VHF module.

Microcontroller At the heart of this hardware module is an ATmega 328P microcontroller. It handles modulation and demodulation of the signal, as well as controlling of the radio.

Power socket and on/off jumper This hardware was designed to be powered by any 5 volt supply. A socket is provided where the correct voltage can be connected. An on/off jumper allows the switching on and off of the module at any time.

Supply voltage filtering As the circuit contains a radio, the chance of RF feeding back onto the voltage supply rails are quite high. Decoupling capacitors are included at carefully chosen locations, decoupling any RF before it can reach the digital circuitry.

VHF radio A Radiometrix SHX1 is included for VHF communication. This is a dumb radio and needs to be controlled on the lowest level by the microcontroller.

PTT transistor The radio has an input line to switch between receive and transmit mode. To transmit, this line needs to be shorted to ground.

Resistor ladder This part is used to generate an audio signal by the microcontroller. The principle behind it was already discussed in section 6.4.2.

Opamp circuit The received audio output from the radio needs to be amplified and shifted to swing around a DC offset. This is so that the microcontroller's ADC input can sample the signal at maximum resolution. An operation amplifier circuit is used to do the amplification and shifting of the signal.

16MHz crystal The modulator and demodulator software that runs on the microcontroller was written and optimised for when the microcontroller runs at a clock frequency of 16MHz. Internally the ATmega 328P can generate a clock of 8MHz, which can only be divided, not mixed up. We therefore need to implement a different source for the CPU's clock, in the form of a quartz crystal oscillator. A quartz crystal also provides a more accurate clock frequency than the ATmega's internal oscillator.

i^2c header This component is unnecessary, but as a PC board is manufactured and further development may need to use i^2c communication for extra sensors, it is always a good idea to include these headers.

UART The UART serial communication port is used for communication with a computer during debugging. When a sensor module is connected to this VHF module, the two modules will also communicate via this UART.

Reset button It is always a good design decision to include a reset button to tell the microcontroller to start its program from the start.

Status LED's These can be used to indicate in what state the module is. One can say it is in transmit mode, receive mode or indicate an error or success (like a received acknowledgement).

Programming header This header is needed to load the software we have written for the module onto the microcontroller. The header also doubles as a SPI sniffing header. SPI is not used in this circuit, but can be useful for adding extra components or sensors.

The software that runs on this VHF module is based on the BeRTOS operating system[35]. This operating system has a very good implementation of AX.25 modulation and demodulation using an AVR microcontroller. Depending on the use case, extra features are implemented to either relay messages, or transmit messages and listening for an acknowledgement.

If a VHF module and a sensor module are connected to each other, it is a good idea to be able to switch the entire VHF module off to save power when it is not needed. This can be done by using a relay on the power on/off header of the VHF module, which is switched by a GPIO output pin on the sensor module. The circuit diagram for this relay-based switch can be seen in figure 7.5.

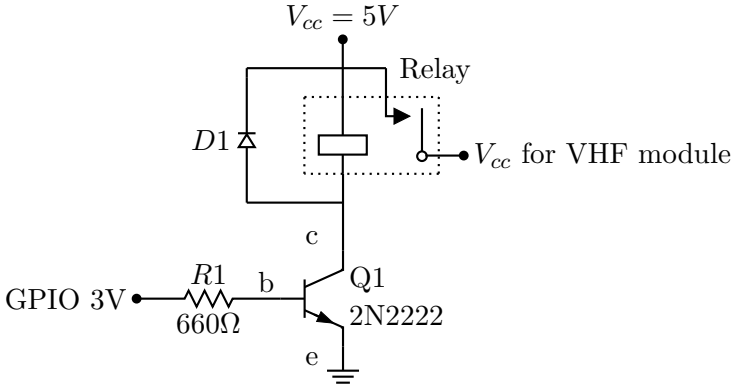


Figure 7.5: A circuit to switch the VHF module on and off with a relay.

Resistor R1 is meant to limit the current flowing out of the GPIO pin and thus protecting the microcontroller against overcurrent. When the GPIO is at 3V, the transistor (Q1) will switch on in saturation mode and allow current to flow through the relay's primary coil. The relay's secondary will switch on, providing current to the VHF module. When the transistor switches off, the current supplying the relay's coil, which is an inductive load, suddenly disappears. A voltage spike is induced in the opposite direction by this inductive load. To prevent damage to the rest of the circuit a clamp diode (D1) is installed in parallel to the coil, shorting any reverse currents, including the voltage spike. This is not a very low power circuit during operation, as a constant loss current flows through the base of the transistor. It however only happens during operating, which is comparatively short periods compared to the times the circuit is powered off, without any losses.

When the sensor module that is powered by 3 volt, is connected to the VHF module which is powered by 5 volt, communication between the two using a UART is not directly possible due to a difference in logic levels. We need to include a simple voltage level shifter on the two UART communication lines so that the two modules can communicate. The circuit diagram for this level shifter is seen in figure 7.6. This circuit is however not power efficient as one of the two transistors will always be in saturation mode, causing a current of 1mA to flow. Higher value resistors for R2 and R3 will help keep this current down. If the resistors are too big, transistor

Q2 will not be drawn into saturation mode. Output load on the 5V output side should also be kept in mind during calculations. Low power integrated circuits exist on the market that can also be used for power efficient level shifting.

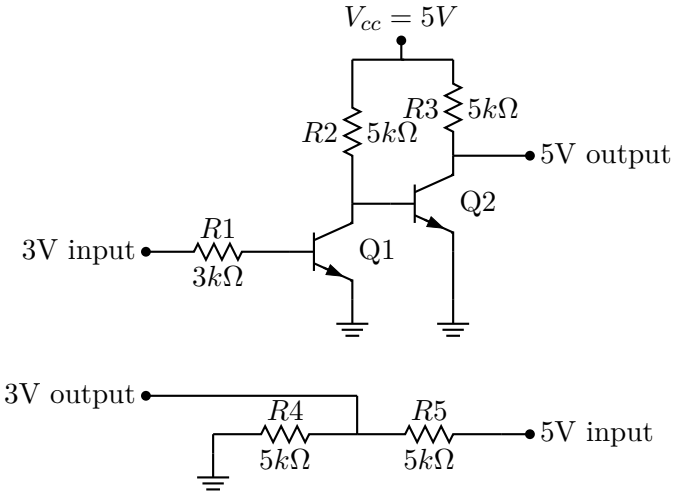


Figure 7.6: The UART level shifting circuit.

The operational amplifier circuit used to amplify and shift the DC offset of the received audio signal can be seen in figure 7.7. Rather than using fixed value resistors two variable resistors, R2 and R3, are included to set the gain of the amplifier (R3) and the DC offset of the output (R2). This is a better design choice, than using fixed value resistors, as the output of every radio is slightly different. Using a test transmission the opamp circuit can be calibrated to have the best output towards the ADC. A better input signal for the ADC will improve the performance of the software demodulator on the microcontroller.

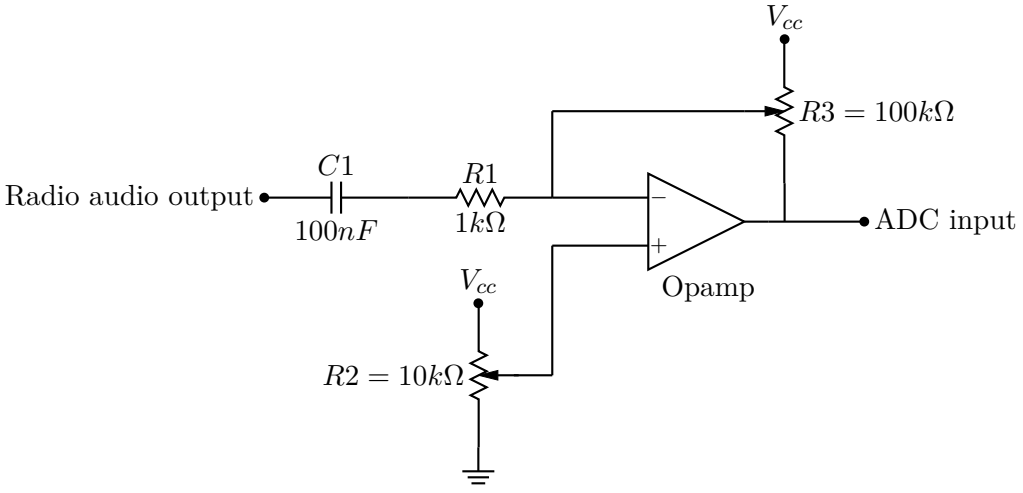


Figure 7.7: The operational amplifier circuit.

For switching between receive mode and transmit mode, one of the radio's lines needs to be shorted to ground. It is possible to do this by connecting the line to an output pin on the microcontroller and pulling the pin to ground. We however do not know how much current there will flow as it is not defined by the radio's datasheet. In receive mode the line needs to float

(high). We can also not pull it high as it is not defined in the datasheet to be high for receive mode, but rather just being left open circuit. The correct way is to short the line to ground during transmit mode using a transistor. We can switch the transistor on with an output pin on the microcontroller, allowing current to flow from the collector to the emitter, giving the impression of a short. This "short" pulls the transmit enable line to ground. When we want to receive, the microcontroller's output pin is low, switching the transistor off, and breaking the "short" between the collector and emitter. The transmit enable line will now float, causing the radio to be in receive mode. A BJT is sufficient for this circuit, but a resistor is needed between its base and the microcontroller's output pin to limit the current flowing out of the microcontroller. Too much current and the microcontroller can be damaged. The diagram for this circuit part can be seen in figure 7.8. "GPIO" is the output pin from the microcontroller, switching to 5 volt for transmit mode, and 0 volt for receive mode. At the opposite end is the radio's transmit enable line, being shorted to ground for transmit mode, and left floating for receive mode. The value of R1 was calculated so that the current flowing from the GPIO pin will be at least a tenth or less of the maximum rating for a GPIO output pin on the ATmega 328P. The current is still high enough to drive the 2N2222 BJT into saturation mode.

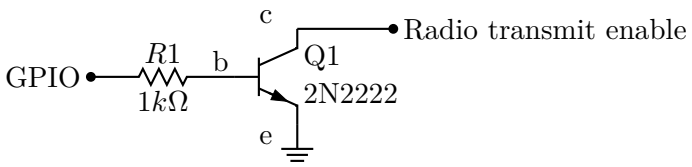


Figure 7.8: The transmit enable circuit with a current limiting resistor.

7.3 Internet gateway

The internet gateway, or base station, is the module that will be located at a central point in the network. We assume this central point is a high site on a mountain, providing very good radio coverage, already has an electricity supply, and has some kind of internet connection. In the case of no internet connection, we assume the site has cellphone coverage, providing us with the possibility to use the cellphone network for internet access.

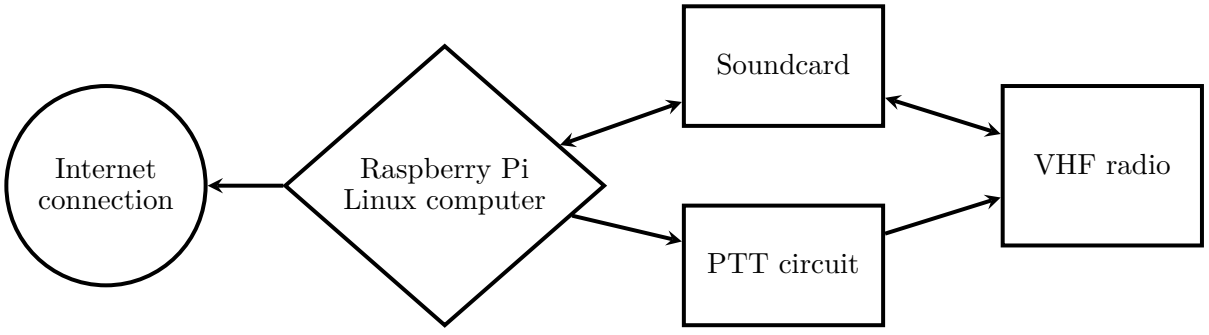


Figure 7.9: Layout of the components used for the base station.

Figure 7.9 shows the logical layout of the components used for the base station. The specifications of every one of these components are as follows:

VHF radio Any two-way FM radio operating at the correct frequency in the VHF band should work here. The most radios on the market provide a data connector, or audio in and out connectors for the purpose of digital communication. During testing a Radiometrix SHX1 module was used. This is the same radio that is used for the embedded VHF module.

Linux computer Any low cost or low power computer should work. A Raspberry Pi was chosen as it is very small, low powered and cheap.

Soundcard A soundcard is used as an ADC and DAC to sample the audio to and from the radio. In the case of the Raspberry Pi, it has no audio input on board the computer. An additional cheap (R60) USB soundcard was used.

PTT circuit The radio needs to be told when to transmit, or when to receive. The computer can do this either with one of the RS232 control lines, or in the case of the Raspberry Pi with a GPIO output pin. A circuit is built to protect the GPIO pin and shift the logic levels to the correct voltages. An extra optional feature of this circuit can be to limit the total time the radio can be put in transmit mode. In case of a software issue that keeps the radio in transmit mode, this circuit will then time out after a certain period of time. An example circuit provided in the manual of the soundcard modem software, DireWolf[39], uses a 555-timer for this purpose.

Internet connection For relaying the received data to a central point, a gateway (base station) needs to have an internet connection. This can be either via the cellphone network, satellite, WiFi or terrestrial data radios.

The software that will be running on the Raspberry Pi is the Raspbian operating system, and a soundcard modem called DireWolf[39]. Processing of the data is done using simple python scripts. An example of such a script, as has been used on a previous project, can be seen at `https://github.com/jpmeijers/picontroller`. Messages that are received by the base station are acknowledged and forwarded to a database server online.

As the base station needs to operate 24/7 in a remote area, steps need to be taken to ensure that software and hardware faults be recovered or handled elegantly. An example of this is the time-out timer on the PTT line. Scheduled tasks also need to be set up to check if all the necessary software services are still running, otherwise restarting them.

High sites on mountains normally have an antenna mast, providing us with the possibility of attaching an antenna high above the ground. With the increase in height there is an increase in the chance of lightning strikes. A commercial antenna that is DC-grounded, and has a good omnidirectional azimuth, but a rather large elevation gain to the horizon, would work very well. The SXC-1506 from Communication Electronics (`http://www.starantenna.com/model-detail.asp?RecID=14&Image=yes`) seems like a good choice. It has a gain of 6dBi with a beam width of 20° towards the horizon. It is also rated to withstand a wind speed of 200km/h, which is important for installation on a high site.

7.4 AX.25 frame and APRS message format

7.4.1 AX.25

Field	Flag	Destination Address	Source Address	Digipeater Address (0-8)	Control Field	Proto-col ID	Payload	FCS	Flag
Number of bytes	1	7	7	0-56	1	1	1-256	2	1

Table 7.1: Format of the AX.25 frame.

The composition of an AX.25 frame[16] can be seen in table 7.1. It has a minimum length of 21 bytes and a maximum length of 332 bytes. The frame is transmitted from left to right as it is shown in table 7.1. The least significant bit of each byte is transmitted first. The purpose of each field in the frame is as follows:

Flag The "frame start" flag and "frame end" flag are one byte each. It has the value 0x7E and has the purpose of separating frames on the air.

Destination address The address can be 6 alphanumeric ASCII characters, plus a 7th byte representing a number between 0 and 15 called the SSID (Secondary Station Identifier). The rest of the bits in the last byte indicate if this address is the last address in the header or not.

Source address The address of the sender. Again 6 alphanumeric characters plus one byte to indicate a number between 0 and 15.

Digipeater address The frame has a variable length header due to a variable number of repeater addresses that can be specified. These repeater addresses indicate which repeater should forward the frame next, and thus is an indication of what path the frame should take in the network. Up to 8 repeater addresses can be stored, allowing us to have networks of up to 9 hops. Due to the variable number of addresses, we need a way to indicate when an address is the last address in the header. The last byte in an address is used for this purpose. When the last byte in the address ends in a 1 it is the last address. Otherwise, if that byte ends in a 0 the next 7 bytes that are transmitted is also an address. This rule applies to all addresses, including the destination and source address.

Control field This is set to 0x03 to indicate that we are using UI-frames. It stands for Unnumbered Information, and means we are not using AX.25's built in link control.

Protocol ID This is set to 0xF0, which means "No layer 3 protocol implemented". We are using a custom protocol (APRS) inside the frame which the AX.25 decoder does not understand. It should just pass the frame on to the higher protocol layer.

Payload The payload can be anything between 1 and 256 bytes long.

FCS The frame check sequence is a 16-bit CRC checksum performed on the frame and attached to the end of the frame for error detection purposes.

7.4.2 APRS

APRS makes use of the address fields and the payload field of the AX.25 frame.

Destination Address In APRS the destination address is not used to indicate the receiving station. It is rather used for: indicating what type of station you are, what symbol should be used when you are drawn on a map, software version number, position data and generic repeater paths. The generic repeater path is indicated by the SSID and can be looked up from a table in the APRS specification[16]. It is mostly a keyword to forward the message a certain number of hops.

Source Address The first 6 alphanumeric characters indicate the address of the source of the message. The SSID may indicate an icon that should be used to display this station on a map.

Digipeater Address APRS redefines these addresses to indicate how many hops the message is allowed to propagate through the network. A repeater address of WIDE7-7 means the message can be repeated 7 times, each time decrementing the SSID. We are left with WIDE7-0, which means the message will not be repeated again. Up to 8 repeater addresses are allowed, giving us the possibility of relaying a message up to 56 hops through the network. Every repeater keeps a copy of the message for a short period of time to match receiving messages against and prevent repeating it again. Loops in the networks are prevented in this way.

Payload Depending on what type of data is being sent, this field's format is different.

APRS provides many different types of payload possibilities, like telemetry data, weather station data, position data. For testing the decision was made to use the "Message" data type, as it is the most versatile and unrestricted APRS data format. A list of all data types and their formats can be seen in the protocol specification[16].

	:	Addressee	:	Message Text (max 67 chars)	{	Message Number
Number of bytes:	1	9	1	0-67	1	1-5

Table 7.2: Format of the APRS message data type.

The format of an APRS message data type payload is indicated in table 7.2. This message will be inserted into the AX.25 frame's payload field. The required fields for both acknowledged and not-acknowledged messages are as follows:

: The literal ASCII colon character. It indicates that we are using the Message data type as payload.

Addressee The destination address of this message. It must be 9 characters long and should be padded with spaces if it is shorter. No other requirements are specified for this field. Normally it will be an amateur radio callsign with a SSID (for example ZS1JPM-15).

: Another literal ASCII colon character indicating the end of the address field and the start of the message text.

Message text Up to 67 ASCII characters, except for `/`, `~` and `{` are allowed.

Optional fields which are only used when a message needs to be acknowledged:

{ A literal ASCII open brace character, indicating the start of the message identifier.

Message number Up to 5 alphanumeric characters (no spaces allowed) can be provided as a unique identifier for this message.

If a frame contains a message number it needs to be acknowledged by the destination station. If the sender does not receive an acknowledgement within a specified timeout time, the message needs to be sent again.

7.4.3 Using the APRS message for wildlife tracking

For a source address we have 6 alphanumeric characters. This provides us with $(26 + 10)^6 \approx 2 \times 10^9$ (2 000 million) combinations. It can be made more human readable by making the first three letters an abbreviation of the type of animal, followed by three numbers to identify the animal. Due to geographical isolation between tagged animals, addresses can be re-used. Just a few examples of addresses are given below to give an idea of what is possible. More complete source addresses can always be embedded inside the 67 byte message text.

`COW123` The 123rd cow with a tracker on it.

`CAR###` Caracal

`CHT###` Cheetah

`ELE###` Elephant

`FOX###` Fox

`GRF###` Giraffe

`JAC###` Jackal

`LEO###` Leopard

`LIO###` Lion

`RIN###` Rhino

`ZEB###` Zebra

For the purpose of a destination address we have 9 ASCII characters. We can easily fit the word `BROADCAST` into that. Any other word like anycast or wildlife will also fit easily into the available space.

The 67 bytes of message text can contain sensor readings, battery voltage, location and a unique sensor identification number. It is also possible to encode this message text to prevent unauthorised users to have access to the data.

A message containing sensor values will be sent out by the animal borne sensor over a UHF link. It will be received by a UHF-to-VHF bridge where it will be inserted into an APRS message. Location data and more sensor values can be added here. The message is then broadcasted, received by one or more repeater stations, which will flood route the message up to 56 hops away. During this process a base station is likely going to receive the message. It will be acknowledged and the acknowledgement will be flood routed back to the original sender.

7.5 Summary

This chapter detailed the design and implementation of:

- A sensor module for low power data acquisition.
- A VHF communications module for use as a bridge between UHF and VHF communication frequencies, or as a repeater node in the VHF network.
- The internet gateway, or base station, providing uplink capabilities from the VHF network to the internet.
- The AX.25 data link protocol and the APRS network and transport layer protocol. An addressing scheme for wildlife monitoring is proposed.

In the next chapter tests are done on these hardware modules to prove their functionality and indicate how well they perform.

§ 8. Tests and Evaluation

In radio communication networks there are two parameters that are important. They are latency and bit error rate. This chapter tests the designed hardware modules by measuring both the latency and bit error rate. All combinations of radio links that can be made with the VHF capable hardware are checked.

The second set of tests are about radio coverage in an area that is representative of an area where a wildlife monitoring system will be used. Measured results are compared to simulations and an adjustment figure to the simulation is found for compensation due to clutter.

At the end of the chapter the power usage of the developed hardware is tested. Calculations are made to predict the average lifetime of the batteries. Proposals regarding the minimum sizes of photovoltaic cells are supplied.

8.1 Latency and error rate laboratory tests

Latency is measured for single hop links only. Two different receiver and transmitter pairs are tested. They are firstly the embedded microcontroller and secondly a computer running soundcard modem software.

8.1.1 Theoretical estimation of latency

The latency is made up of mainly two delays. It takes time to send the data over the air at a certain baud rate, it is accumulated in a buffer, and when transmission is done the data needs to be sent over a serial link to the computer for logging and timestamping. Some more, but shorter, delays are added by the software doing the demodulation of the signal and processing of the data.

8.1.1.1 Transmission delay

On air communication is done at 1200 baud. The Bell 202 standard we use only communicates one bit per symbol. Every bit in our message therefore takes the period of 1200 baud to transmit.

Message size 222 bytes

CRC check 2 bytes

Start and stop flags 1 byte each, total 2 bytes

Total frame size $(222 + 2 + 2) = 226$ bytes

Transmit delay 226 bytes $\times$ 8 bits $\times \frac{1 \text{ second}}{1200 \text{ baud}} = 1.5067s$

The frame is wrapped in a preamble and a postamble, containing the "frame start" and "frame stop" flags. These are added by the modem. On the receiving side the entire preamble is received causing an extra delay. The frame is processed immediately when the "frame stop" flag is received, so the entire postamble does not cause a delay. Because the start and stop flags are part of the pre- and postamble, these two bytes should not be counted if the pre- and postamble delays are used in the calculation.

Preamble delay 300ms

Postamble delay 50ms

The transmission delay is recalculated taking this into account.

Total frame size $(222 + 2 + 1) = 225$ bytes (without the start flag, but with the stop flag)

Transmit delay $(225 \text{ bytes} \times 8 \text{ bits} \times \frac{1 \text{ second}}{1200 \text{ baud}}) + 300ms = 1.80s$

8.1.1.2 Transfer of data over serial link to computer

The received frame is transferred to the computer using the KISS protocol. The message contains the raw data frame, without the start and stop flags and without the CRC check bytes. The KISS protocol delimiters are however included before and after the frame, as well as a data type identifier. This totals up to an extra 3 bytes. For RS232 and UART serial communication there is an extra start and stop bit for every byte, so we transfer 10 bits for every byte of data.

Total frame size $222 + 3 = 225$ bytes

Baud rate of serial link 19200

Number of transferred bits per byte in the frame 10

Transfer delay $225 \text{ bytes} \times 10 \text{ bits} \times \frac{1 \text{ second}}{19200 \text{ baud}} = 117.19ms$

8.1.2 Embedded transmitter, Embedded receiver

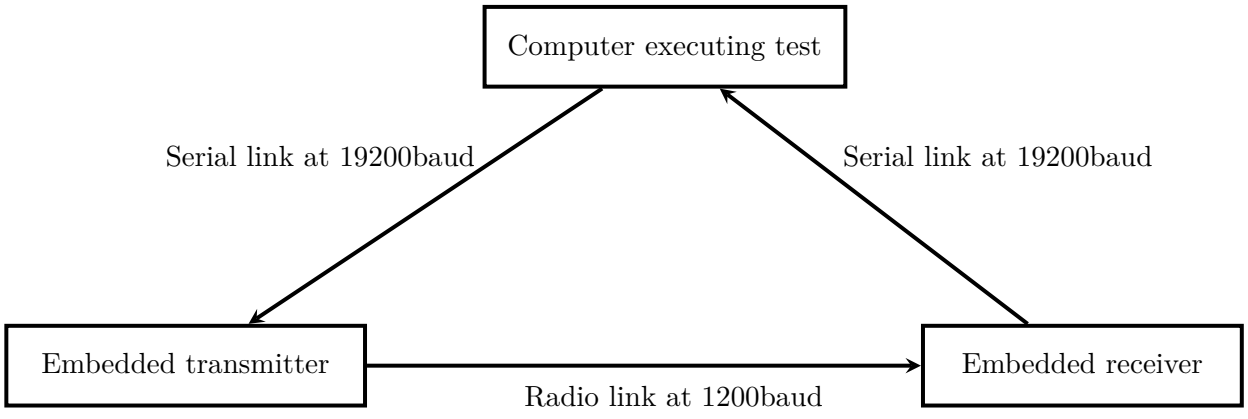
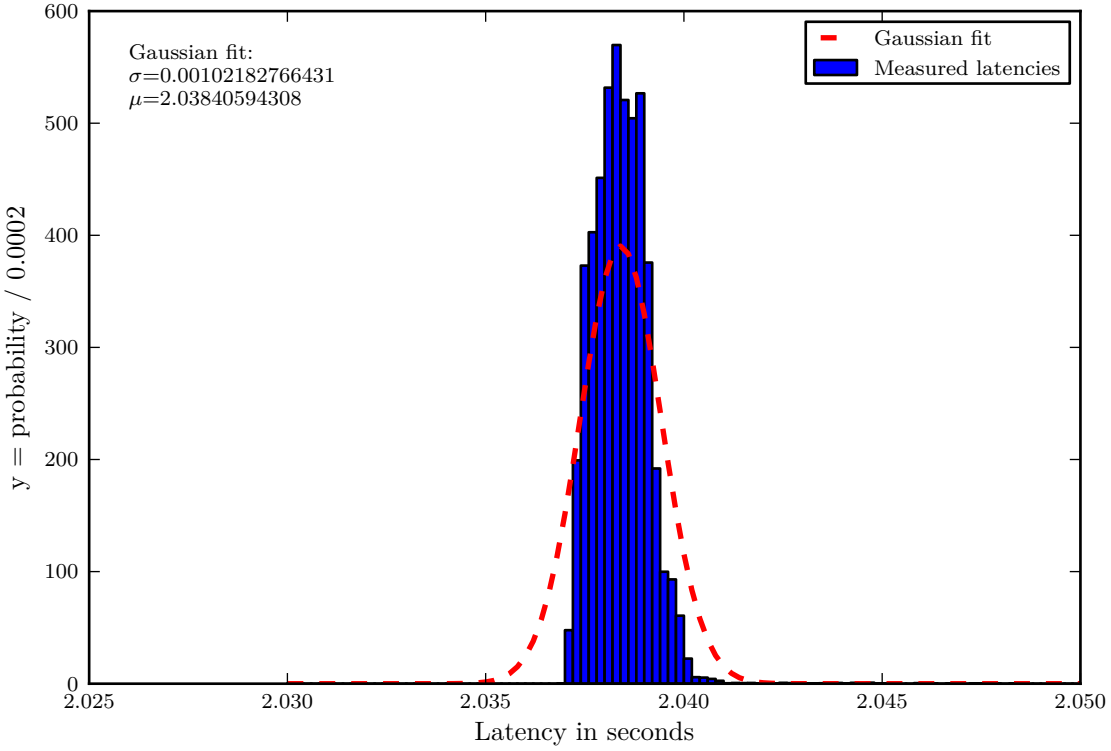


Figure 8.1: Layout of test between two embedded transceivers.

The latency between an embedded transmitter and an embedded receiver includes the transmission delay and twice the serial transfer delay. In other words it is the sum of the propagation delay via every link indicated by an arrow in figure 8.1. This theoretically equates to $1.80 + 0.11719 + 0.11719 = 2.034s$. Demodulation and processing of the signal and data take a short but variable amount of time, causing the small jitter that can be seen in the histogram, graph 8.1. The jitter can be numerically expressed as the standard deviation (σ) of the Gaussian fit, whereas the expected value of the latency is expressed by μ. As can be seen the measured $\mu = 2.0384s$ is quite close to the theoretical calculated $2.0340s$. The difference is about 4.3σ, which is the extra varying delays due to demodulation and processing.

Graph 8.1: Histogram of measured latencies between two embedded transceivers.



During the $55\frac{1}{2}$ hours this test ran, a bit error rate (BER) of 9.8×10^{-3} was measured. Although the test was done under laboratory conditions, there are a couple of reasons why the BER seems higher than expected.

- The radios did not radiate into proper antennas.
- Interference from external sources could have had an influence on the measurements.
- Reception on the embedded microcontroller use the built in ADC above its rated specification.
- Demodulation on the embedded microcontroller uses an approximated software filter.

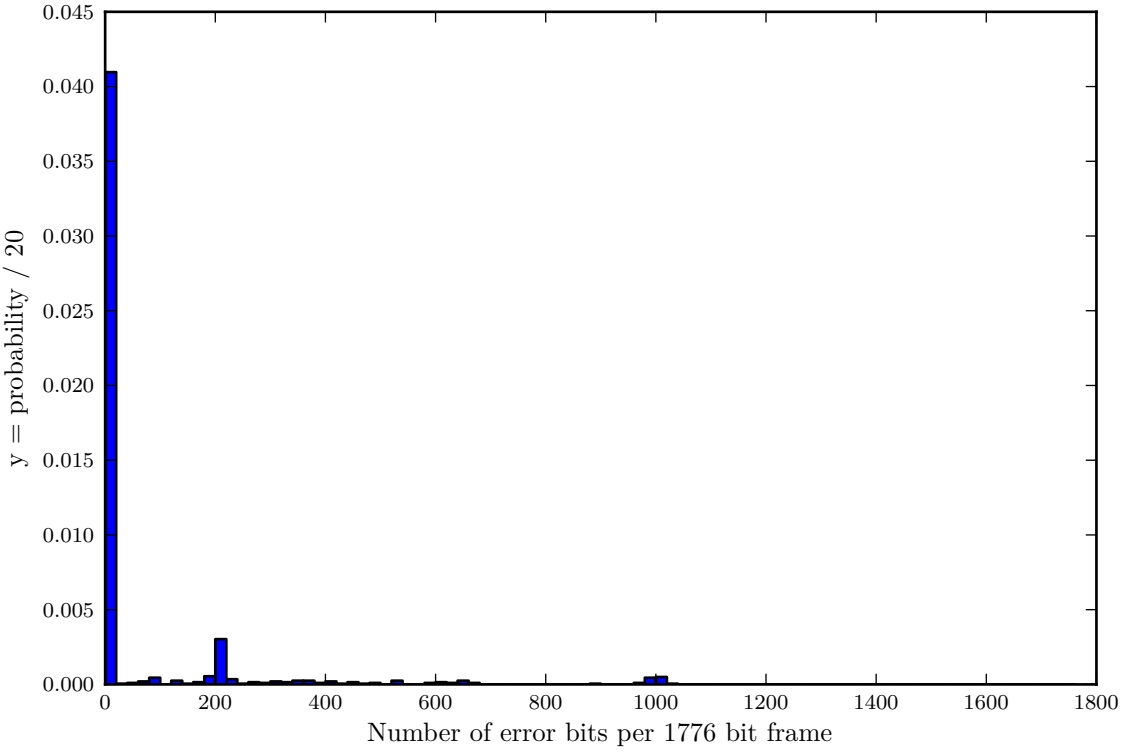
The histogram in graph 8.2 indicates the number of error bits per frame in frames with more than 0 bit errors. Because a CRC check is transmitted with the frame, it is quite possible to fix 1 bit, but even fixing up to 3 consecutive bit errors is common practice [40, Chapter 10.5 - One

Frame type	Number	Percentage of total frames
All frames	39658	100%
No errors	38651	97.46%
One bit error	6	0.015%
Two bit errors	495	1.248%
Three bit errors	1	0.0025%
More than three errors	505	1.273%

Table 8.1: Summary of bit error results between two embedded transceivers.

Bad Apple Don't Spoil the Whole Bunch], especially if the expected length and format of the frame is known.

Graph 8.2: Histogram of error bit count for frames with more than zero error bits between two embedded transceivers.



From this test it can be seen that communication between two embedded transceivers perform as expected. Because of the limitations of an embedded microcontroller the performance of specifically demodulation of received signals is less than ideal. This is a known shortfall and can be improved by using dedicated modem IC's, like the CML MX614, rather than trying to demodulate the signal in software. Offloading the computational overhead to dedicated hardware can also have a positive effect on power usage.

8.1.3 Embedded transmitter, Linux computer receiver

The alternative transceiver is a Linux computer running soundcard modem software. In this case a Raspberry Pi was used, with a USB soundcard and running DireWolf. A layout of the test setup can be seen in figure 8.2.

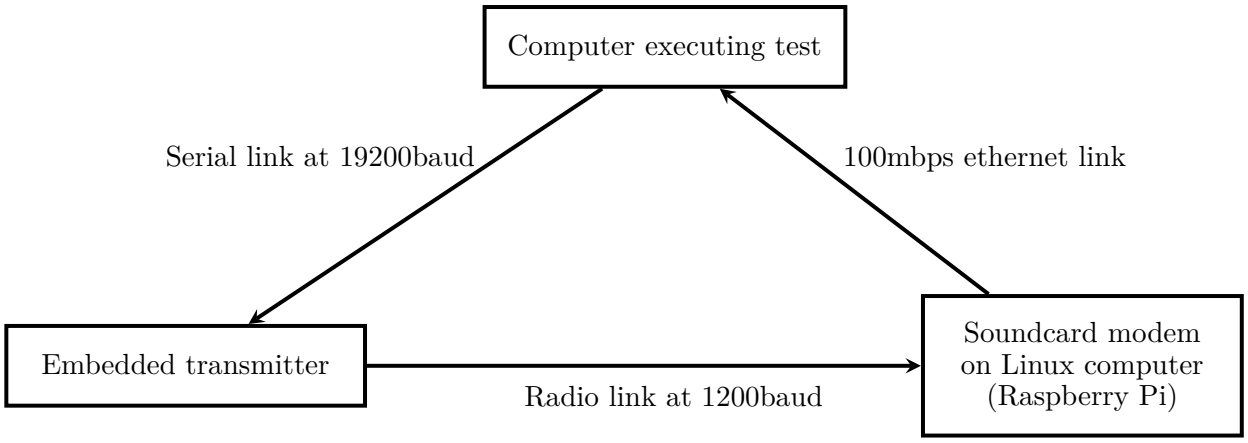


Figure 8.2: Layout of test between an embedded transmitter and Linux computer receiver.

In this test the theoretical latency will include the transmission delay, one serial link delay, and the delay to transfer the data across 100mbps ethernet. Because the ethernet link uses TCP, our data is wrapped in a couple of lower level protocol headers, increasing the amount of data transferred via ethernet. TCP is also not guaranteed to deliver the data within a specific time, so this delay can not be theoretically determined. The closest we can come is to specify a minimum time it can take to transfer our data via a 100mbps link.

Minimum ethernet transfer delay

$225\text{bytes} \times 8\text{bits} \times \frac{1\text{second}}{100 \times 10^6 \text{bits per second}} = 18\mu s$

Total theoretical delay

Transmission delay + Serial transfer delay + Ethernet transfer delay

$1.80s + 117.19ms + 18\mu s = 1.9172$ seconds

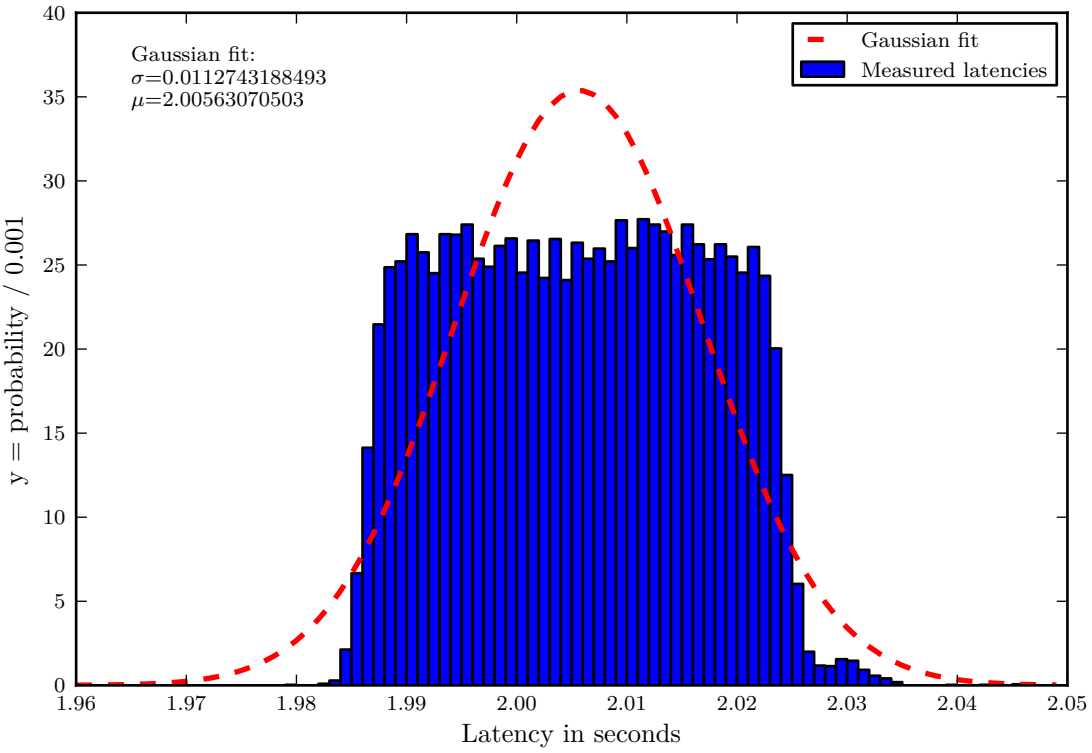
As can be seen in the histogram in graph 8.3, the expected latency measured was $\mu = 2.0056s$, with a jitter of $\sigma = 0.0113s$. The total latency is thus shorter than in the test with two embedded transceivers, but this is mostly because of one serial link that is replaced by a much faster ethernet link. The measured expected latency is about 7.8σ higher than the theoretical minimum, which is higher than in the two embedded transceivers' case. This increase in delays that are not theoretically included is most likely due to overhead and task switching in the Linux operating system, as well as extra TCP overhead on the ethernet link.

During the $45\frac{1}{2}$ hours this test ran, a bit error rate (BER) of 129×10^{-6} was measured, which is 77 times better than in the test with two embedded transceivers.

The histogram in graph 8.4 indicates the number of error bits per frame in frames with more than 0 bit errors. A summary of the number of errors can also be seen in table 8.2.

From these test results we can see that the soundcard software receiver on a Linux computer functions quite well and the number of errors are very low compared to the embedded receiver. The little extra jitter on latency is completely overshadowed by the improved error rate.

Graph 8.3: Histogram of measured latencies between an embedded transmitter and Linux computer receiver.



Frame type	Number	Percentage of total frames
All frames	31596	100%
No errors	31497	99.68%
One bit error	0	0%
Two bit errors	45	0.142%
Three bit errors	0	0%
More than three errors	54	0.171%

Table 8.2: Summary of bit error results between an embedded transmitter and Linux computer receiver.

8.1.4 Linux computer transmitter, Embedded receiver

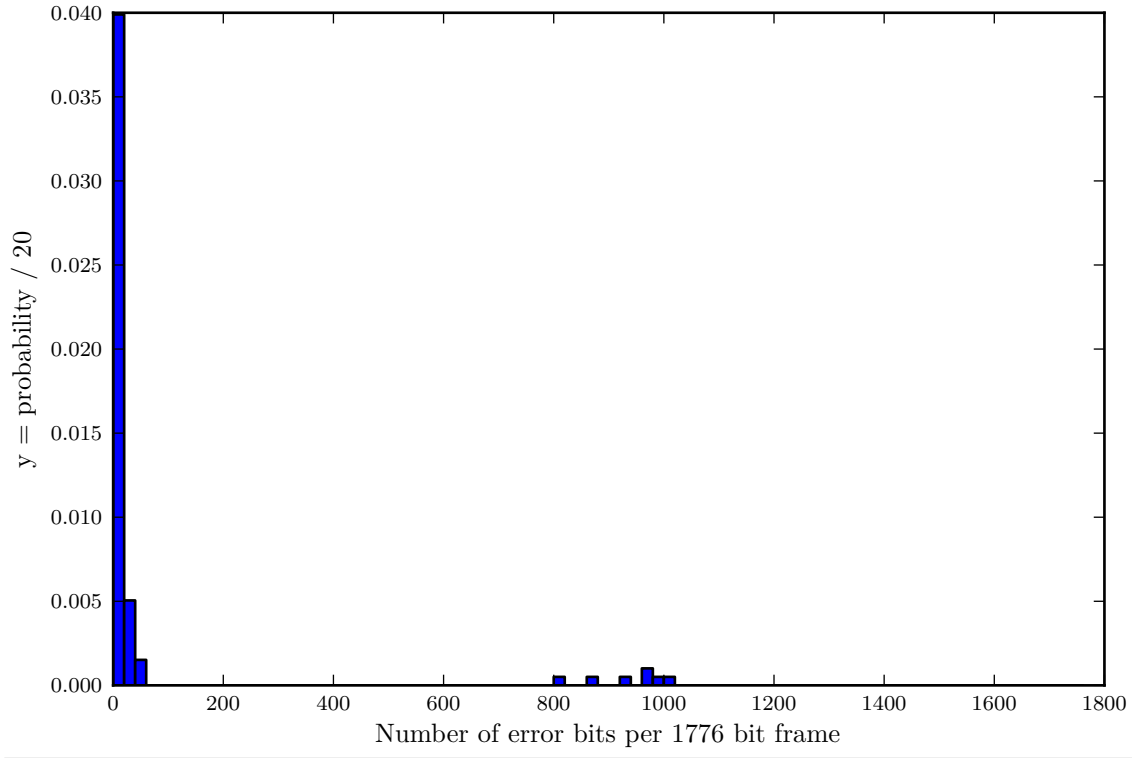
This test swapped the transmitter and receiver from the previous test. The goal is to see what we should expect for acknowledgement messages if the base station is a Linux computer and the remote nodes in the bush are embedded transceivers. The layout of this test can be seen in figure 8.3.

Latency	Value
Minimum	2.0524s
Maximum	5.0059s
Mean	2.6595s

Table 8.3: Summary of latency measurements between a Linux computer transmitter and an embedded receiver.

From the histogram in graph 8.5 we can see that the result is not a normal distribution as

Graph 8.4: Histogram of error bit count for frames with more than zero error bits between an embedded transmitter and Linux computer receiver.



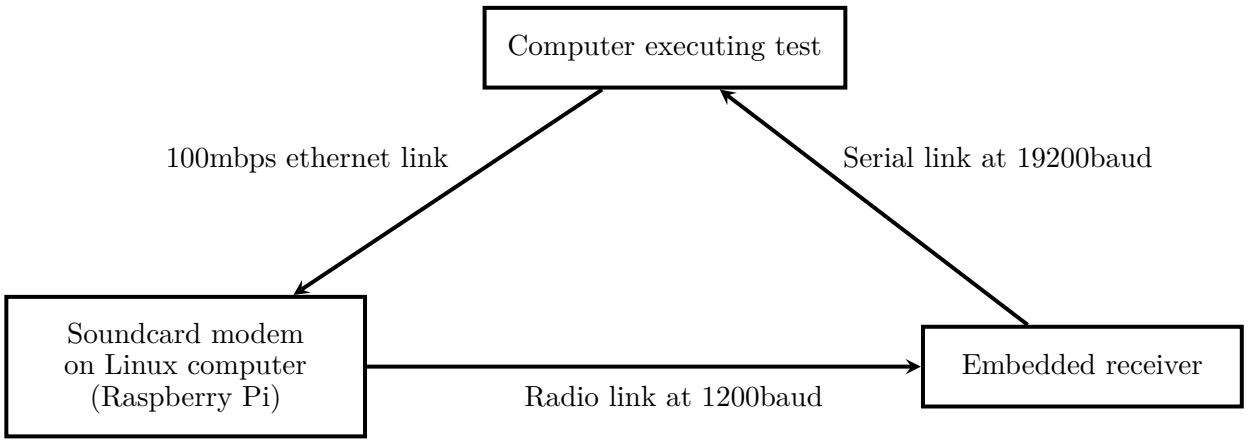


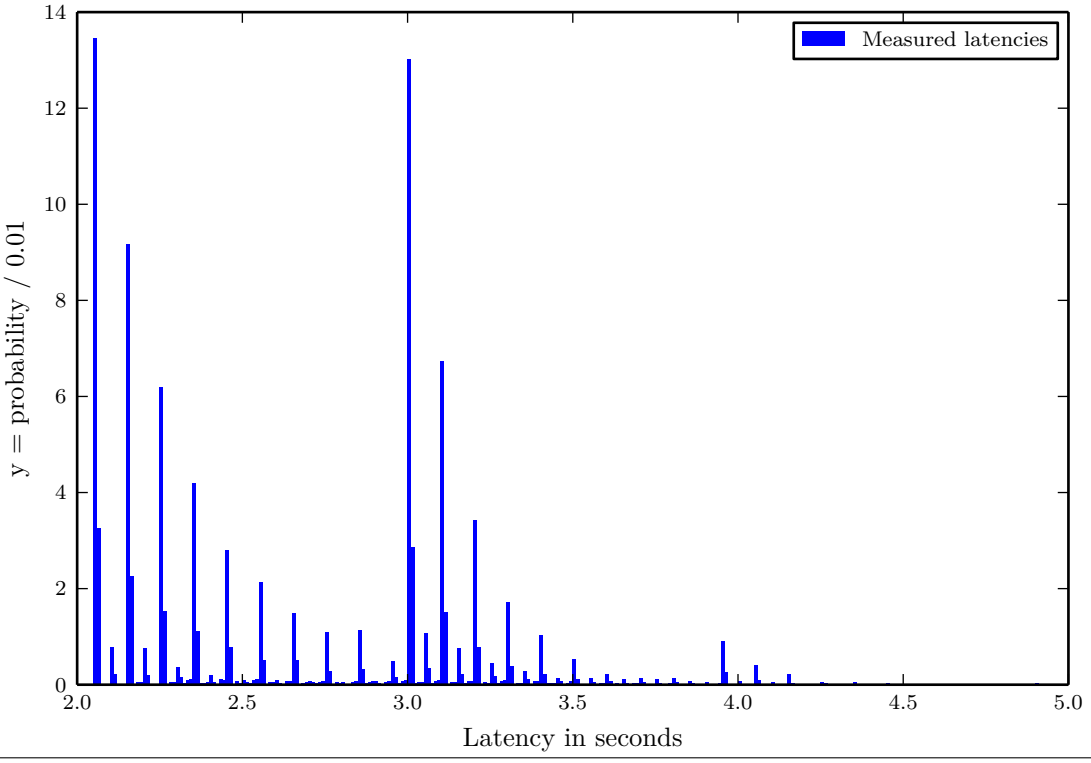
Figure 8.3: Layout of test with a Linux computer transmitter and an embedded receiver.

we would expect, but rather an exponential-like distribution. There are three possible reasons for this.

1. Initial suspicions were that the cause for the exponential-like distribution was the process scheduler in the Linux operating system. Changes to the scheduler did have an effect on the distribution of latencies, but the effect was minor.

2. The testing computer to the transmitter used is a 100mbps ethernet link. This link uses TCP [41]. TCP has a couple of mechanisms to prevent network flooding that causes extra and unpredictable delays[42], [43]. The worst one is Nagle's algorithm[44] that is meant to prevent small packets to flood the network, by delaying the transmission until enough

Graph 8.5: Histogram of measured latencies between a Linux computer transmitter and an embedded receiver.



data is buffered. In our case our frames can be considered small compared to the normal size of TCP packets on 100mbps ethernet. It is possible to switch off this functionality by using the TCP_NODELAY socket option[45], but either this setting did not work, or it did not have any effect on this test.

3. The soundcard modem software used on the Linux transmitter, called DireWolf, has the functionality to check the radio channel to see if it is clean, delay a random amount of time, and only then transmit the frame. It can be seen as a primitive implementation of CSMA/CA. From the DireWolf user guide[40, 8.2.4 Radio Channel – Transmit timing] the following quote:

   > SLOTTIME and PERSIST are used to generate a random time between the time when the channel is clear and when we start transmitting.
   > They have the same traditional meanings as in nearly every TNC going back 30 years. You probably want to keep the defaults. This delay is not used when transmitting digipeated frames.
   > SLOTTIME 10
   > PERSIST 63

   This is the most probable reason for the latency distribution we see in graph 8.5.

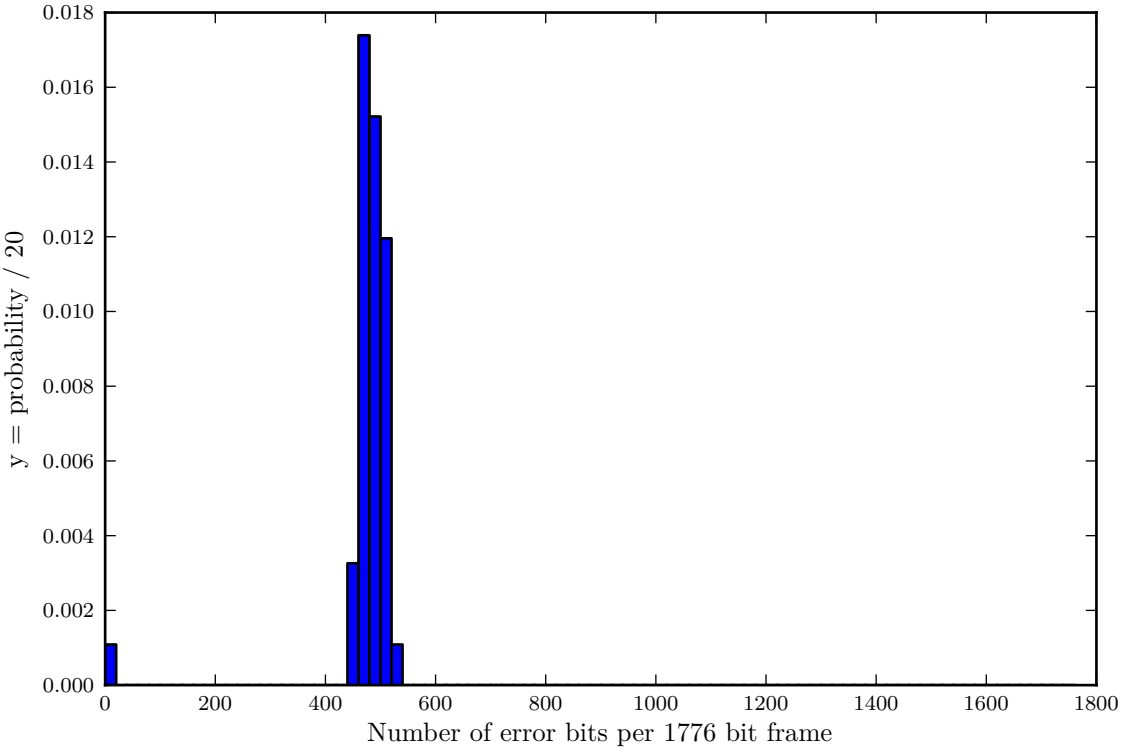
The same theoretical latency as in the previous test is applicable: $1.9172s$. This compares well to the minimum measured latency of $2.0524s$ as in table 8.3.

During the $43\frac{1}{2}$ hours this test ran, a BER of 393×10^{-6} was measured, which is 3 times worse than with the Linux computer receiver, but 25 times better than using both an embedded transmitter and embedded receiver.

Frame type	Number	Percentage of total frames
All frames	31277	100%
No errors	31231	99.85%
One bit error	0	0%
Two bit errors	1	0.003%
Three bit errors	0	0%
More than three errors	45	0.14%

Table 8.4: Summary of bit error results between a Linux computer transmitter and an embedded receiver.

Graph 8.6: Histogram of error bit count for frames with more than zero error bits.



Graph 8.6 shows the histogram of number of errors for frames with more than zero error bits. A clear and interesting result is the cluster of frames with error bit counts of around and just under 500. This is about 30% of the frame. Looking at the composition of the frame we can see 150 of the 222 bytes are data, leaving 72 for headers. The headers make up 32% of the frame. A logical conclusion to make is that the embedded receiver sometimes gets out of sync at the transition from the header part of the frame to the data part. The number of times this happened is not significant enough to increase the BER too much. It can however be improved with a better embedded receiver with external modem IC, as stated in 8.1.2.

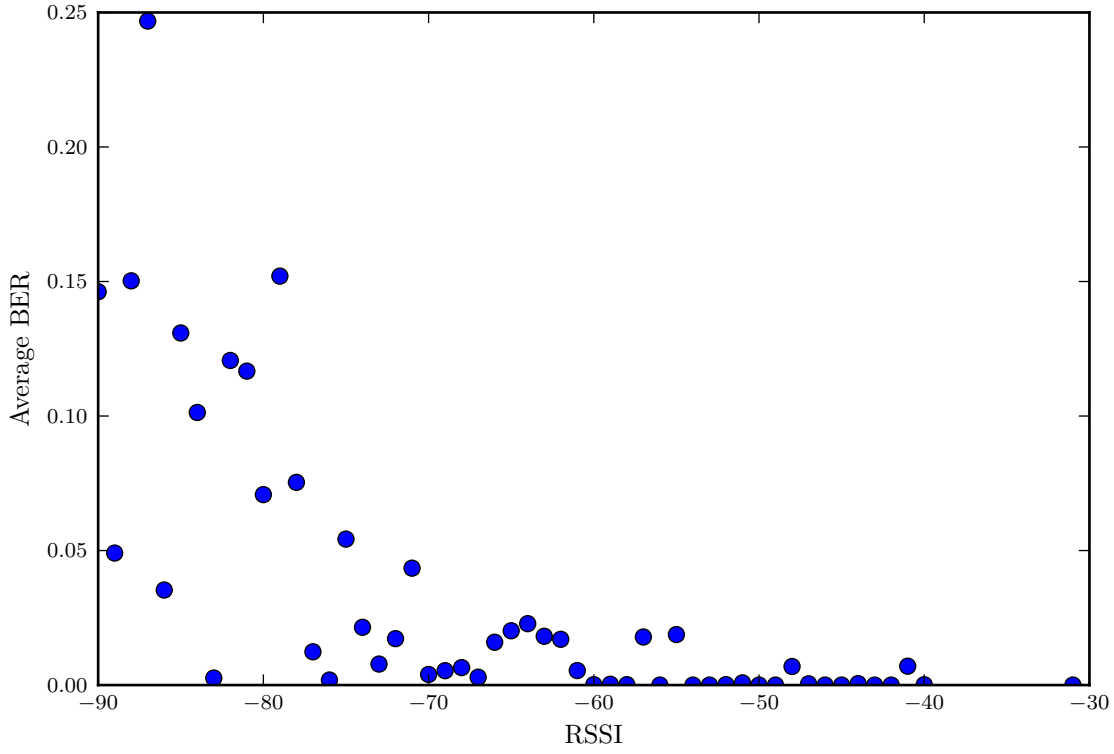
As a random sequence of data was used during this test, it is quite possible that certain random bytes are more susceptible to causing bit errors when received by the embedded receiver.

8.2 Field tests and comparison with simulations

These tests were all done using both an embedded transmitter and an embedded receiver, except where otherwise stated.

8.2.1 BER versus RSSI

Graph 8.7: Bit error rate versus received signal strength indication.



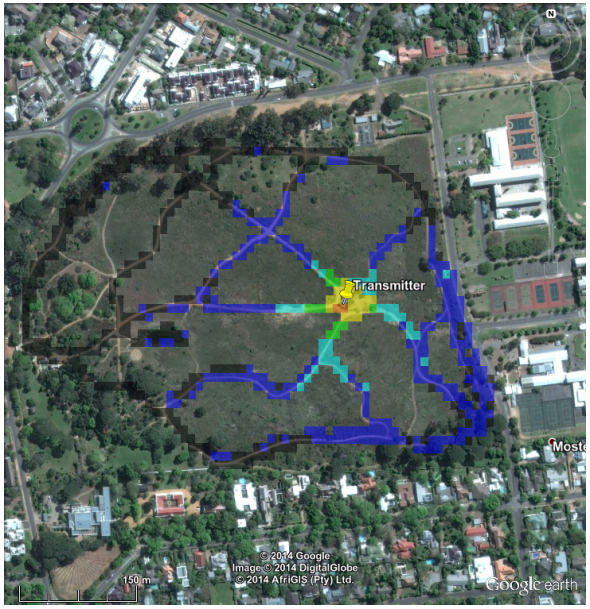
Graph 8.7 indicates the performance of the embedded receiver's radio and demodulator. No frames were received at signal strengths of below $-90dBm$. The general form of the graph is what one would expect, with more errors at lower signal strengths. It also tends towards an exponential distribution, but can not be assumed.

Although the result of this test looks like expected, especially with a less than ideal receiver, it should be done differently. Using the RSSI on the x-axis is practical as it is easily measured. A better approach will however be to plot BER versus signal-to-noise-ratio (SNR) or versus E_b/N_0 (energy per bit). This is however a difficult value to obtain.

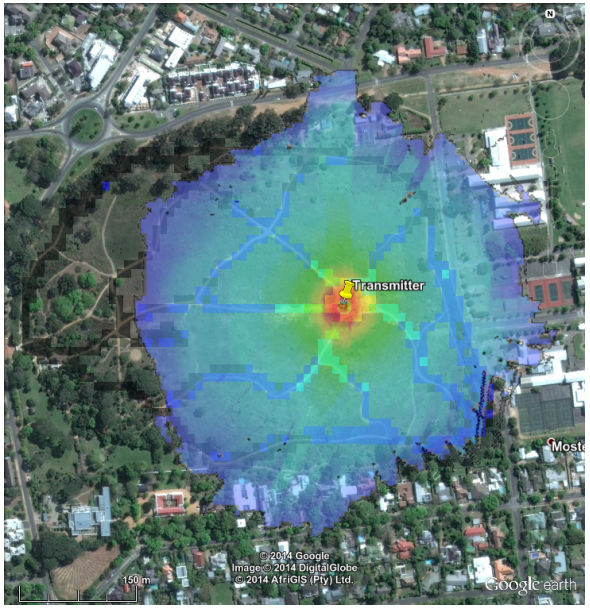
The likely reason for the spread-out distribution of samples in graph 8.7 is that the RSSI was falsely higher due to external interference. A SNR measurement should improve this. All results below $-87dBm$ should also be ignored as the noise floor averages around this value and causes invalid results in the RSSI as it was measured in this test.

8.2.2 Flat area: Jan Marais Park

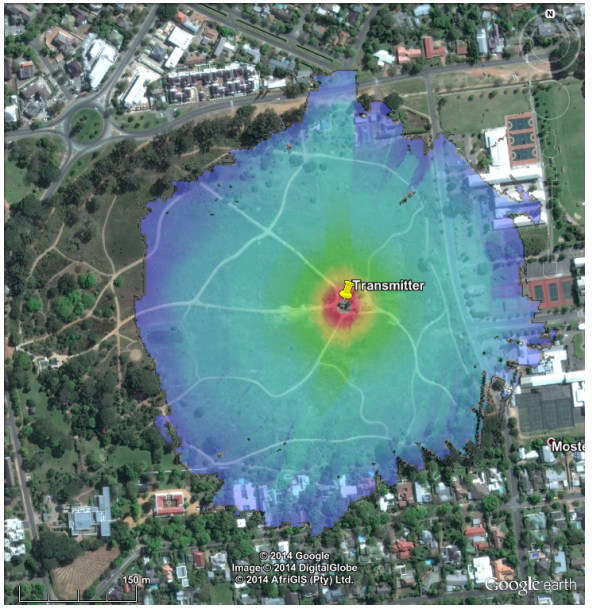
This test was done in the Jan Marais Park in Stellenbosch. The transmitter was at a location of 33°55'57.40"S, 18°52'39.15"E. This park is quite flat with vegetation similar to what is to be



(a) Measured coverage



(b) Simulation and measurement overlayed



(c) Simulated coverage

Colour	RSSI (dBm)
	< -87
	$-87 \leftrightarrow -80$
	$-80 \leftrightarrow -70$
	$-70 \leftrightarrow -60$
	$-60 \leftrightarrow -50$
	$-50 \leftrightarrow -40$
	$-40 \leftrightarrow -30$

(d) Legend: measurement

Colour	RSSI (dBm)
	-87
	-83
	-79
	-75
	-67
	-63
	-59
	-55
	-51
	-47

(e) Legend: simulation

Figure 8.4: Comparison of measured and simulated radio coverage in the Jan Marais Park in Stellenbosch.

expected in unspoilt bushveld areas. In figure 8.5(a) the transmitter can be seen standing on the ground between the indigenous vegetation and in figure 8.5(b) the receiver, which is a portable spectrum analyser (Signal Hound) and a GPS receiver, can be seen carried in a backpack by a person riding a bicycle.

(a) The embedded VHF transmitter (b) The mobile receiving station used for the test.

Figure 8.5: Test setup used for the test in the Jan Marais Park.

In figure 8.4 the results of the measurement and a simulation of the coverage of the same area can be seen side by side. Using a GPS and a portable spectrum analyser, the RSSI was measured and mapped. Results are clustered in 0.0001° by 0.0001° rectangles and the average RSSI per rectangle is translated to a colour to plot on the map. The legend in 8.4(d) shows the colour to RSSI value mapping.

A simulation was done afterwards with Radio Mobile, firstly with the actual parameters the test was done with. The simulation results were too optimistic. The radios were very close to the ground and strongly affected by the vegetation in the park. To compensate for the extra loss due to the clutter, extra attenuation was built into the simulation. An extra 20dB loss seems to give a good match between the simulation and the real-world measurements. The simulation results seen in figure 8.4(c) is with this extra 20dB attenuation. More details about this simulation and how the 20dB attenuation figure was obtained can be seen in appendix E.

Because the simulation does terrain analysis, it only depends on topographic data. Any other man-made or natural obstacles are completely ignored. The simulation will therefore never perfectly match the real-world situation. The naive approach of building extra losses into the simulation seems to work quite well. Even though the simulation results are not perfect, it can at least give a rough idea of what is to be expected in an area with clutter.

Parameters used for the test in the Jan Marais Park

Frequency 145MHz

Actual output power 27dBm

Simulation output power 7dBm

Antenna Vertical, Omnidirectional (2.15dBi)

8.2.3 Mountainous area: Jonkershoek

The following few tests were done in Jonkershoek, Stellenbosch. It is an area with a mixture of pine tree plantations and indigenous vegetation. On both the northern and southern side of the Jonkershoek valley are mountains with steep inclines. Because of the changes in elevation, there are areas with good line of sight, but also areas with no visibility due to thick vegetation.

8.2.3.1 Partial vegetation with line of sight

In this test an area was chosen that has thick vegetation a short distance from the transmitter, but line of sight is restored quickly thereafter as the slope increases. It can be clearly seen in the measured results that radio coverage close to the transmitter is much worse than expected from the simulation, but as soon as the receiver is elevated above the vegetation and line of sight is restored, it performed well again.

For this test a transmitter location of 33°59'16.51"S, 18°57'19.68"E was used.

Legend 8.6(a) and legend 8.6(b) are respectively applicable to the measurement in 8.7(a) and the simulation in 8.7(b). The markers in 8.7(b) are for locations where frames were received and decoded. As can be seen in 8.6(c), the colour of the marker indicates how many bit errors occurred per frame.

Colour	RSSI (dBm)
	< -87
	$-87 \leftrightarrow -80$
	$-80 \leftrightarrow -70$
	$-70 \leftrightarrow -60$
	$-60 \leftrightarrow -50$
	$-50 \leftrightarrow -40$
	$-40 \leftrightarrow -30$

(a) Legend: measurement

Colour	RSSI (dBm)
	-87
	-83
	-79
	-75
	-67
	-63
	-59
	-55
	-51
	-47

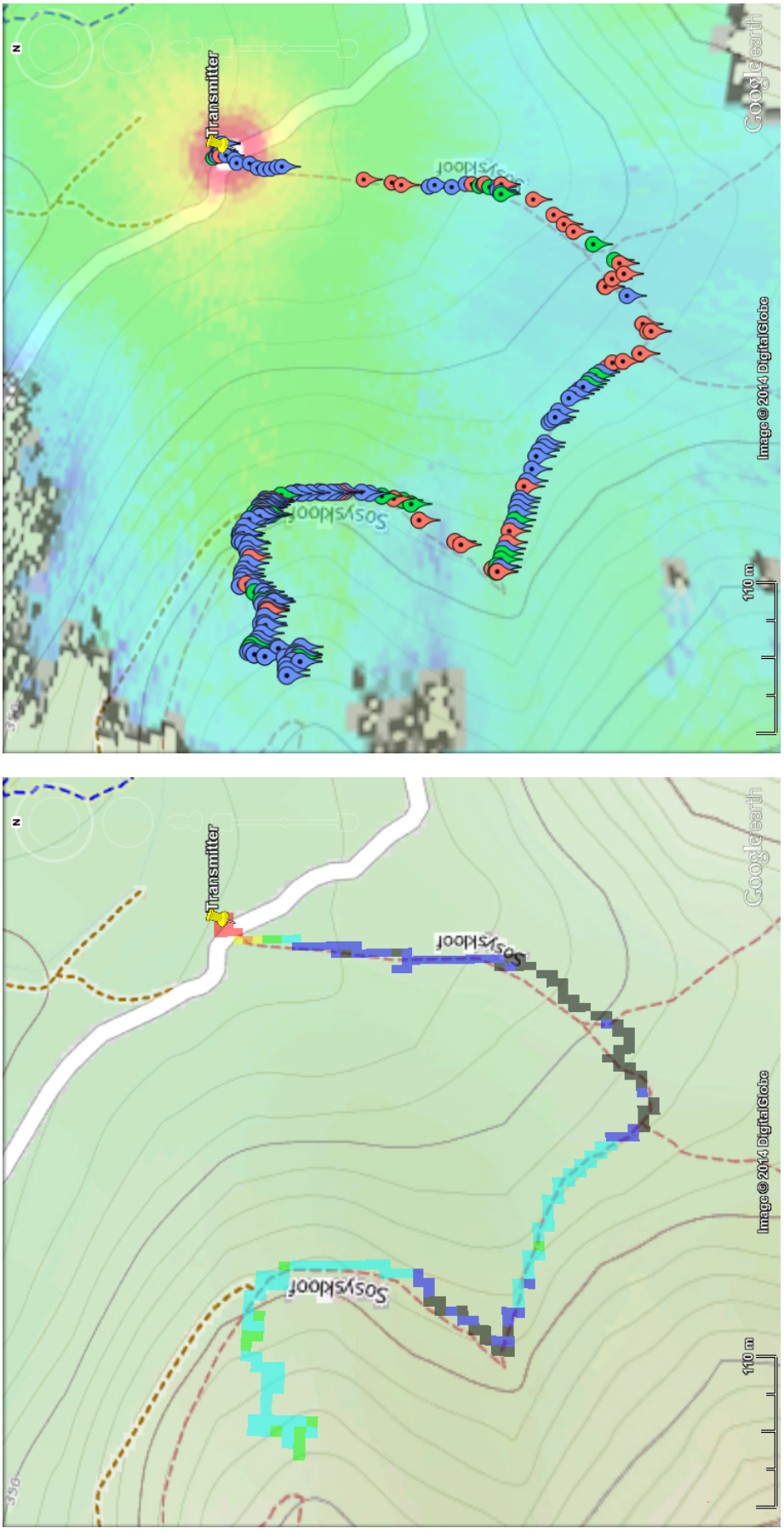
(b) Legend: simulation

Mark	Frame error count
	0 error bits
	1, 2 or 3 error bits
	>3 error bits

(c) Legend: markers

Figure 8.6: Legend for coverage plots in figure 8.7.

The simulation is done at $7dBm$, while the real power output was $27dBm$. Even with the $20dB$ lower power, the simulation still seems a little too optimistic. It can however be seen that lower signal strength locations correspond between the measurement and simulation.



(a) Measured RSSI

(b) Simulated coverage with received frame locations

Figure 8.7: Comparison of measured and simulated radio coverage in Jonkershoek, take 1.

In a radius of about 50m from the transmitter frames were successfully received, while the area between 50m and 250m from the transmitter was prone to errors and frame losses. Because the first area closest to the transmitter is quite flat, the vegetation has a big influence on the signal strength. The measurement was done while walking and the time it takes to transmit a frame is about 2 seconds. Therefore even though the average signal strength might be good enough for a transmission, the variability in signal strength caused bits to get corrupted or frames to be completely indistinguishable from noise.

Where the path follows the contour, partial line of sight is re-established as vegetation on the transmitter's side of the path is lower than the receiving antenna. Received signal strength went up and more frames were successfully received. Getting closer to a small valley, the vegetation becomes thicker and line of sight disappears again. Signal strength went down and many frames got lost. As soon as line of sight is restored after leaving the valley the most frames are transferred correctly.

8.2.3.2 Thick vegetation with line of sight later

An area with thick fynbos was chosen for the next test. The path takes a constant gradual climb in quite a straight line. The fynbos has a height of 2m, obscuring the line of sight path completely. Only after about 450m the path takes a slow turn left and follows a contour line for a short while. At this point line of sight is partially restored.

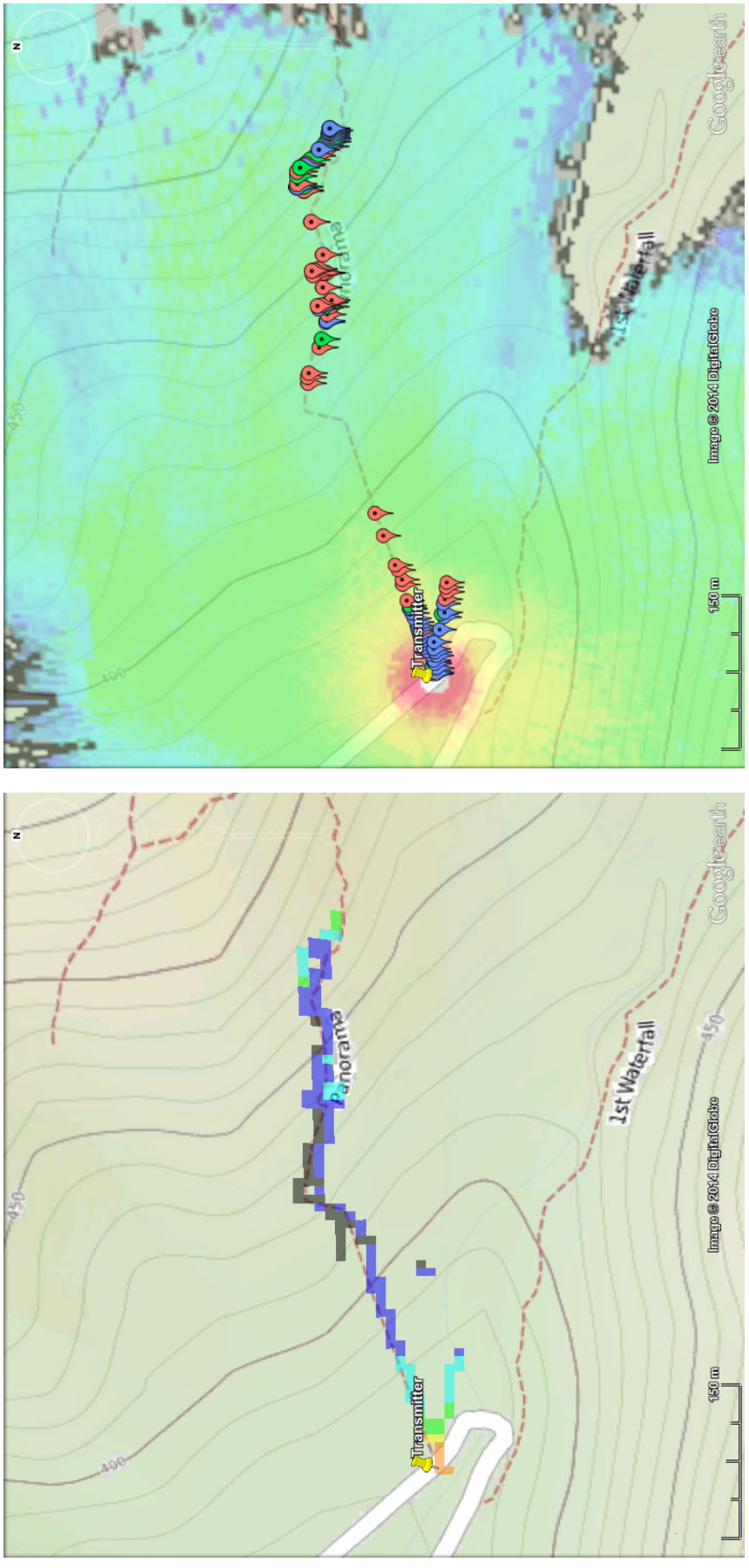
For this test a transmitter location of 33°59'36.40"S, 18°58'28.09"E was used. The same legend as in section 8.2.3 applies. The simulation was again done at $7dBm$, while the test was performed at $27dBm$.

The result of the test and the simulation can be seen in figure 8.8. It is clear from the contour lines and simulation that the expected coverage should be good as there should be good line of sight. Due to the vegetation this is not the case and in practice the radio link can't make it further than 100m. The measured signal strength also only corresponds in areas where partial line of sight is available. Again the conclusion is that vegetation has a huge influence on the coverage and that even a simulation at 20dB less power is still too optimistic in cases with thick vegetation.

8.2.3.3 Wide area

In this test a wider area in Jonkershoek was used to measure signal strength. Signal strength was measured with a portable spectrum analyser. Data frames were received with a commercial two-way radio set of the brand Icom, which is commonly used by amateur radio operators. The performance of this radio is better than the Radiometrix radio modules used in the embedded transceiver. Decoding of the frames was done using the DireWolf soundcard modem software, with error recovery enabled for up to one error bit. The received frames in this test are therefore only frames with at most one error bit.

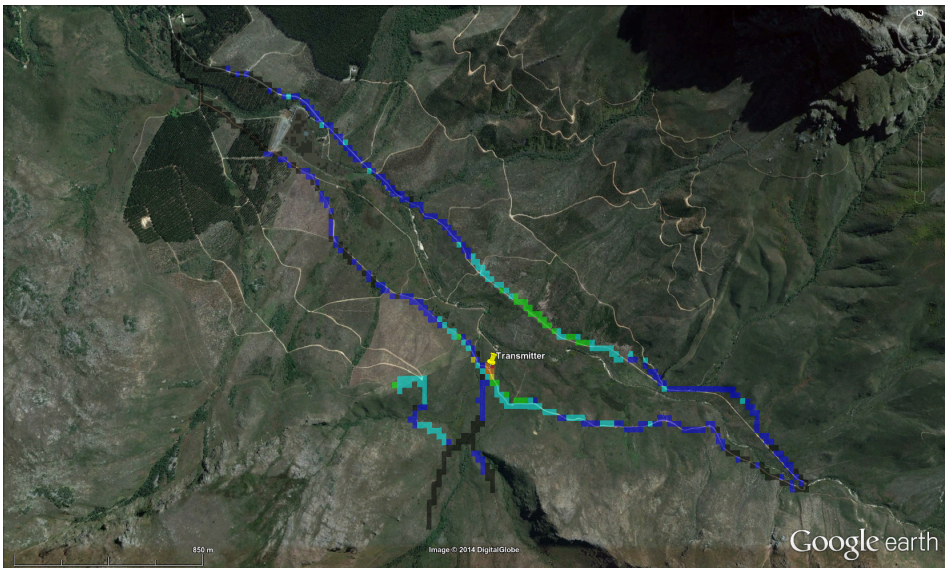
The wider Jonkershoek area forms a clear open valley. Line of sight is thus possible in quite a large area. This can be seen in the simulation and by the number of successfully received frames. In contrast to north of the transmitter where line of sight was good, coverage to the south was worse due to line of sight being obscured by vegetation.



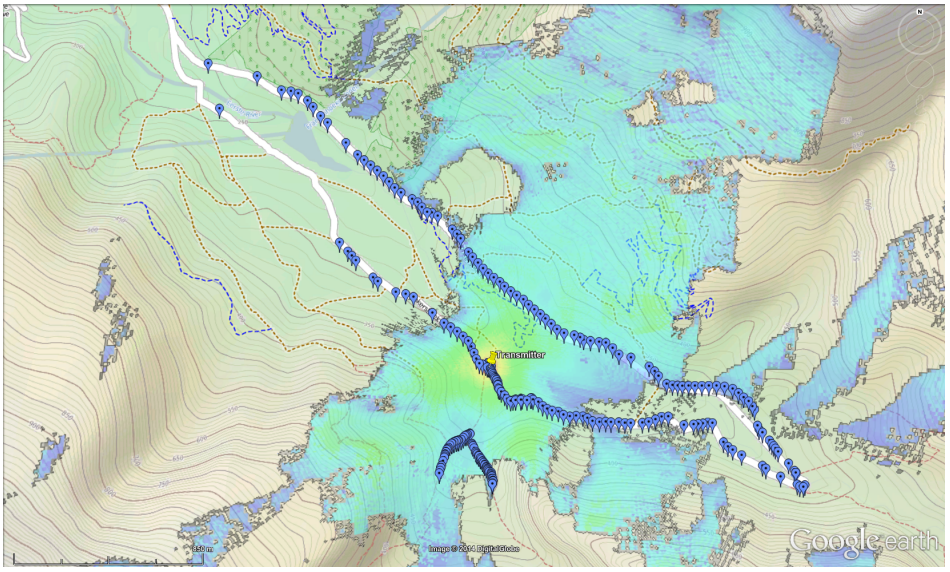
(a) Measured RSSI

(b) Simulated coverage with received frame locations

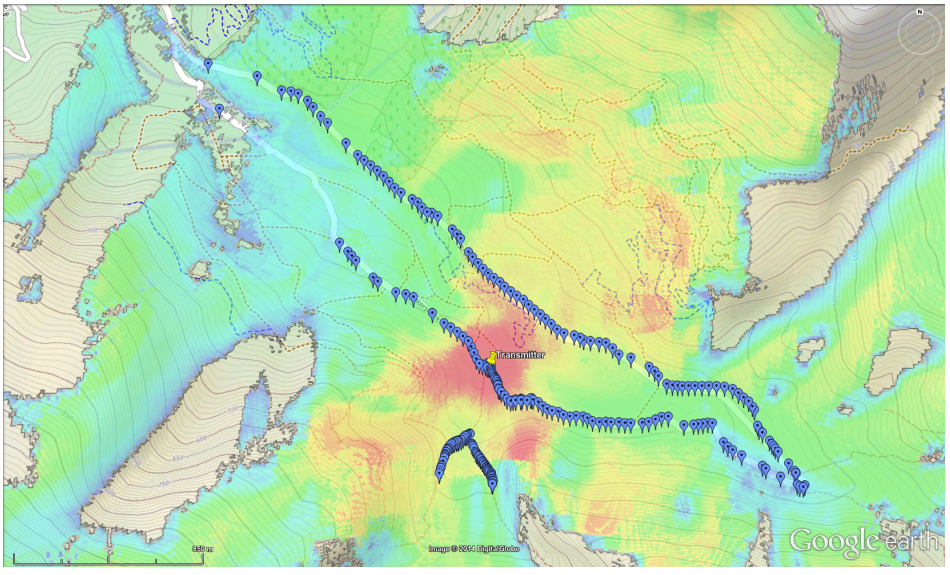
Figure 8.8: Comparison of measured and simulated radio coverage in Jonkershoek, take 2.



(a) Measured RSSI, 0.0003° by 0.0003° rectangles



(b) Simulated coverage at 7dBm with received frame locations



(c) Simulated coverage at 27dBm with received frame locations

Figure 8.9: Comparison of measured and simulated radio coverage in Jonkershoek, take 3.

Looking at the overall coverage simulation and received frames, it seems like the model with extra 20dB attenuation (output power of 7dBm) is too conservative in certain parts. A simulation at the actual 27dBm matches the real-world measurements in some places better than the one with extra attenuation. The most likely reason for this is that reflections from the cliffs and ridges in the valley help with propagation of the signal. The simulation does not take reflections into account.

Performance of a radio system in a real-world scenario can therefore be either worse than expected due to attenuation from vegetation, but it can also be better than anticipated as reflections can have a positive effect.

8.3 Power usage of the system

Power consumption of electronics is a very important topic, especially if it needs to be powered by battery and lasts as long as possible. This section gives an overview of the power consumption for the hardware that was developed and used during testing.

This hardware does not include all the necessary components needed for a working system, like a GPS receiver. The consumption values must be seen as an indication of what will be the minimum requirements for a communication system. All extra added components and sensors will increase these power consumption values.

8.3.1 Sensor module

The sensor module has a number of sensors and components which can consume energy. The most important ones are:

- UHF Radio
- Accelerometer
- Temperature sensor
- Watchdog timer

Depending on which of these sensors are needed, the power usage can be optimized by switching the unnecessary ones off.

All measurements for the sensor module is done at a supply voltage of $3V$, and the results can be seen in table 8.5. When all the sensors are switched off, the microcontroller can be put into sleep mode. A summary of power usage in the sleep modes can be seen in table 8.6.

8.3.2 VHF radio module

The VHF radio module consists of an ATmega 328P microcontroller, a Radiometrix SHX1 radio and some other components to connect the two together, including an opamp for amplifying the received audio level.

In the receive state the radio consumes power for demodulation of the FM signal. The opamp shifts the received audio to the correct levels for maximum voltage swing to input to the ADC. If valid data is received it is processed and transferred on the UART.

ATmega 328P	RFM12b radio	LM75A temperature sensor	MMA7455 accelerometer	Power usage
On	TX	off	off	$24.225mA$
On	RX	off	off	$13.665mA$
On	off	On	off	$3.05mA$
On	off	off	On	$3.44mA$
On	off	off	off	$2.528mA$

Table 8.5: Measured power consumption of the sensor module and its components. For all these tests the watchdog timer was switched on. During transmit mode the RF power output of the radio is set to maximum (7dBm).

ATmega Sleep mode	Watchdog Timer	Power usage
ON	On	$2.528mA$
IDLE	On	$932\mu A$
IDLE	off	$931\mu A$
POWER DOWN	On	$14.180\mu A$
POWER DOWN	off	$9.948\mu A$

Table 8.6: Measured power consumption of the ATmega 328p microcontroller in all its power saving states, with all sensors and the radio switched off.

During radio transmission the microcontroller generates audio by use of a R2R circuit (resistor ladder). The SHX1 transmits at 500mW RF output power.

All measurements for the VHF module is done at a supply voltage of $5V$.

State	Components used	Power usage
RX (receiving valid data)	ATmega 328P, Radiometrix SHX1 RX, ADC, UART	$38.63mA$
RX (no valid data)	ATmega 328P, Radiometrix SHX1 RX, ADC	$38.35mA$
TX	ATmega 328P, Radiometrix SHX1 TX, Resistor ladder	$269.4mA$

Table 8.7: Measured power consumption of the embedded VHF transceiver.

8.3.3 Power requirement calculations

To do an estimation of the power consumption of every part in the system, the scenario in which it will be configured and used need to be known. The following assumptions and decisions were made:

1. The sensor module transmits its status once in 24 hours, and for the rest of the time it is in sleep mode.

2. In case of an emergency the sensor module is woken up by an external trigger and transmits an alarm message. An alarm message will be sent maximum once per day.

3. After every transmission the sensor module needs to listen for an acknowledgement before it can go back to sleep. If no acknowledgement is received the message is sent again up to ten times. Let's say this happens within a minute, or else we will give up.

4. The bridge node needs a UHF radio that listens permanently and will acknowledge any received message.

5. We can safely say all UHF transmissions are done within a second, although it will be much faster.

6. The bridge node forwards the message on VHF, which takes 2 seconds. It then needs to listen for an acknowledgement, which can take up to a minute.

7. A repeater node only has a VHF radio that listens constantly and forwards all messages it receives. The number of messages it needs to forward per day is proportional to the number of sensor modules in the system.

8.3.3.1 Sensor module

To have enough reserve power to send an alarm message at any time, we should have enough energy to send 2 messages at the moment when the 24 hour status is sent. Thus 2 messages every 24 hours.

Transmission of one message:
$1s \times 24.225mA \times 3V \ = \ 72.675mWs \ = \ 20.1875 \times 10^{-6}Wh$

The message needs to be repeated up to 10 times:
$20.1875 \times 10^{-6}Wh \ \times \ 10 \ = \ 201.875 \times 10^{-6}Wh$ for transmission per day.

Then the radio needs to listen for up to 1 minute, for at least two transmissions:
$13.665mA \times 60s \times 3V \ = \ 2.4597 \times 10^{3}mWs \ = \ 683.25 \times 10^{-6}Wh$
$683.25 \times 10^{-6}Wh \ \times \ 2 = 1.3665 \times 10^{-3}Wh$ for reception per day.

For the rest of the day the sensor module is in sleep with the watchdog timer enabled to wake it up once a day:
24 hours $\times$ 60 minutes $\times$ 60 seconds $\times$ $14.18\mu A \times 3V \ = \ 3.6...\mu Ws \ = \ 1.02096 \times 10^{-3}Wh$ for sleeping the rest of the day.

Total theoretical consumption for sensor module: $2.589335 \times 10^{-3}Wh$ per day.

8.3.3.2 Bridge node

The bridge node listens on UHF permanently. The UHF radio needs to be on in receive mode and the microcontroller can not sleep, or need to wake up very often to check the radio's receive buffer.

UHF radio reception, with microcontroller on. UHF radio functions at 3 volt:
$24 \text{ hours} \times 60 \text{ minutes} \times 60 \text{ seconds} \times 13.665mA \times 3V \;=\; 3.541968 \times 10^6 mWs \;=\; 983 \times 10^{-3} Wh$

The VHF radio is switched on whenever needed. Maximum twice per day to transmit for 2 seconds each time. The VHF radio operates at 5 volt:
$2 \times 2 \text{ seconds} \times 269.4mA \times 5V \;=\; 5.388 \times 10^3 mWs \;=\; 1.4967 \times 10^{-3} Wh$

Afterwards the VHF radio needs to remain on in receive mode for a maximum of a minute to receive the acknowledgement. This also happens maximum twice per day:
$2 \times 60 \text{ seconds} \times 38.63mA \times 5V \;=\; 23.178 \times 10^3 mWs \;=\; 6.43 \times 10^{-3} Wh$

Total theoretical consumption of the bridge node: $990.9267 \times 10^{-3} Wh$

8.3.3.3 Repeater node

A repeater node always needs to listen on VHF and relay any received message. This will be up to two messages and their acknowledgements per sensor node per day.

VHF radio in receive mode:
$24 \text{ hours} \times 60 \text{ minutes} \times 60 \text{ seconds} \times 38.63mA \times 5V \;=\; 16.68816 \times 10^6 mWs \;=\; 4.6356Wh$ per day

Transmission of frames:
N sensors $\times$ 4 transmissions $\times$ 2 seconds $\times$ $269.4mA \times 5V \;=\; 10.776 \times 10^3 \; N \; mWs \;=\; 3N \; Wh$ per day

Total theoretical consumption of a repeater node in a network with only one sensor: $7.6356Wh$

8.3.3.4 Lifetime of sensor module on two AA batteries

A normal Duracell 1.5V Alkaline AA Battery (MN1500) available from RS components has a rated capacity of 2.850Ah [38, table 1], but from their test conditions[38, figure 3] a more conservative value of 2Ah (50 hours at 40mA discharge) seems to be the case.

Two 1.5V batteries in series provide us with the needed 3V, at a capacity of 2Ah. This is 6Wh. The sensor module consumes $2.589335 \times 10^{-3} Wh$ per day, thus it will last about 2317 days, or just more than 6 years on the two AA batteries.

These Duracell Alkaline-manganese AA batteries have a quite long shelf life, with 85% of the capacity still available after 4 years[38, section 5.7]. This translates to a loss of $75mAh$ per year. Adding this to our daily consumption:
$75mAh/\text{year} \;=\; 205\mu Ah/\text{day} \Rightarrow 616.44\mu Wh/\text{day}$ (at 3V)
$2.589335 \times 10^{-3} Wh \;+\; 616.44 \times 10^{-6} \;=\; 3.206 \times 10^{-3} Wh$
Taking this loss into account, the sensor module will theoretically last:
$6Wh \div (3.206 \times 10^{-3} Wh) \;\approx\; 1871 \text{ days} \;\approx\; 5 \text{ years}$

For a start 5 years seem like a good lifetime to expect from a sensor without any on-board power generation. Future development can include a small solar panel to increase the lifetime.

8.3.3.5 Power needed for the bridge node

Depending on the layout of the network, the bridge node can either be on the animal itself, or be integrated with the network of repeaters. In case it is on the animal, power consumption is an important consideration.

A solar panel can work on the animal, but direct sunlight is not guaranteed. Simple calculations can indicate a minimum size for such a solar panel, but tests need to be done to indicate exactly how much light is to be expected on such a solar panel. Using the calculations from 8.3.3.6, it will mean a solar panel rated at minimum 207mW.

Another solution is to use a kinetic charger.

8.3.3.6 Power needed for the repeater node

Repeater nodes can be distributed by hand across an area. It is therefore possible to power these nodes with solar panels (photovoltaic cells) as the panels can be directed for optimal sunlight during installation of the node.

Solar panels are rated according to how much power they can deliver at STC (standard testing conditions). This is when they are irradiated by 1000Watt/m^2 of sunlight, at a temperature of 25°C[46].

In Stellenbosch the real average irradiance a solar panel would have received (including nights) according to measurements for the past 4 years (`http://weather.sun.ac.za`) is 217.06W/m^2. We can therefore safely divide a solar panel's rating by 5 to compensate for cloudy days and nights. In other words Stellenbosch gets an average of $24 \div 5 = 4.8$ hours of STC-sunlight per day.

A repeater node needs $(4.6356 + 3N)Wh$ per day. Say we have a network of one sensor, then we need about 7.6356Wh per day. It will take a solar panel rated at $7.6356Wh \div 4.8h = 1.59W$ to generate this amount of power.

If we repeat this calculation for a network with 10 sensor nodes it becomes:
$(4.6356 + 3N)Wh = 34.6356Wh$ needed per day
$34.6356Wh \div 4.8h = 7.22W$ rated photovoltaic cell

All these calculations take the worst case of two transmissions per sensor node per day into account. A 10W solar panel will therefore be excellent for $14\frac{1}{2}$ sensor nodes, but will more than likely be sufficient for double that number, a network of around 30 sensor nodes.

8.4 Summary

This chapter tested the latency and bit error rates obtained by the developed hardware across a VHF radio link. Radio coverage was measured and compared to the simulation and thus providing an idea how simulations should be done to compensate for clutter. Power usage is

measured and proposals are made on how to provide enough energy for the devices that are used in the wildlife monitoring network.

§ 9. Conclusion

This chapter reviews what has been done in this project, how well it performed and what can be done to improve the system.

9.1 Discussion of Results and Summary

9.1.1 Sensor module

Power usage of the sensor module is very good and better than expected. Miniaturising the circuit took considerable effort, but the end result is a fully working and very versatile sensor module with UHF communication possibilities. The PCB design was made to fit on the back of a double AA battery holder, providing an easy but sturdy mechanical design. Augmenting the power supply with a small solar cell can allow this unit to function as an always-on UHF network node. The coverage range for the UHF radio is not as good as for VHF. This is to be expected, but the small size, low power usage and the fact that the sensor module can be made at a much lower cost than a VHF module, means a denser network can be deployed.

During testing a few mistakes were found in the PCB design. The ADC reference voltage pin on the microcontroller was connected to V_{cc}, which is the supply from the battery. In chapter 7.1 the battery voltage measurement is described. An ADC voltage reference of 1.1 volt, which is internally generated, must be used to accurately measure the battery voltage. The ADC reference voltage pin must therefore not be connected to V_{cc}, but rather be connected to a decoupling capacitor to ground.

A user button was added to the circuit in addition to the reset button. As the number of available pins on the microcontroller was limited, this button was added to an unused pin. Unfortunately the specific pin that was chosen can not be used as a GPIO, but only as a ADC input. This can be rectified by swapping the pins used by the battery voltage measurement circuit and the button.

9.1.2 Embedded VHF transceiver module

Tests indicated that the transmission part of the VHF module, using the resistor ladder to generate an audio sine wave, functions quite well. The bit error rate obtained when using the the VHF module as transmitter was acceptable.

The receiving and demodulating part of the VHF module worked well, but can be improved. The bit error rates were much higher (worse) when using the embedded VHF module than using a Linux computer to demodulate the data signal.

This circuit can be improved by using a dedicated modem IC, rather than demodulating the data signal in software. Adding a dedicated modem will decrease power usage as the microcontroller can remain in sleep mode until valid data is received. The modem IC is power efficient

as it is designed for a specific use. It uses less energy to demodulate the data signal than what the microcontroller would use to do the same in software.

Optimization of the PCB and the use of surface mount components can help in the miniaturisation of this circuit. The main part is the VHF radio which can however not be decreased in size. The radio can be replaced by another one if such a component becomes available on the market. Alternatively a handheld VHF radio (walkie-talkie style radio) can be used next to this circuit. Such a radio will provide better sensitivity and a high power output, increasing the coverage range. The price of such a radio is not much higher than the Radiometrix module, but there is a significant size increase. If the module is used as a static repeater station the size increase is not a problem.

9.1.3 Linux-based base station

The software that runs on the Linux computer performed very well. Tests indicated that the bit error rate obtained when using the software modem on the computer was much lower (better) than when using the embedded receiver. No improvement to the software modem on the computer is necessary.

The computer itself, in our case the Raspberry Pi, needs to be set up in a way that it performs more robustly. It needs a stable power supply providing enough current for the computer and all the accessories plugged into its USB ports. After a power failure the system should automatically start up and start the relevant software. A major problem with the Raspberry Pi is that its SD card, which is used for permanent storage, gets corrupted quite easily. To prevent this, writing to the SD card should be done as seldom as possible. Operating systems are not generally designed like this and constantly write log files to the system storage. The solution is in the form of an Industrial Perennial Environment from NutCom Services. It is a custom Linux operating system for the Raspberry Pi which is set up once, writing to the SD card, after which only reading is done from the SD card. All write operations that are necessary are done in RAM.

9.1.4 APRS as communication protocol

APRS and the AX.25 protocols proved to be a suitable, but not an ideal protocol stack to use. It is a very old, but proven system. Modulating and demodulating the signal is relatively easy. Off-the-shelf hardware can be used. The entire protocol stack is open and therefore implementation and modification of the protocols is quite possible. Adding minor modifications for wildlife tracking falls within the current protocol specification, allowing one to use existing hardware and software in the tracking network. The most of them are opensource allowing us to implement modifications as needed.

The biggest shortfall of the AX.25 protocol is the lack of proper error checking and the total lack of forward error correction. In 2006 an extension to AX.25 was published, called FX.25[47]. It implements forward error correction inside the AX.25 frame, allowing backwards compatibility to older hardware. The main use for this extension to the protocol is in satellite downlinks. If the AX.25 and APRS system is to be used for wildlife tracking, the implementation of FX.25 is definitely a recommendation.

Audio Frequency Shift Keying at 1200 baud, as used by Bell 202 (and therefore AX.25) does not make full use of the 25kHz bandwidth available on a normal VHF FM channel. This is mostly due to the limited audio bandwidth available on voice radios, as well as the pre-emphasis and de-emphasis implemented by voice FM radios. A 9600 baud alternative for AX.25 exists and is called G3RUH modulation[48]. Rather than injecting the baseband signal into the audio port of the voice radio, it is injected after the audio stages just before the IF stage or the FM modulator. From the original G3RUH paper[48] the following quote: *"The theoretical minimum audio bandwidth required to send 9600 baud binary data is 4800 Hz. Since a typical NBFM radio has an unfiltered response from zero to some 8 kHz, transmission of 9600 baud binary data is perfectly possible through it."*. Normal voice FM radios can still be used, but with small modifications. However, the amateur radio community still mostly uses 1200 baud Bell 202 modulation because of its simplicity, ease of implementation and a better bit error rate than G3RUH. Due to the higher baud rate of G3RUH a better signal to noise ratio is needed. In our case the speed of communication is not a problem, but rather the reliability and range of communication. Bell 202 modulation at 1200 baud is therefore for our use the better modulation scheme of the two.

Other alternatives that exist are mostly FSK or higher order IQ-modulation. To use these on low power embedded systems is less practical as more processing power is needed for the modulation and demodulation. Radios exist that have the necessary modems built in. They are normally only compatible with their own modulation scheme, forcing us to use the exact same radio at every point in the network. These modulation schemes may have better noise immunity, but being locked in to one radio or one vendor is not ideal. At this stage the recommendation is still to remain with Bell 202 AFSK modulation, because the wide array of hardware that is available makes implementation much easier and affordable.

On the network layer the flooded routing provided by APRS is sufficient. For better power efficiency a simple edge routing or geo-routing algorithm can be implemented on top of APRS. Route discovery can also assist in finding the shortest, or most effective route to a gateway. As the APRS protocol specifications are available in the public domain, any changes like these are quite possible. If a communication protocol like ZigBee is used, modifications like this will not be possible.

9.1.5 Radio frequency and coverage

From the radio coverage measurements it can be seen that communication between two animals that are more than a few tens of meters apart is highly unlikely. This is due to vegetation having a huge effect on the propagation of the radio signal. A system like Zebranet where communication between animals form the basis of the routing algorithm will not work in areas with relatively dense vegetation. Zebranet was tested on the grasslands of Kenia where the openness of the area allows for proper radio communication between the animals. Zebras are also a species that live in herds, always close together. This allows the use of low power radios for communication in the network.

Monitoring solitary animals needs a larger radio coverage. The VHF frequency band proved to be a good compromise between antenna size and propagation distance. Extremely low power

radios will not be ideal in this case as there will not be a receiving node close by. To overcome the attenuation from vegetation, the path through the vegetation should be as short as possible. Therefore high sites need to be used. The animal-borne radio should have enough power to be able to reach the nearest high-site. Adding repeater nodes to this will make the radio links shorter and the necessary radio power less. The antennas used by the repeaters need at least to stand out above the average vegetation height. This could be done quite easily using a 2 meter high mast. Painting the mast black or brown will camouflage it in the surroundings and not attract attention. A monopole antenna is quite thin (few millimetres) and won't be easily seen. The idea is sketched in figure 9.1.

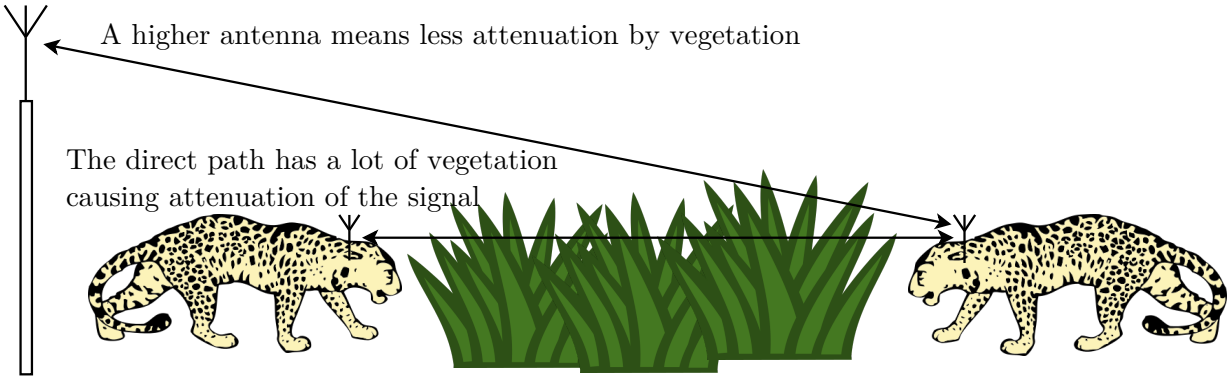


Figure 9.1: A graphical depiction why inter-animal communication is a bad idea, while communication with a repeater node with a higher antenna is better.

When deciding on the locations of sites for repeaters and base stations, simulations are normally used. From the results obtained the recommendation is that simulations should be done very conservatively, using a power output of at least 20dB lower than the actual, and using a radio sensitivity of just above the noise floor (-87dBm in our case). Reflections may cause the actual range to be better than this conservative simulation, but at least the worst-case coverage can be seen in advance.

9.2 Summary of Technological Extensions

A number of topics were dealt with in this project that can be regarded as advancements in the field of wildlife tracking and monitoring. These can be summarised as follows:

- Development of an extremely low power sensor module that can be used as a mortality sensor. The device is still versatile enough to be used for a number of other sensor and wireless network uses and can be manufactured at a relatively low cost.
- An APRS and AX.25 based VHF data link was proven to be sufficient for data communication in a wildlife monitoring network. An addressing scheme that will fit in well with this field was defined.
- Coverage measurements proved that VHF is a suitable frequency for wildlife monitoring, but that the effect of attenuation by vegetation can hugely decrease the performance and need to be taken into account.

- Guidelines to follow during radio coverage simulations were set up after comparing real-world measurements by a number of preliminary simulations.
- Algorithms were designed to do logging of motion data at constant sample rates, as well as logging of data at very low power usage levels.
- An electronic steerable antenna prototype was designed and built that can be used in the optimisation of wildlife tracking networks.
- Simulation of an antenna on the leg of an animal was done and can be seen as a novel result.

9.3 Future work

A number of improvements are already mentioned in section 9.1. They are however focused on specific parts of the system. When looking at the overall design there are still a number of design changes and expansions that can be made.

The current design goes with the assumption that full radio coverage of a region is possible, and that a tracking collar on an animal will almost always be able to transmit a message to the network of radios. In practice this might be impossible. The system design will need to be optimised to forward data whenever the possibility arises. A tracking collar should listen for repeater nodes that are close, and as soon as the collar hears a repeater station it can send out its captured data. In the case where an emergency transmission is to be made, one would either have to have a reserve of power that can transmit a high energy burst, or a separate radio or satellite modem that can be powered up in the exceptional case of an emergency.

The inclusion of electronic steerable antennas in the network can help increase the range of repeater stations. They can constantly change the direction of the antenna's main lobe, finding from what direction a transmission is coming, increasing the reliability of the radio link as well as possibility triangulate the source of the transmission.

All changes to the network are dictated by the capabilities of the radio, modems and protocol that are used. As was previously said, APRS is sufficient for the use in wildlife tracking networks, but it can be improved. Especially the physical layer and its immunity to noise can be improved. The bandwidth usage can also be improved. When a better physical layer has been found, a protocol stack needs to be chosen or created. At this stage the "emergency broadcast" and "antenna steering" functionality should be built in.

More work is also needed on the provision of electrical power, especially on the animal borne equipment. Tests need to be done to see how much energy can practically be obtained from solar cells, both on the animal and in the field.

9.4 Final word

This thesis proved that AX.25 and APRS is an easily implementable solution for wildlife monitoring. Even though it is not ideal, it can be put into practice using affordable and widely available voice FM radios. These type of voice communication networks already exist in wildlife

reserves. Existing infrastructure can therefore be used and slightly augmented to form a voice and data hybrid network.

As wildlife conservation is an area which is normally run on a non-profit principle, this affordable solution is ideal. At the current increasing rate of poaching in Africa, an already working system that can be rolled out immediately with an as small as possible capital layout, is preferred.

...
But there's one sound
That no one knows
What does the fox say?
...
Big blue eyes
Pointy nose
Chasing mice
and digging holes
...
Tiny paws
Up the hill
Suddenly you're standing still
...
Your fur is red
So beautiful
Like an angel in disguise
...
The secret of the fox
Ancient mystery
Somewhere deep in the woods
I know you're hiding
What is your sound?
Will we ever know?
Will always be a mystery
What do you say?

You're my guardian angel
Hiding in the woods
What is your sound?
...
Will we ever know?
...
I want to
...
I want to
I want to know!
...

— Ylvis, *The Fox*

Appendices

§ A. Accelerometer Logger

This appendix describes the process followed to design a device that can log accelerometer measurements at a constant sample rate of 100Hz.

A.1 SD card with FAT file system, log after every sample

An initial experiment that was done to log accelerometer data, used a SD card formatted with the FAT file system. A sample was read from the accelerometer, then stored on the SD card. This was repeated immediately after the SD card writing routines were completed. The system ran overnight and logged accelerometer data as quickly as it could. The time it took to log a sample is shown in graph A.1 against the number of samples already logged on to the SD card.

As can be seen in graph A.1 the time it takes to store a sample on the SD card increased with the number of samples already stored on the SD card. The accelerometer provides a valid sample every 10ms, so the major part of the delay is due to the SD card writing.

The FAT file system needs to keep a table of where data is written to the SD card. This table needs to be updated every time data is written. As the total amount of data that was written increases, the table grows. Every time the table is updated it needs to be read from the SD card, updated, and written back to the SD card. Therefore the more samples that are stored on the SD card, the bigger the FAT table and the longer it takes to read and write it to the SD card. This is the most likely reason for the increase in logging latency in graph A.1.

The delay caused by the SD card writing is in any case too large if we want to be able to log at a constant sample rate of 100Hz.

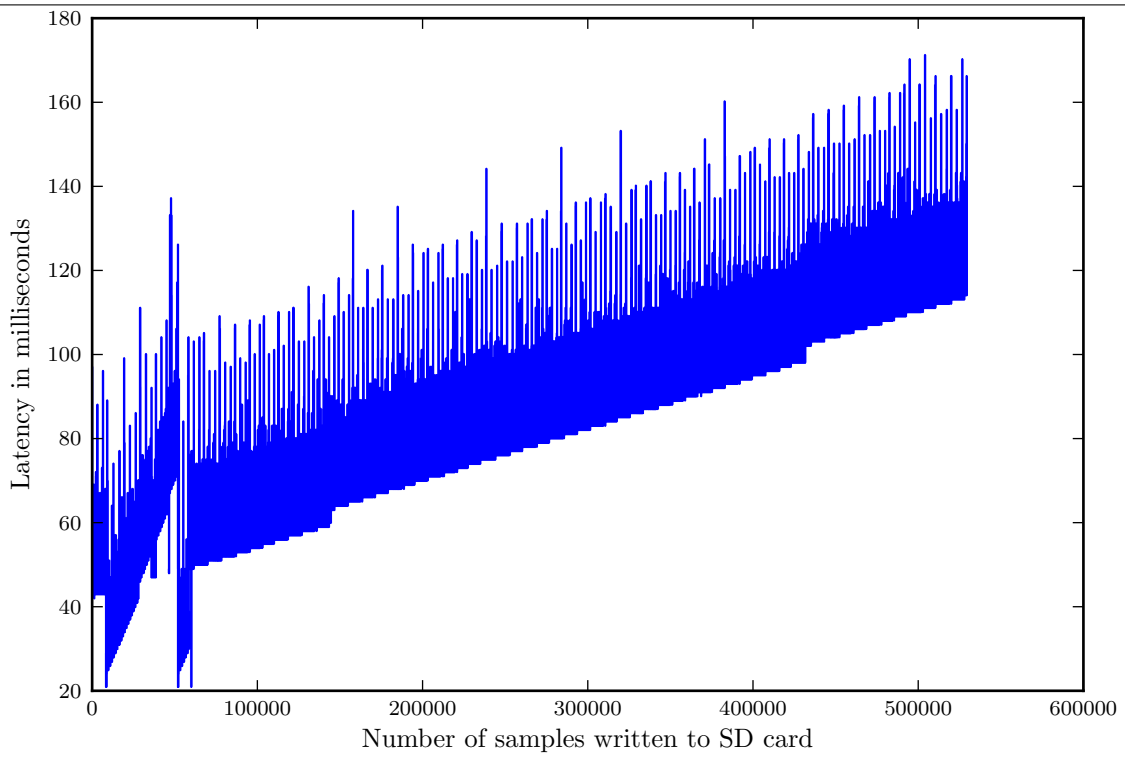
A.2 SD card without any file system, log after every sample

Graph A.2 indicates the increase in delay when no file system is used on the SD card. Data is directly written to specific addresses on the SD card. The delay is still too large to sample and store values at 100Hz. There is also still an increase in the latency as more samples are stored onto the SD card.

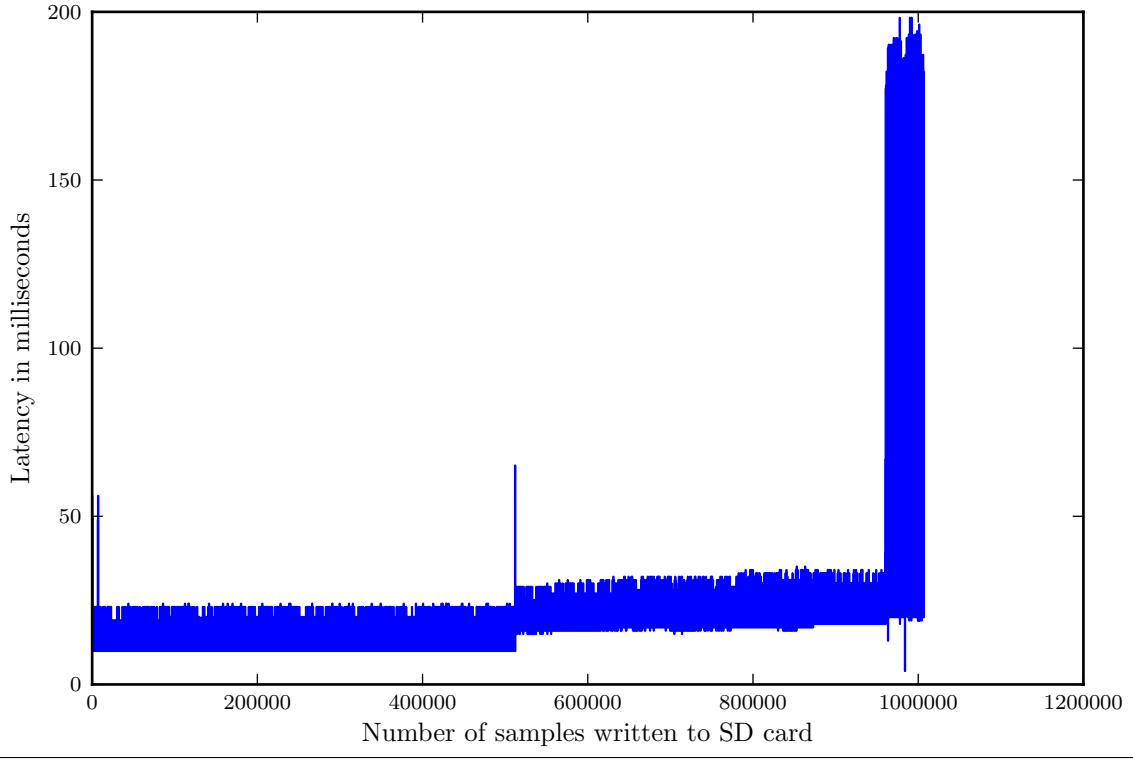
The reason can be that cheaper SD cards can access storage at lower addresses quicker than at higher addresses. This is unfortunately quite difficult to prove.

The test was repeated a second time, of which the result can be seen in graph A.3. The same general form of the graph is observed with a minimum latency of around 12ms. A spike in the latency is seen during logging of the first number of samples. The sudden increase just before a million samples is not seen any more. These seemingly random changes in the latency can be attributed to wear levelling and sector remapping that is done internally on the SD card. Flash memory only has a limited number of read-write cycles. If one is to constantly write to the same sectors (the start of the SD card in our case), those sectors will die earlier than the rest of the card. Wear levelling is done to spread the load across all sectors. If a sector does stop

Graph A.1: Latency to store a sample versus number of samples stored on the SD card, using the FAT file system. Every sample is stored immediately before another sample is taken.



Graph A.2: Latency to store a sample versus number of samples stored on the SD card, using no file system. Every sample is stored immediately before another sample is taken.



functioning, that memory address is remapped to a different physical sector. Moving sectors takes time and, from our viewpoint, can happen at any random time.

Up to now a cheap class 4 SD card was used for all the testing. Using a more expensive class 10 SD card from a different manufacturer theoretically should provide lower latency. In graph A.4 the latency results are shown for the test using a class 10 SD card. Rather than having increasing steps in latency, the latency is uniformly spread out in a wider range of times. The minimum latencies measured increased from around 12ms to around 30ms.

A.3 Aggregate samples, then write

The sectors on flash media, including SD cards, are blocks of 512 bytes. It is not possible to write or read a single byte from a SD card. The entire 512 byte sector needs to be read into memory, changes applied in the buffer in memory, and then the entire sector of 512 bytes written to the SD card again. This is another reason for the high latency observed during the previous tests. Every time a sample is written to the SD card, an entire sector needs to be read, modified and written again. By writing blocks of 512 bytes on 512 byte aligned addresses the reading of the sector can be skipped.

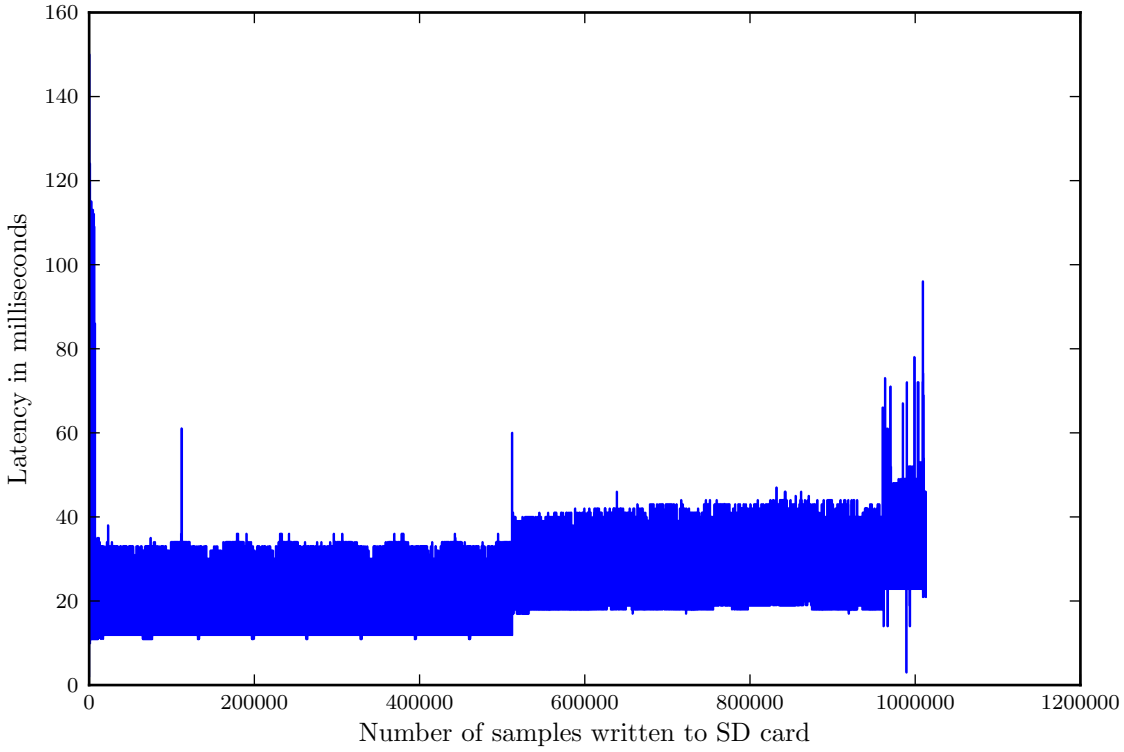
The improved system aggregates samples until a 512 byte buffer is full and then writes that buffer to the SD card. The latency results for this approach can be seen in graph A.5. The minimum latency between samples obtained using this approach is much lower, as two samples can be measured directly after each other, without waiting for the SD card. Unfortunately after the buffer is full (in our case after 26 samples) the buffer needs to be written to the SD card. Comparing the maximum values in graph A.5 with those in graph A.4, we can see that there is however a decrease in the peaks in the latency graph when using aggregated logging. This is attributed to the fact that we do not have to read the sector before we can write the data. Even though the minimum latency decreased to well below the 10ms (100Hz) goal, we still do not have a constant sample rate.

By limiting the maximum rate of sampling, the buffer grows a little slower and the SD card is given a little more time to write the data. The result is a latency graph that is more uniform with a lower average latency. It can be seen in graph A.6. There are however still spikes of high latency and a constant sample rate is still not achieved. The sampling is limited to 100Hz, but latencies of less than 10ms are measured. This is due to a sample that is taken after writing to the SD card being “late”. The following sample is taken a bit sooner to compensate. This effect can be seen in figure A.1. The average sampling rate is now closer, and almost exactly, 100Hz, but it is still not a constant sample rate.

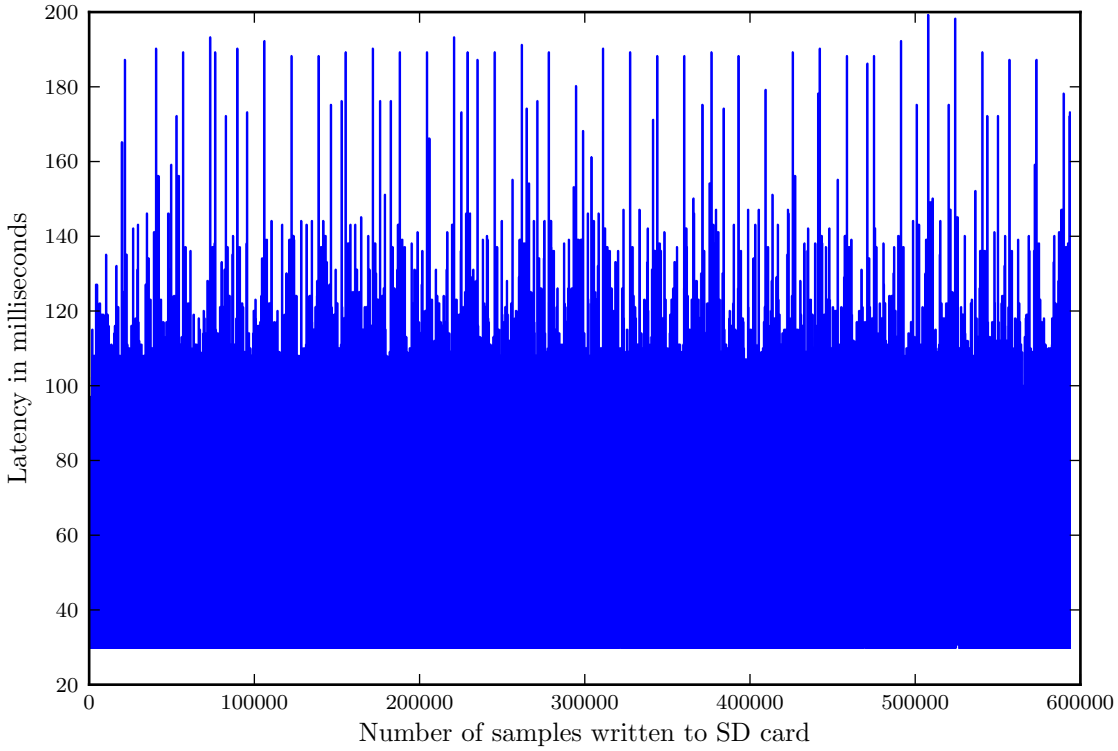
A.4 Aggregate samples, write, but continue sampling

The next evolution of this system is by using separate program threads for sampling and logging respectively. A 100Hz timer is used to read a sample from the accelerometer and add the sample to the buffer. The timer causes an interrupt to fire, and the appropriate routine to execute, halting all other tasks on the microcontroller. This can be seen as one of the threads.

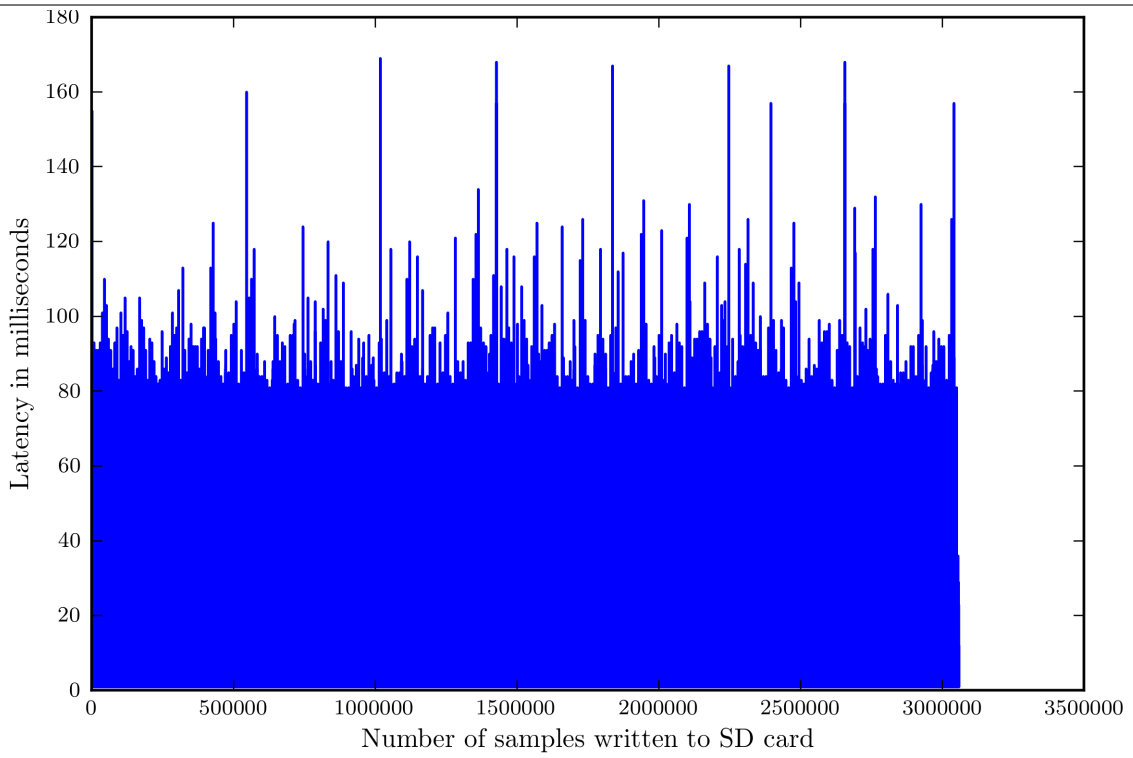
Graph A.3: Latency to store a sample versus number of samples stored on the SD card, using no file system. Every sample is stored immediately before another sample is taken. Second tests.



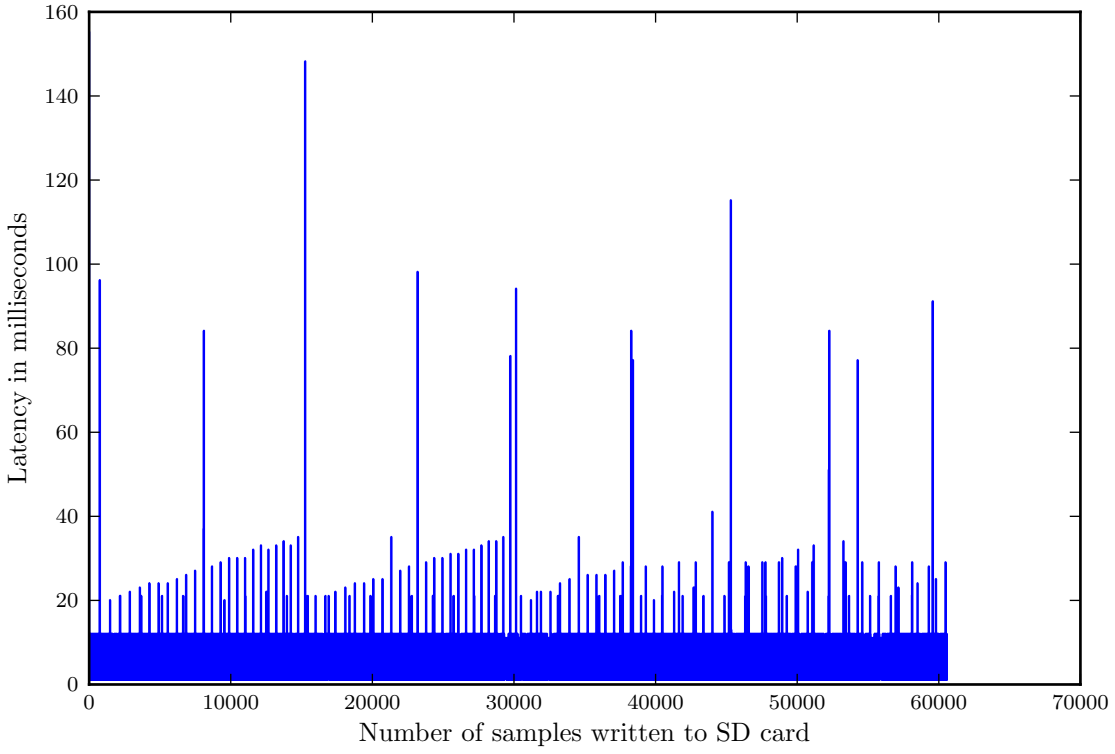
Graph A.4: Latency to store a sample versus number of samples stored on the SD card, using no file system. Every sample is stored immediately before another sample is taken. Third tests using a class 10 SD card.



Graph A.5: Latency to store a sample versus number of samples stored on the SD card, using no file system. Samples are aggregated until a 512 byte buffer is full and then the buffer is written to the SD card. Samples are taken as quick as possible.



Graph A.6: Latency to store a sample versus number of samples stored on the SD card, using no file system. Samples are aggregated until a 512 byte buffer is full and then the buffer is written to the SD card. Sampling is limited to 100Hz.



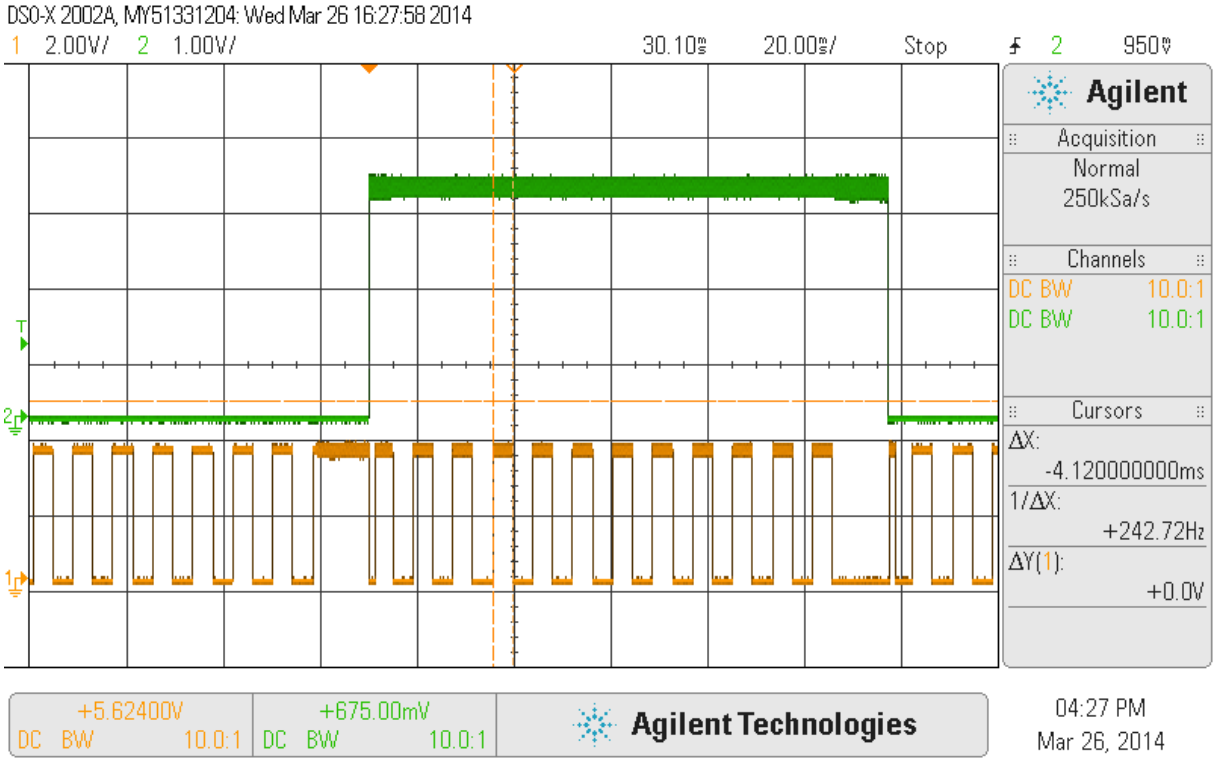


Figure A.1: An oscilloscope trace for the sampling times and logging times. Sampling was done as quickly as the accelerometer could provide data and is indicated by the transitions in the orange line. As soon as 26 samples are buffered it is written to the SD card, which on completion is indicated by a transition in the green line. The variation in sample rate due to delay in SD card writing can be clearly seen between a green line transition and the last orange line transition before it, an the two orange line transitions following it.

In the background there is an endless while-loop executing, checking to see if the buffer is full or not. If the buffer is full, the data is written to the SD card. This is seen as the second lower priority thread, that can be stopped at any moment to execute the sampling thread.

This method works well to obtain a constant 100Hz sample rate. However, sometimes the main thread that copies the data from the buffer to the SD card takes too long. In such a case the data in the buffer is overwritten by newer samples before it is written to the SD card. The order of samples get mixed up, but worse is that samples are overwritten and lost.

In some cases the sampling thread interrupts the logging thread at the worst possible moment, during copying, and the entire buffer gets corrupted. As the entire buffer is corrupted, all the containing samples are lost.

To combat this issue the principle of double buffering can be used. A separate buffer should be used for storing new samples, while another buffer is being copied to the SD card. Unfortunately the ATmega 328P only has 2048 bytes of RAM available in which buffers can be held. With all the overhead of the SD card, accelerometer and UART libraries, not much more than 600 bytes remain to be used as buffer. Therefore double buffering is impossible.

Even though two complete buffers can not be kept, there is still enough space to buffer a single sample. Let's call this partial double buffering. While the main buffer is copied to the SD card, the first sample taken by the interrupt routine is not copied to the buffer, but to a separate temporary storage space. From the second sample onwards the samples are again written to the

main buffer. When the buffer is almost full, the first element that is stored in the temporary storage is copied to the start of the buffer. This method allows the SD card copying process to use up to twice the sampling rate ($50Hz \Rightarrow 20ms$). The temporary storage can be enlarged up to a point where no more RAM is available. For every extra sample that can fit into the temporary storage, another 10ms is available to the SD card writing routine.

This method proved to be effective but sometimes, though rarely, a sample is still missed. Lowering the sample rate allows the SD card more time. A better approach would however be to increase the amount of RAM, or saving RAM in other areas in the code to allow for full double buffering.

A.5 Layout of a data sample

The chosen layout for a sample as it is written in raw form to the SD card is shown in table A.1. The total size of a sample is 152 bits, or 19 bytes. We can therefore store up to 26 samples per sector on the SD card.

Field	Run ID	Time (ms)	Sample count	X data	Y data	Z data	Passive Motion triggers
Number of bits	8	32	32	16	16	16	32

Table A.1: The layout of a single sample as it is written to the SD card.

When reading the data back from the SD card, this layout, along with the data type and signedness must be known. A simple python script was implemented using the struct library's unpack function and can be seen in listing A.1. The raw bytes are read from the SD card by this script and converted to python number formats. It is then printed to the standard output as CSV (comma separated values) data for further processing by other software. Using this method it can be piped directly into another program using Unix inter process piping.

A.6 Accelerometer read method

The accelerometer can sample the three axes of acceleration it can measure at different rates and different resolutions. Depending on which accelerometer is used, the available rates and resolutions are different. For the Analog Device ADXL345 a resolution of 13 bits at a $\pm 16g$ scale was used. This translates to a value of 256 for a 1g acceleration. Further, the accelerometer was set up to provide samples at a rate of 100Hz. This setting needs to be programmed into the accelerometer. The accelerometer needs to enable a 50Hz low pass filter to band limit the signal so that it can be sampled within the Nyquist limits.

Another characteristic of accelerometers to keep in mind is that the three axes of acceleration do not necessarily measure a value of 0 for a zero acceleration. At startup the three axes need to be calibrated for zero DC offset. This can be done by putting the device on a level surface

Listing A.1: A python script to read samples from the SD card.

```
#!/usr/bin/python
import sys,os
import struct
from pprint import pprint

disk = file("/dev/sdb",'rb')
sector_size=512

disk.seek(0x40000)
sector = disk.read(sector_size)

lastWriteLocation = struct.unpack('>Q',sector[0:8])[0]
runID = struct.unpack('>l',sector[8:12])[0]
loops = struct.unpack('>l',sector[12:16])[0]

# print "Last run ID: "+str(runID)
# print "stored at location: "+str(lastWriteLocation)
# print "iterations done: "+str(loops)

# print ""

for sectorNr in range(1, loops):

        #while True:
        disk.seek(0x40000+(sectorNr*sector_size))
        sector = disk.read(1*sector_size)

        #print "Length sector: "+str(len(sector))
        #pprint(sector)

        for blockNr in range(0, 25):
                block = sector[blockNr*19:blockNr*19+19]

                #print "Length block: "+str(len(block))
                #pprint(block)

                runID = struct.unpack('>B',block[0:1])[0]
                uptime = struct.unpack('>L',block[1:5])[0]
                loopCount = struct.unpack('>L',block[5:9])[0]
                x = struct.unpack('>h',block[9:11])[0]
                y = struct.unpack('>h',block[11:13])[0]
                z = struct.unpack('>h',block[13:15])[0]
                passive = struct.unpack('>L',block[15:19])[0]

                if(runID!=0):
                  print str(runID)+','+str(uptime)+','+str(loopCount)+','+
                    str(x)+','+str(y)+','+str(z)+','+str(passive)

disk.close()#close the device
```

and calibrating for a 1g acceleration downwards due to gravity. This is normally the Z-axis. The other two axes should be calibrated to read 0, while the device is kept still.

A good approach will be to put the device down on all sides consecutively, measuring +1g and -1g on all three the axes. This way any offset in gain can also be corrected.

The tests that were performed on the logging of acceleration data used gravity as a reference during startup. The offset of the axes were changed up or down until either 0 or 1g, depending on the axis, were read from the accelerometer for 3 successive samples. This iterative process was used to combat variations in the samples due to noise.

A.7 Timer setup

Timers on a microcontroller are implemented using a hardware counter module. The module is driven by a clock source which can be set to a number of factors of the main system clock. Every time the clock source completes a full clock cycle, the counter module increments the value in a specific register. With this knowledge we can calculate how many times the value in the register will be incremented per second. The basic layout of the counter hardware can be seen in figure A.2.

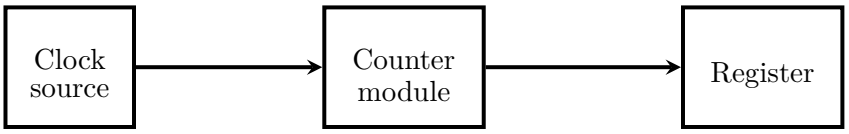


Figure A.2: A clock source, driving a counter, which increments the value in a register.

To execute code at a specific time we can set the system up to cause an interrupt when the value in the register becomes larger than a configured value. Resetting the value in the register to 0 every time the interrupt is triggered causes the interrupt to be executed at set intervals.

For logging the accelerometer data at 100Hz, we need an interrupt function to execute every 10ms. The hardware that was used had a main system clock of 20MHz. As an additional feature the interrupt can be executed every 1ms to act as generic 1ms timer for other purposes too. Every tenth execution the logging code can be executed.

The ATmega 328P has a couple of counters, of which we used counter 2 as it is the one that can operate in some of the microcontroller's low power modes[49, Table 9-1]. To set up the counter and interrupt, four registers need to be programmed.

TCCR2A Set the mode in which the counter should operate. Up counter, with register OCR2A as the value above which an interrupt should occur.

TCCR2B The divider for the clock source. We used 256 in this case to divide the 20MHz system clock down to 78125Hz.

OCR2A The value up to which the counter should count before the interrupt will be triggered.

TIMSK2 Enable the interrupt.

A clock of 78125Hz has a period of 12.8×10^{-6} seconds. The counter will therefore increment the register value every $12.8\mu s$. To get to 1ms we need to count to $\frac{0.001}{12.8 \times 10^{-6}} = 78.125$. Registers can only hold a positive 8 bit integer. Therefore we need to use the value 78 and make a small error. Register OCR2A gets the value 78.

Doing the calculation in reverse we get: $78 \times 12.8 \times 10^{-6} = 998.4\mu s$. Which is an error of $(1000 - 998.4) \div 1000 = 0.16\%$. During a 24 hour day we will make an error of 138.24 seconds. For our logging purposes this approximation is accurate enough.

The timing of the counter can be made more accurate by using an external quartz crystal to generate the main system clock. Keeping the above formulas in mind while choosing the crystal's clock frequency, a crystal can be chosen with a frequency that divides down better and allows for an error-less timer. When choosing the clock divider for register TCCR2B, the maximum value for an 8 bit register (256) must be kept in mind as that is the maximum value that can be stored in OCR2A. A smaller clock divider means a larger frequency and thus a larger value the counter needs to count to. A larger divider means a lower frequency, which gives us less counts per second and therefore a lower resolution of intervals the value of OCR2A can be chosen from.

A.8 Circuit diagram and PCB for the accelerometer logger

Below follows the PCB layout and the circuit diagram for the accelerometer logger.

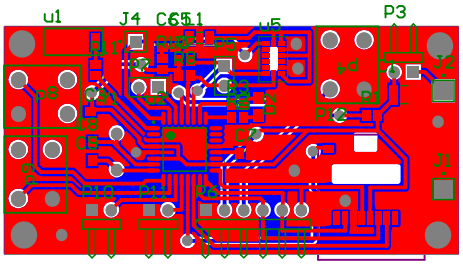
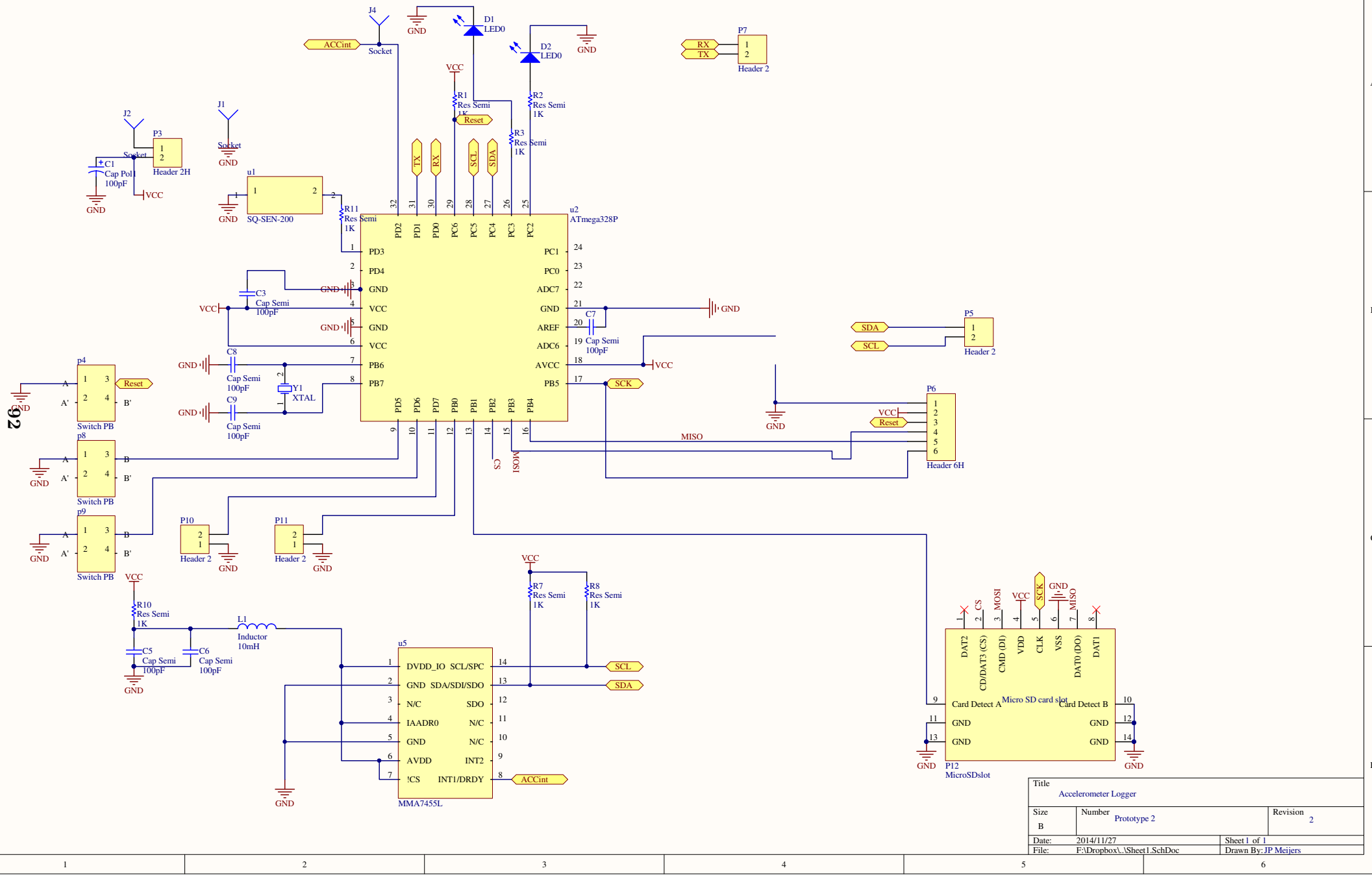


Figure A.3: PC board layout for the accelerometer logger.



§ B. Digital thermometer measurements

Digital thermometer IC's provide data in a format that is optimised for resolution in the room temperature range. Looking at the LM75A i^2c temperature sensor from Texas Instruments, which is used in the Sensor Module that was designed for this project, we see that the temperature is stored in the higher 9 bits of a 16 bit register (table B.1). To read it, two 8 bit register reads need to be performed over i^2c.

The higher 8 bits and the lower 8 bits are therefore read into two separate variables. One LSB translates to a value of 0.5°C. The formula for calculating the temperature from these two variables is as follows:

$$\text{Temperature} = \{[(\text{higher byte} \times 2^8) + \text{lower byte}] \times 2^{-7}\} \times 0.5 \tag{B.0.1}$$

The multiplication by 2^8, which must be implemented using a left bit shift, is necessary to get the MSB in the correct position for the software to interpret the value as a two's complement binary value. If it is not done, negative temperatures will not work but seen as very large positive integers. The values read from the sensor were stored in unsigned 8 bit variables, but the first part of the calculation is done by using integers, which are 16 bits wide in this case. This allows enough space for the bit shift to the left by 8 to be possible. The answer after division is stored in a double precision floating point variable so that decimal values can be kept.

A C-code implementation of the reading and calculation of temperature can be seen in listing B.1.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB	x	x	x	x	x	x	x

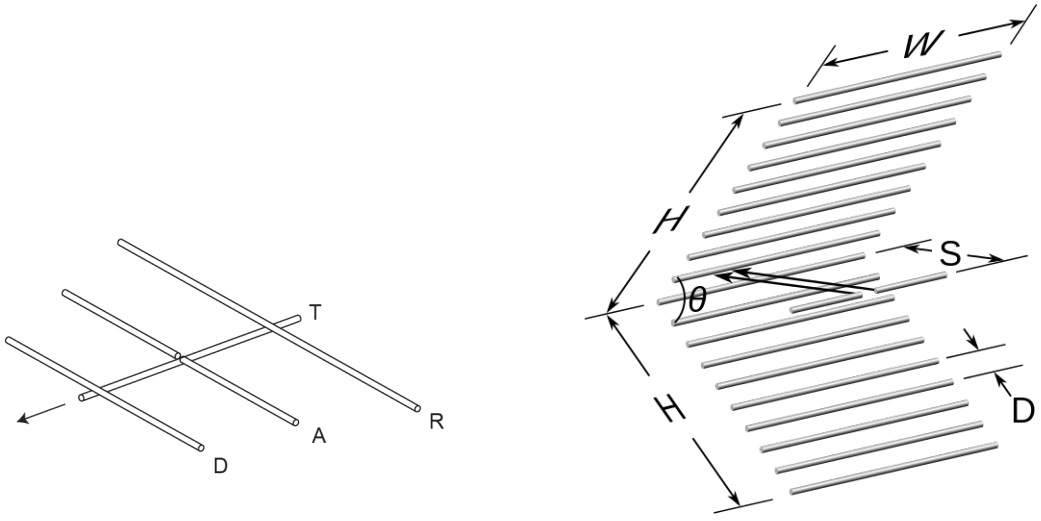
Table B.1: Layout of the LM75A temperature register.

Listing B.1: C-code to read and calculate the temperature from a LM75A sensor.

```
i2c_start_wait(0b10011000+1); // set device address and read mode
i2c_readAck(&MostSigByte);    // read one byte from sensor
i2c_readNack(&LeastSigByte);  // read one byte from sensor
i2c_stop();

// calculation of temperature
int temp = (((MostSigByte << 8) | LeastSigByte) >> 7);
double temperature = (((float)(temp))*0.5);

// convert the number to an ASCII representation
dtostrf(temperature, 5, 3, buffer);

// print the temperature out on the UART
uart_puts("Temp1: ");
uart_puts(buffer);
uart_puts("*C\t\r\n");
```

§ C. A simple electronic steerable antenna

Probably the simplest directional antenna is the Yagi-Uda antenna. It can be seen as a dipole with a reflector element behind it and a director element in front of it. An alternative to this is the so-called corner reflector antenna, which has at least three reflecting elements arranged to make a 90° angle, with a radiating element sitting inside this corner.



(a) A Yagi-Uda antenna with director D, reflector R and driven or active element A.

(b) A corner reflector with the driven element at position S.

Figure C.1: A Yagi-Uda and a Corner Reflector antenna.

Usually in a Yagi-Uda the reflector is a little longer and the director is a little shorter. We can compromise by making them both a little shorter, and then additionally add switches to short the element to ground, or make them float. When the element is connected to ground it will "push" the radiation pattern away from it. By doing this we can electronically switch in which direction the radiation pattern will be pushed.

Taking this further, we replace the dipole with a monopole above a ground plane. The other elements are also roughly halved in length. Now we add two more possible reflecting elements. The design looks like in figure C.2. Lengths and distances of the elements were optimised for a as low as possible S11 reflection coefficient at an impedance of 50Ω during simulation.

Shorting three of the outer four elements to ground, and letting the fourth float to be a director, the simulation tells us the radiation pattern will look as in figure C.3.

It can clearly be seen how the radiation pattern is pushed away from the grounded elements, while it is pulled towards the floating element. Unfortunately with this design the ground plane also pushes the radiation pattern upwards.

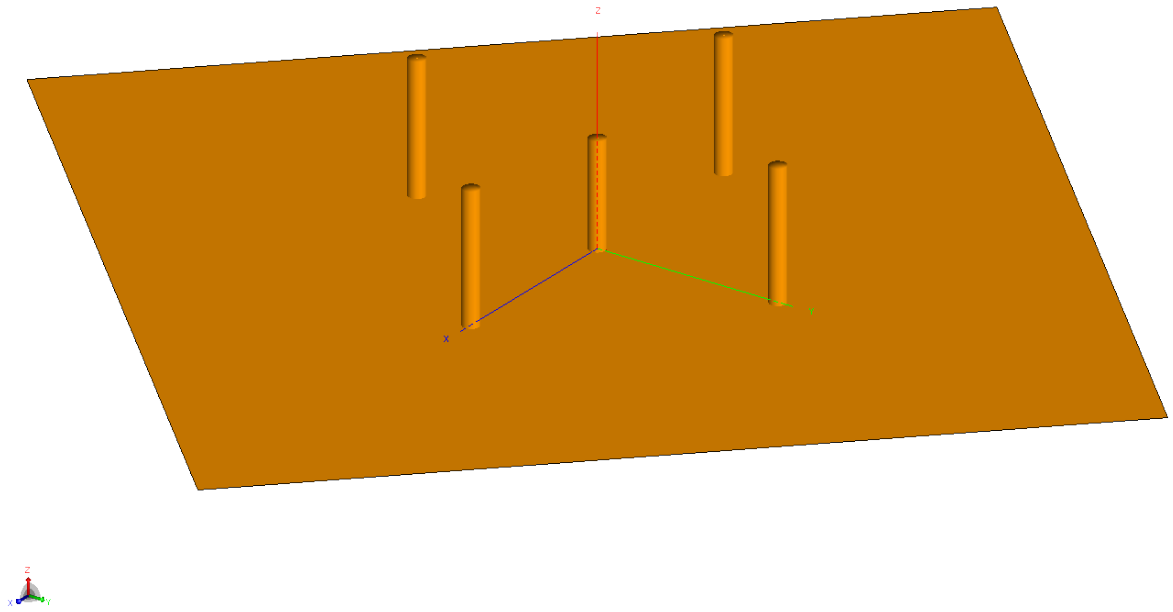


Figure C.2: Layout of antenna.

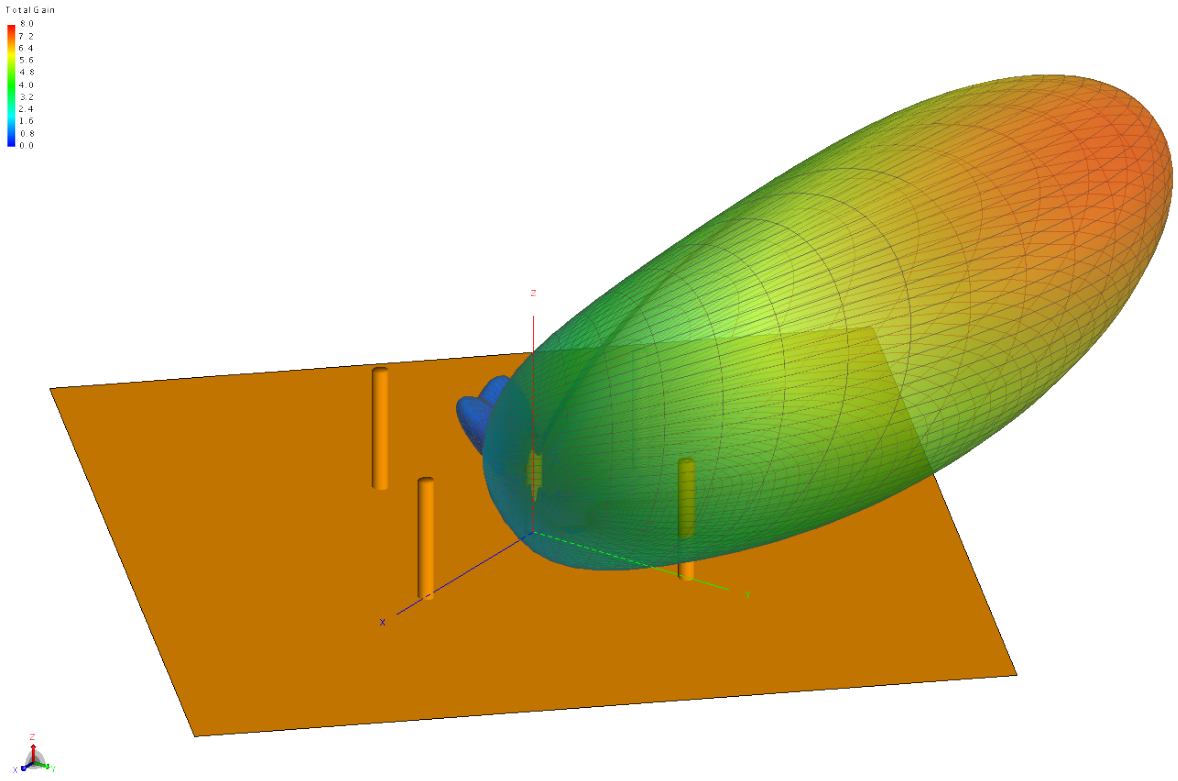
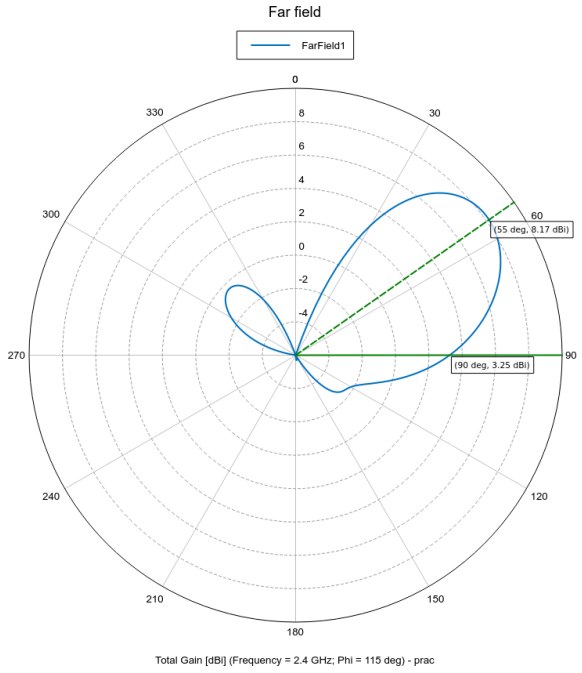
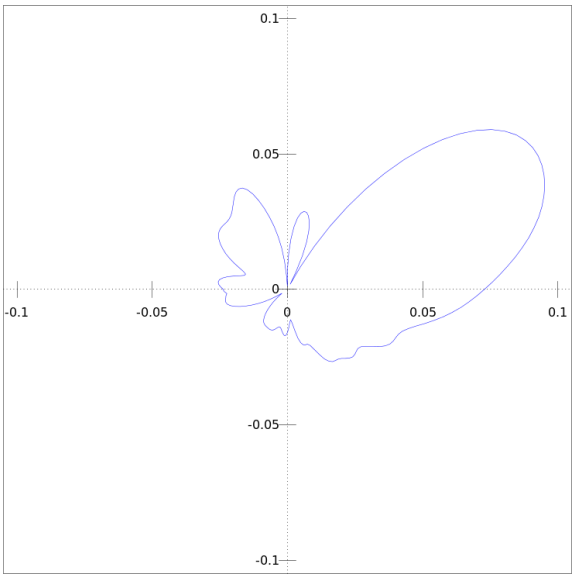


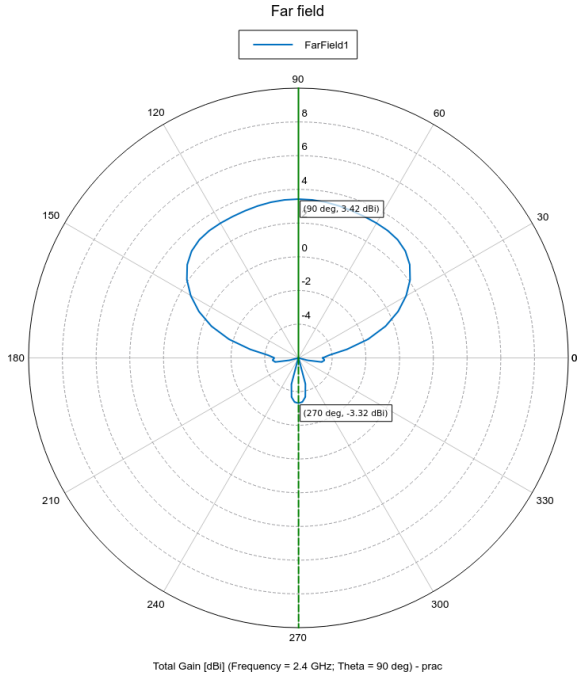
Figure C.3: Simulated 3D radiation pattern.



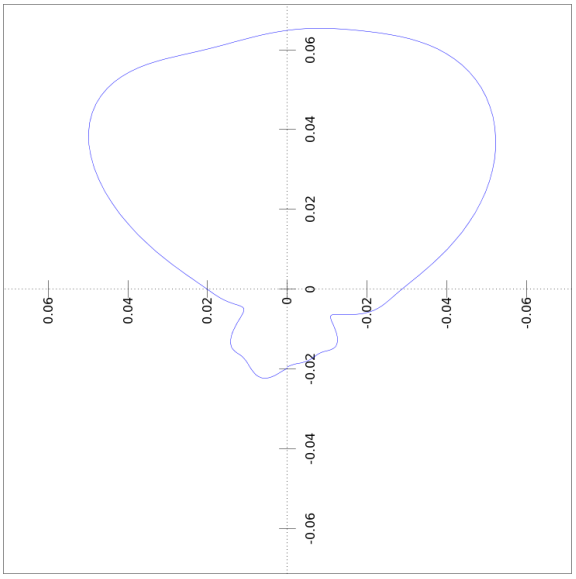
(a) Simulated elevation.



(b) Measured elevation.



(c) Simulated azimuth.



(d) Measured azimuth.

Figure C.4: Comparison of simulated and measured radiation pattern.

In figure C.4 the simulation results can be seen next to measured results obtained from analysis of the prototype antenna in an anechoic chamber. The measured results agree very well with the simulation.

For the 2.4GHz prototype the elements were about 3cm long and the outer elements were on a 12cm radius from the centre element. If this antenna would be built for 144MHz, the elements will become around 50cm in length, on a radius of about 4m. This is too large to be practical.

A design using a dipole in the centre could be investigated, but likely the radius of the outer elements will not change much.

§ D. Circuit Diagrams and PCB layouts

Listed below is firstly the PCB layout for the sensor node, followed by the PCB layout of the VHF node, then the circuit diagram for the sensor node and lastly the circuit diagram for the VHF node.

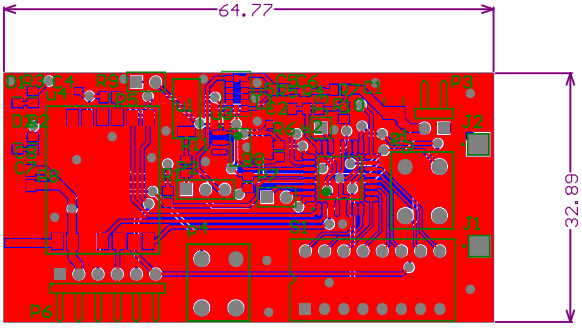


Figure D.1: The PCB layout for the sensor node.

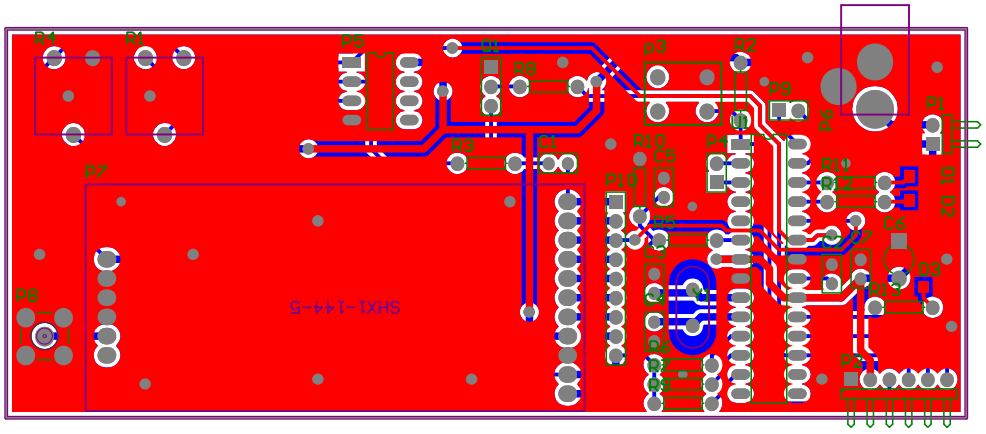
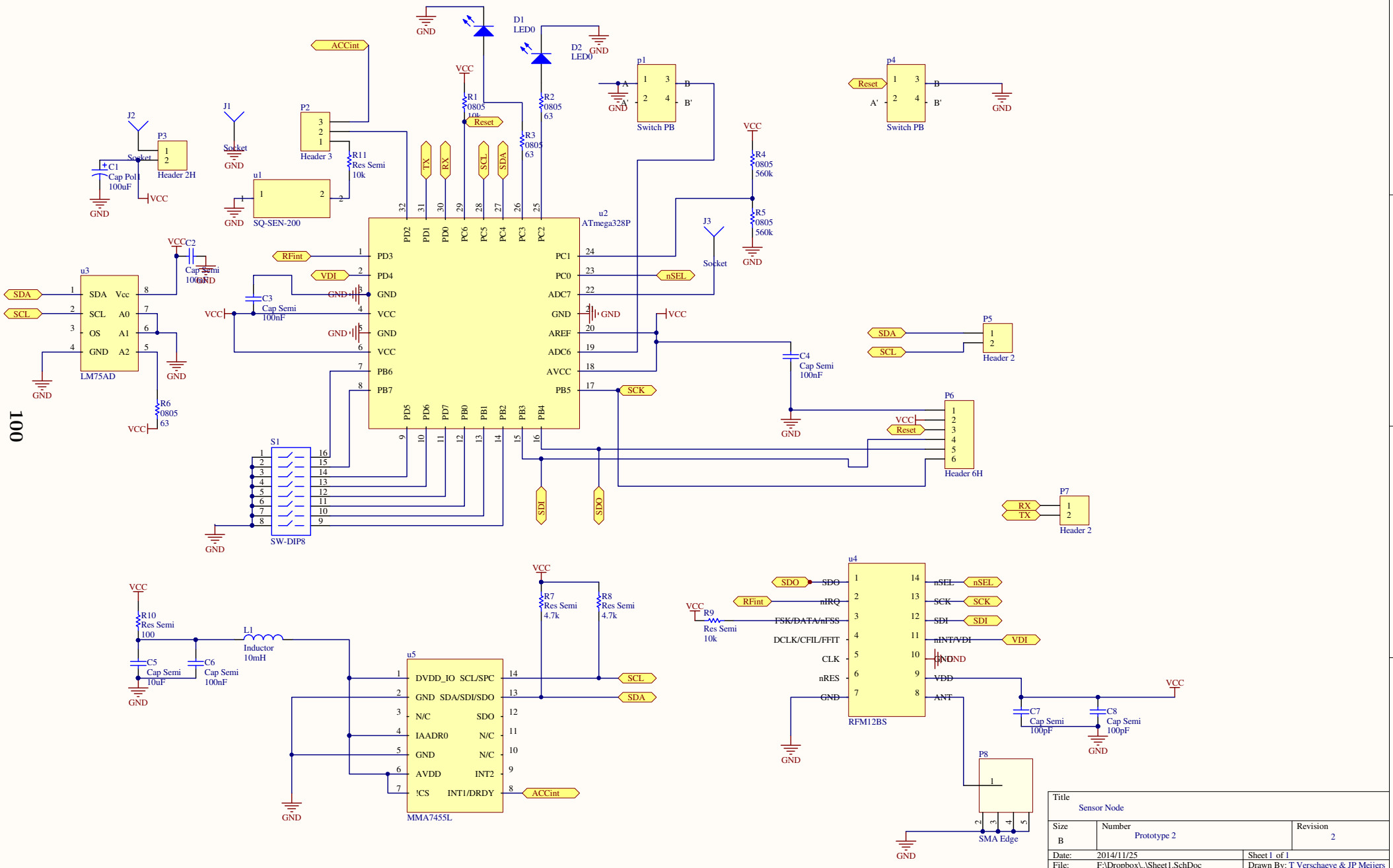
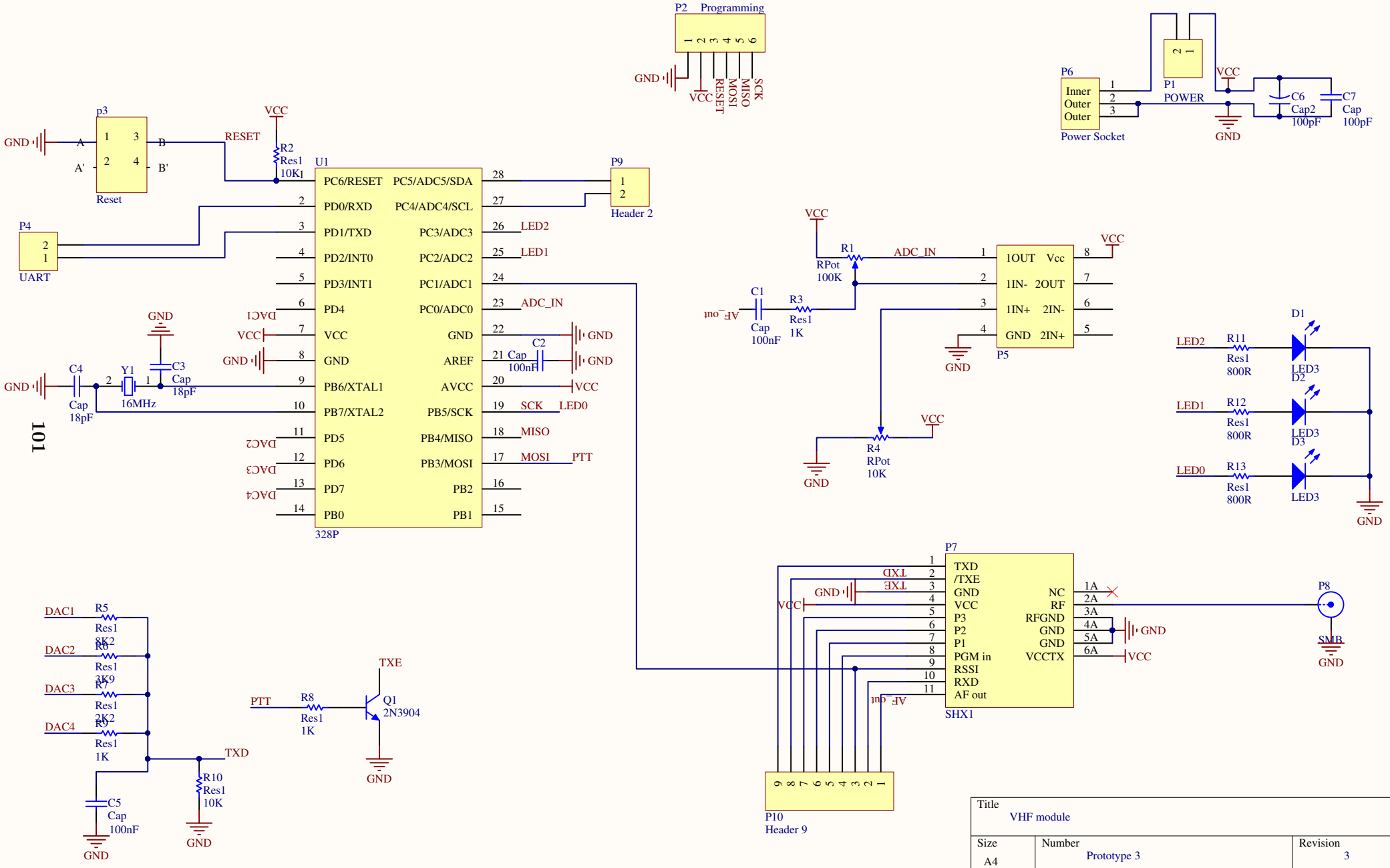


Figure D.2: The PCB layout for the VHF node.





Title: VHF module		
Size: A4	Number: Prototype 3	Revision: 3
Date: 2014/11/25		Sheet 1 of 1
File: F:\Dropbox\..\Sheet1.SchDoc		Drawn By: JP Meijers

§ E. More on radio coverage simulations

The first set of field measurements were gathered in the Jan Marais Park in Stellenbosch. It is a flat area with quite a bit of clutter. These characteristics lend itself perfectly to measure the influence of clutter on the strength of the radio signal. The results of the signal strength measurement can be seen in figure E.1.

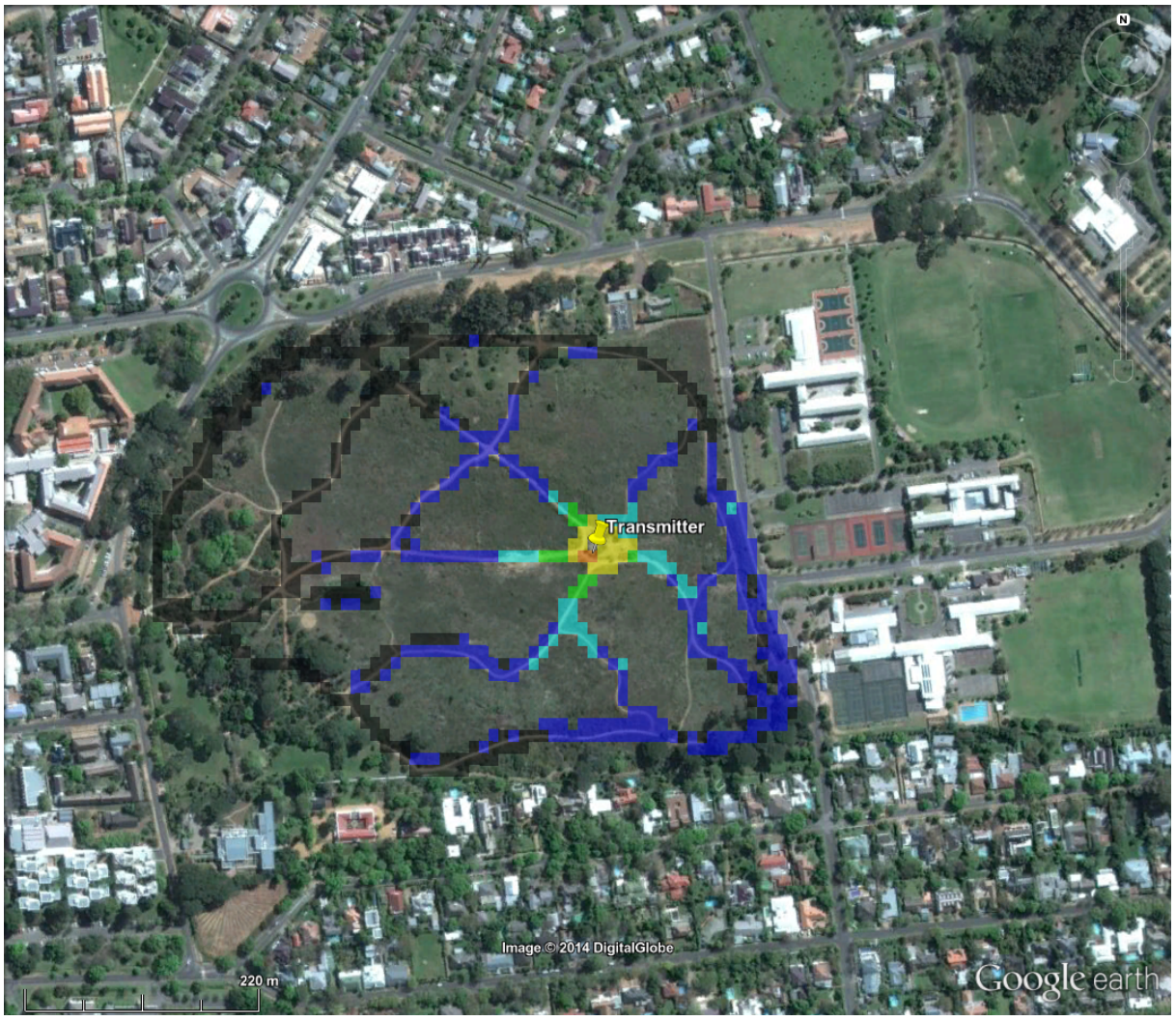
As the receiver moves, the signal varies due to different paths of propagation through the vegetation and reflections from the leaves and surroundings. Figure E.2 indicates the difference between the minimum signal strength and maximum signal strength measured per block of 0.0001°x0.0001°. A greater difference means the signal experienced more reflections and different paths through the clutter. An as low as possible variance is preferable. The darker tiles in the map in figure E.2 are therefore the better results. The observation can be made that the signal varies quite a bit close to the transmitter. This is due to the inverse square law. A few tens of meters further away the effect of the vegetation can be seen. Areas with more vegetation have a higher variation in the signal strength. On the outer areas of the park the variance in the signal is low. This is due to the signal not reaching those areas. Only the noise floor plays a role there.

The measured results are now compared to a number of simulations. Two different simulation engines are used. The first couple of results are from the free Radio Mobile radio coverage simulator. The last couple of results are from the commercial Pathloss package. Radio Mobile uses the Longley–Rice radio propagation model. This includes a percentage parameter for forest or urban obstructions. Pathloss gives the choice between three diffraction algorithms to compensate for obstructions. They are called Pathloss (the default or this package), Tirem and NSMA. These different simulations are now compared.

Figure E.3 shows a Radio Mobile simulation for the Jan Marais Park with a transmitter output power of 27dBm and a receiver sensitivity of -87dBm. This is the same output power that was used for the field test and the sensitivity of the receiver is set to just above the measured noise floor, as can be seen in graph E.1. Comparing the measured results in figure E.1 with the simulation, it can be seen that the simulation is far too optimistic. Using the same simulation parameters in the Pathloss package, using the default Pathloss model for obstructions, the result in figure E.4 is obtained. This is more optimistic than the Radio Mobile simulation.

Next the transmitter output power is lowered drastically to a level of 0dBm. This result can be seen in figure E.5. In this case the simulated result is too conservative, as would be expected. The output power used for the transmitter is slowly increased until the simulated coverage matches the measured result as best as possible. It turned out that a 7dBm transmitter output power for the simulation worked quite well to match the field measurement done at 27dBm. This means that an additional 20dB loss must be built into the simulation for a good match. The result of a Radio Mobile simulation done at an output power of 7dBm can be seen in figure E.6.

To compare the Radio Mobile and the Pathloss simulation engines, the same simulation parameters (7dBm output power, -87dBm receiver sensitivity) were used to do simulations with



(a) Measured signal strength

Colour	RSSI (dBm)
	< -87
	$-87 \leftrightarrow -80$
	$-80 \leftrightarrow -70$
	$-70 \leftrightarrow -60$
	$-60 \leftrightarrow -50$
	$-50 \leftrightarrow -40$
	$-40 \leftrightarrow -30$

(b) Legend

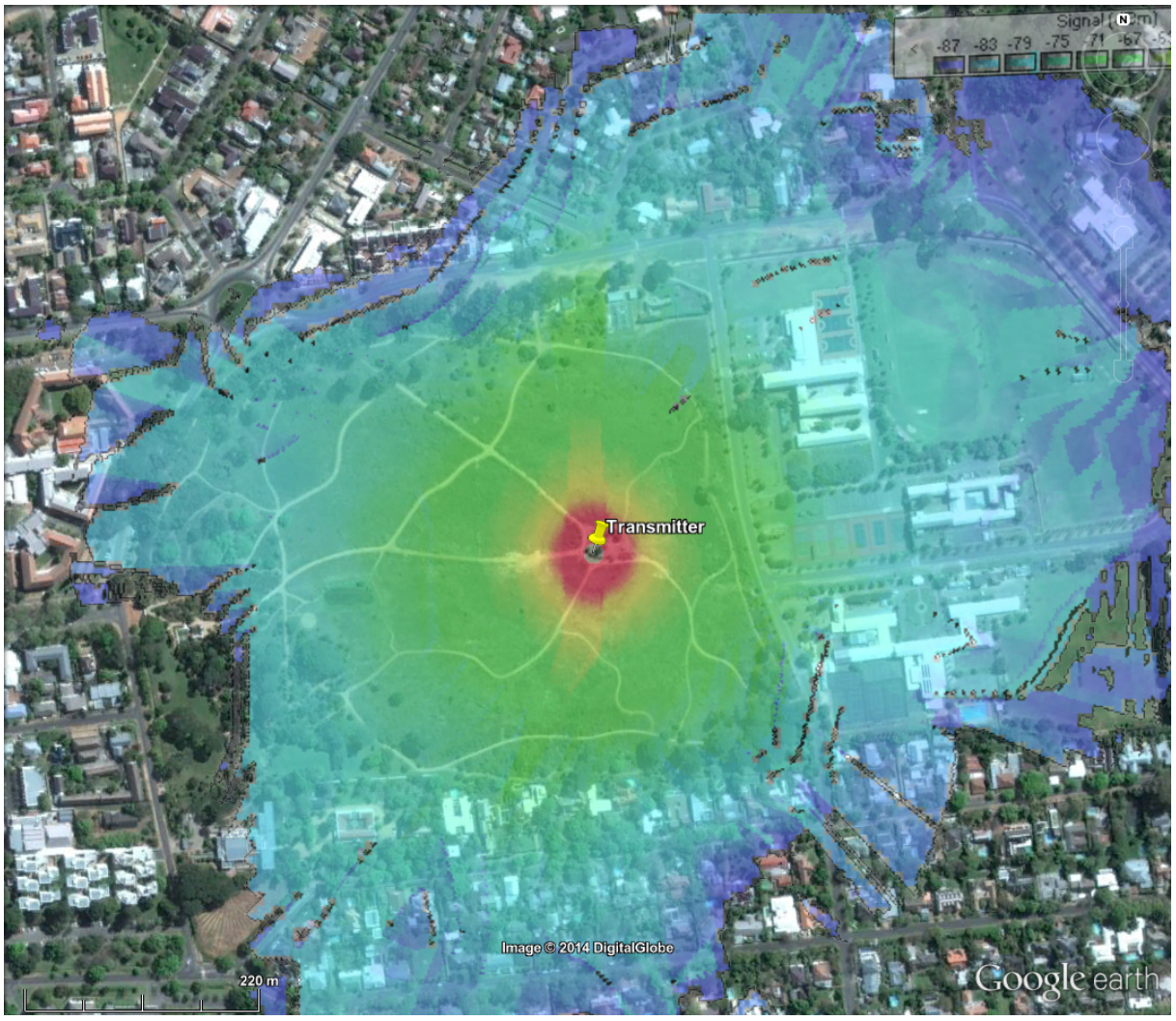
Figure E.1: Measured signal strength for a 27dBm transmitter in the Jan Marais Park.

(a) Measured signal variance

Colour	Difference between maximum and minimum signal strength (dB)
	< 1
	$1 \leftrightarrow 3$
	$3 \leftrightarrow 6$
	$6 \leftrightarrow 9$
	$9 \leftrightarrow 12$
	$12 \leftrightarrow 15$
	> 15

(b) Legend

Figure E.2: Measured signal variance for a 27dBm transmitter in the Jan Marais Park.

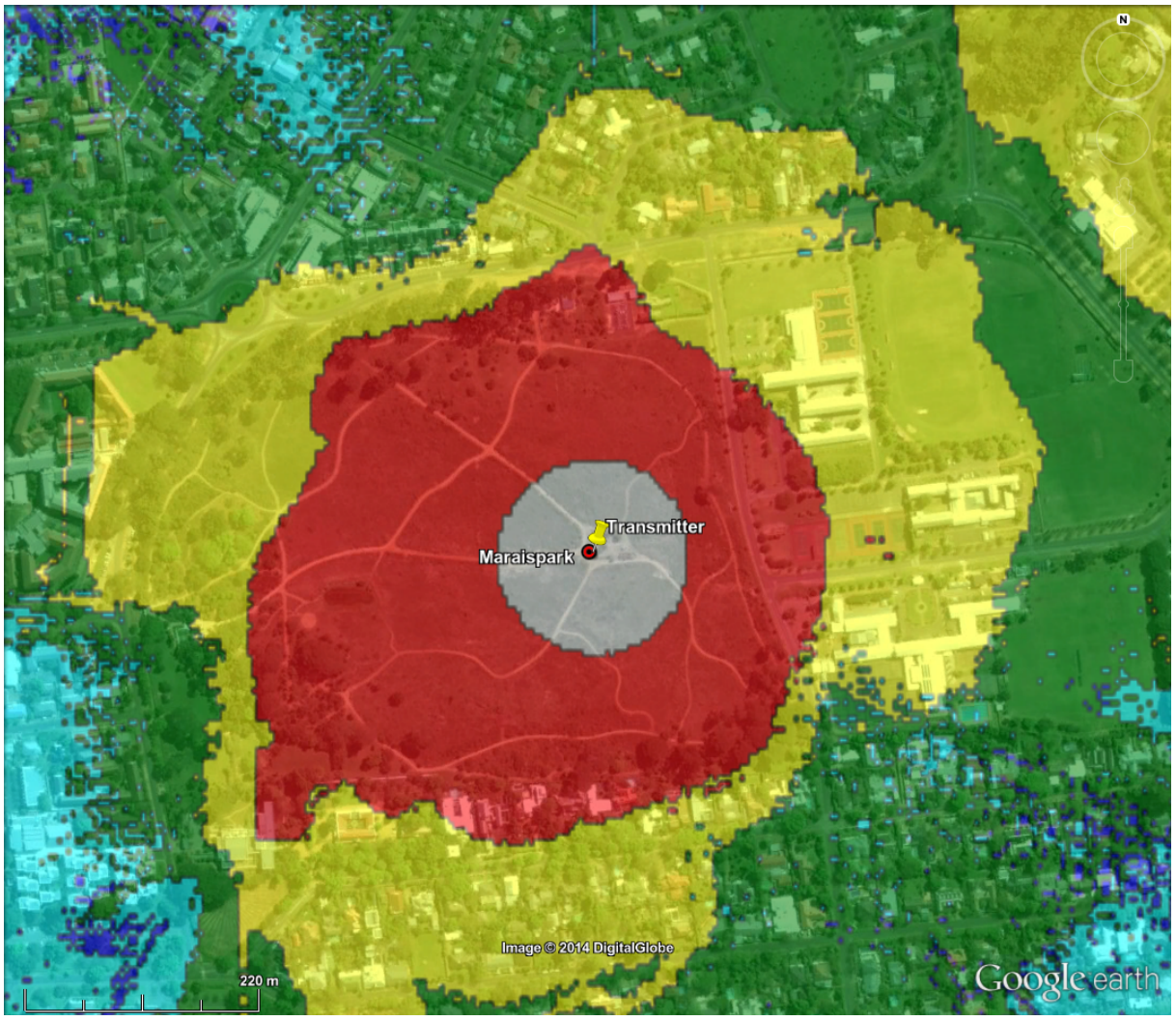


(a) Radio Mobile simulation at 27dBm transmit power.

Colour	RSSI (dBm)
	−87
	−83
	−79
	−75
	−67
	−63
	−59
	−55
	−51
	−47

(b) Legend

Figure E.3: Radio Mobile simulated coverage for a 27dBm transmitter in the Jan Marais Park.

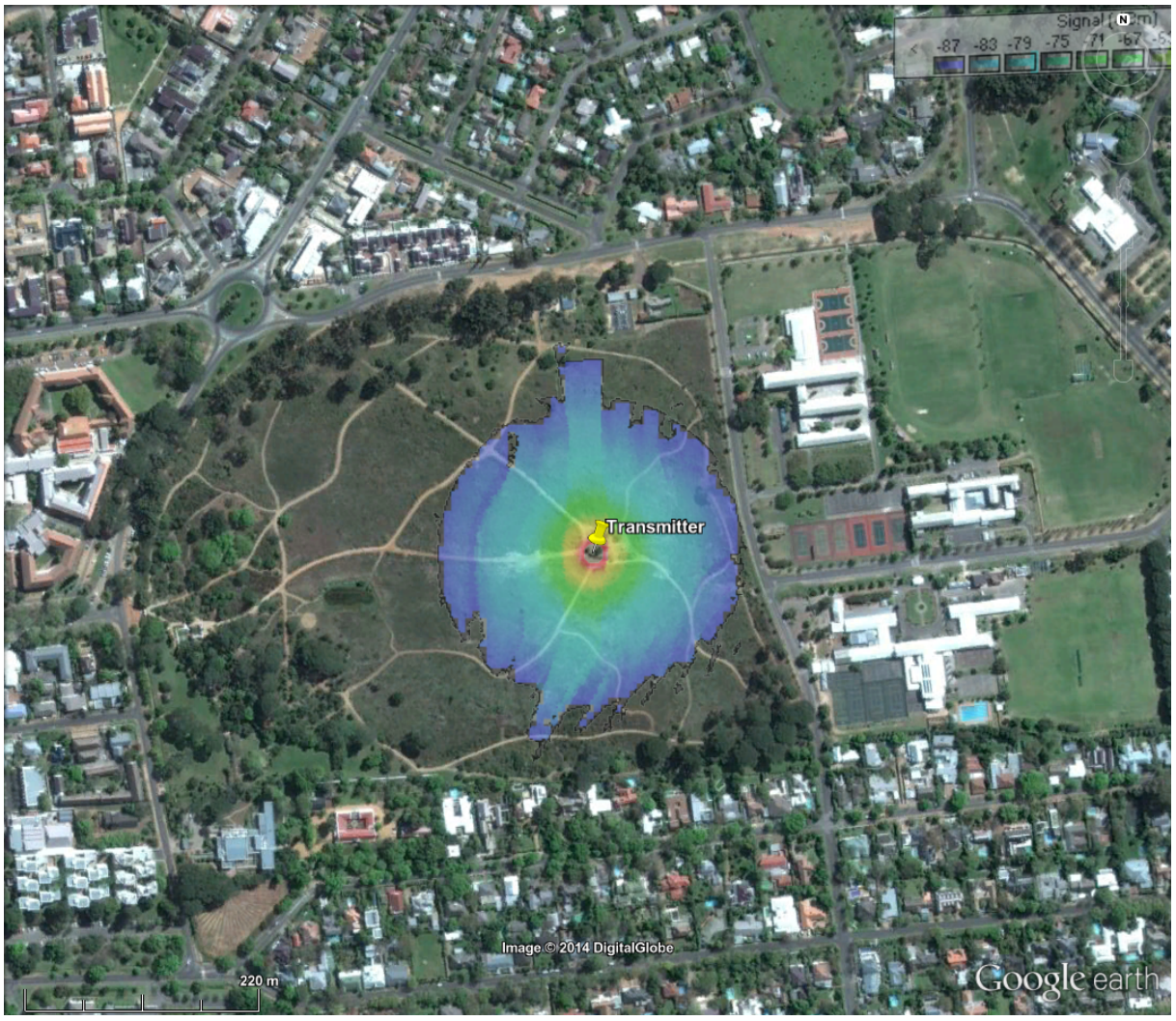


(a) Pathloss simulation at 27dBm transmit power.

Colour	RSSI (dBm)
	$-85 \leftrightarrow -110$
	$-75 \leftrightarrow -85$
	$-65 \leftrightarrow -75$
	$-55 \leftrightarrow -65$
	$-40 \leftrightarrow -55$
	> -40

(b) Legend

Figure E.4: Pathloss simulated coverage for a 27dBm transmitter in the Jan Marais Park.

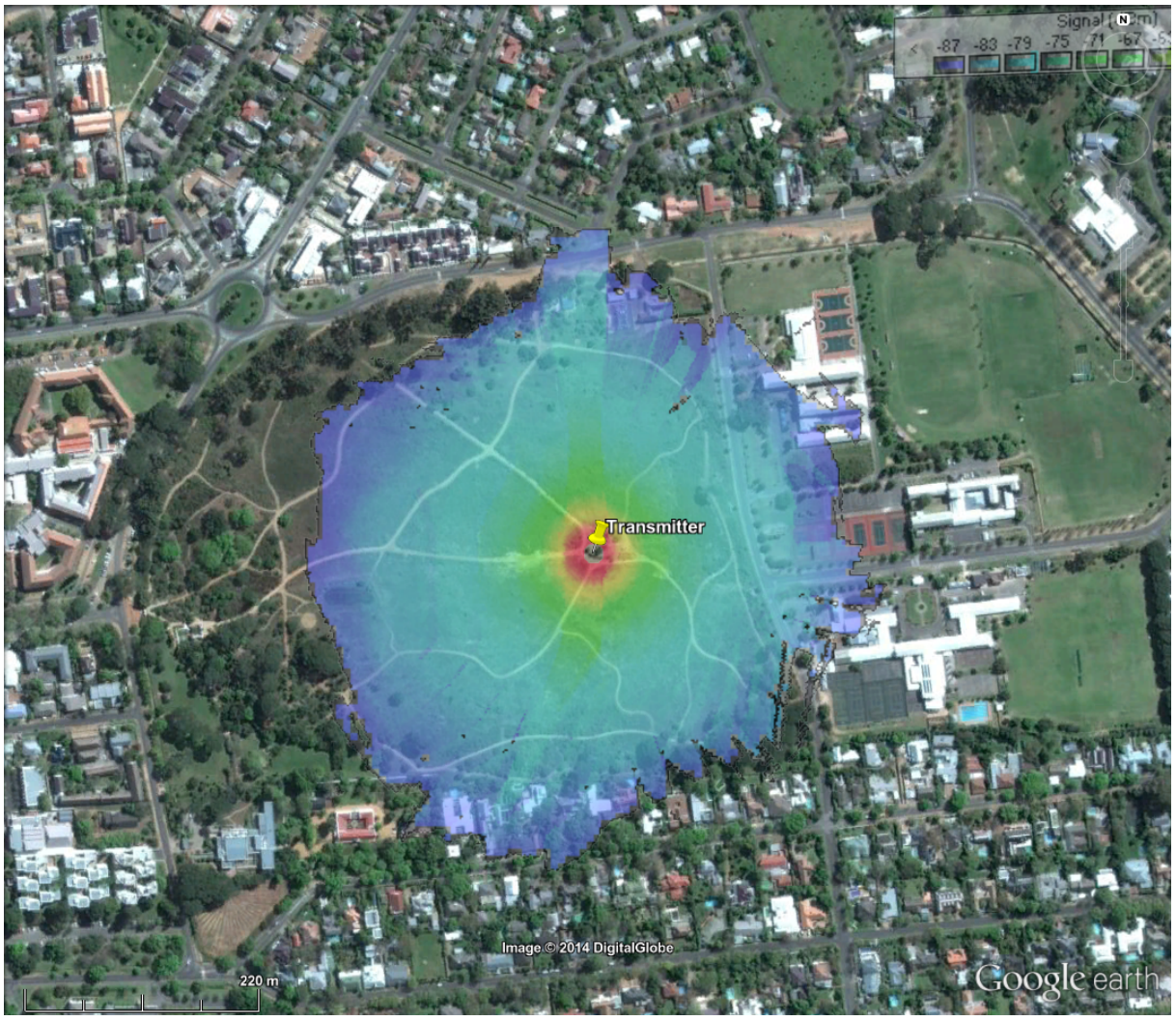


(a) Radio Mobile simulation at 0dBm transmit power.

Colour	RSSI (dBm)
	−87
	−83
	−79
	−75
	−67
	−63
	−59
	−55
	−51
	−47

(b) Legend

Figure E.5: Radio Mobile simulated coverage for a 0dBm transmitter in the Jan Marais Park.



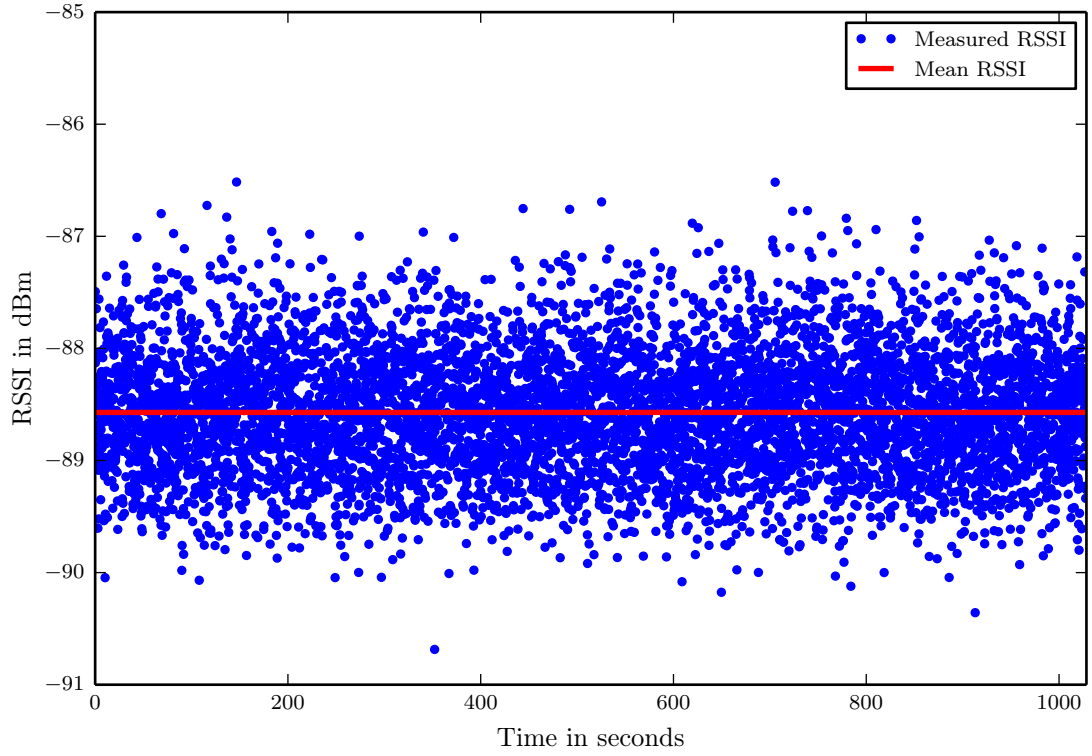
(a) Radio Mobile simulation at 7dBm transmit power.

Colour	RSSI (dBm)
	−87
	−83
	−79
	−75
	−67
	−63
	−59
	−55
	−51
	−47

(b) Legend

Figure E.6: Radio Mobile simulated coverage for a 7dBm transmitter in the Jan Marais Park.

Graph E.1: The noise floor as it was measured over a time of 1000 seconds in the Jan Marais Park.



Pathloss. The three algorithms included in Pathloss were tested one by one and the results are as follows. Tirem seems to be the most optimistic, as in figure E.7. NSMA is a little less optimistic, figure E.8. The most conservative of the three is Pathloss' own model, figure E.9. All the Pathloss simulations are however too optimistic.

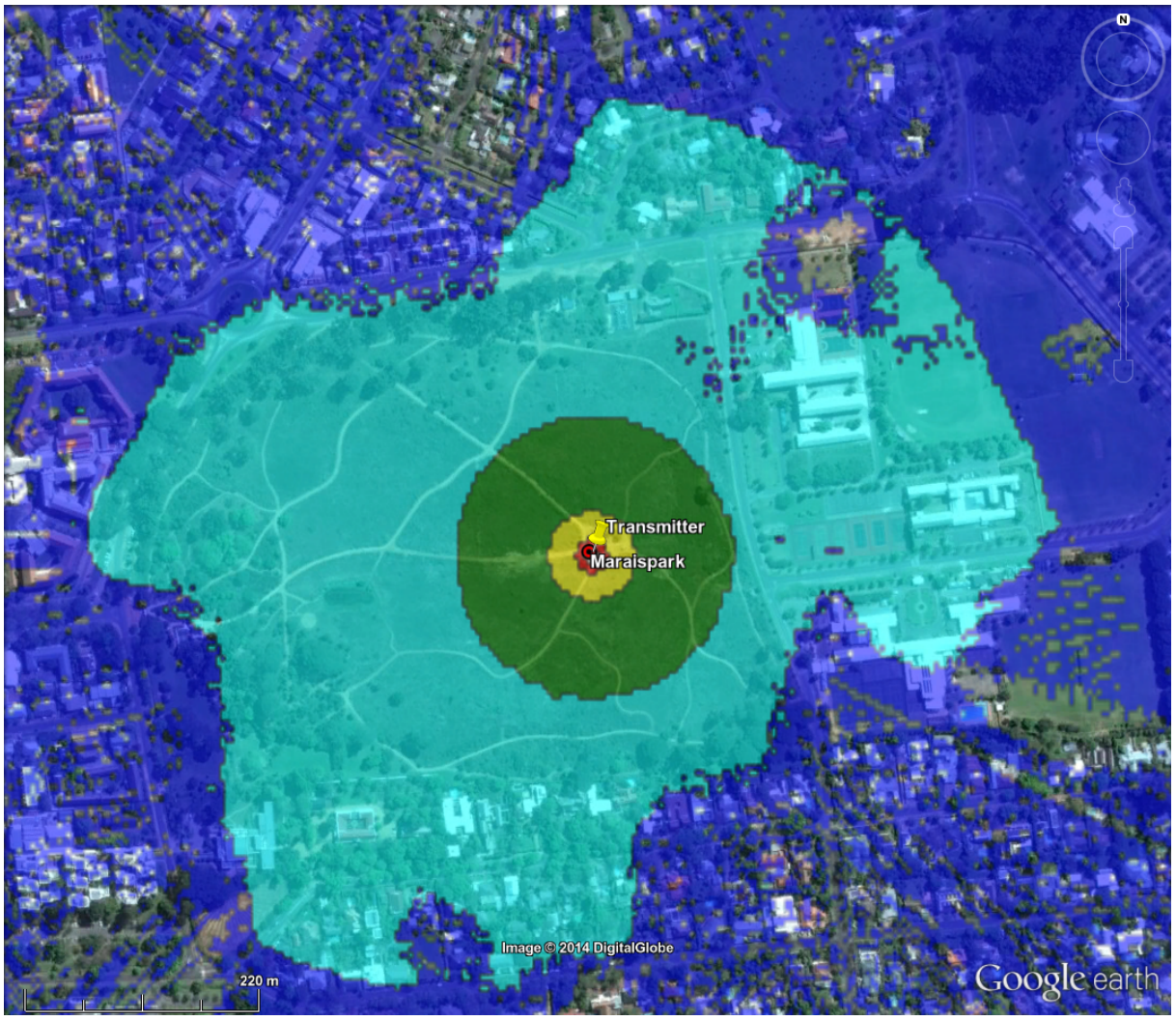
From these simulations the observation was made that Radio Mobile produces a good simulation match to the real world situation. The Longley-Rice propagation model used by Radio Mobile should use a parameter for forest obstruction of 100%. The transmitter output power should also be set to 20dB lower in the simulation than what is to be actually used.

(a) Pathloss simulation at 7dBm transmit power using the Tirem algorithm.

Colour	RSSI (dBm)
	$-85 \leftrightarrow -110$
	$-75 \leftrightarrow -85$
	$-65 \leftrightarrow -75$
	$-55 \leftrightarrow -65$
	$-40 \leftrightarrow -55$
	> -40

(b) Legend

Figure E.7: Pathloss simulated coverage for a 7dBm transmitter in the Jan Marais Park using the Tirem algorithm.

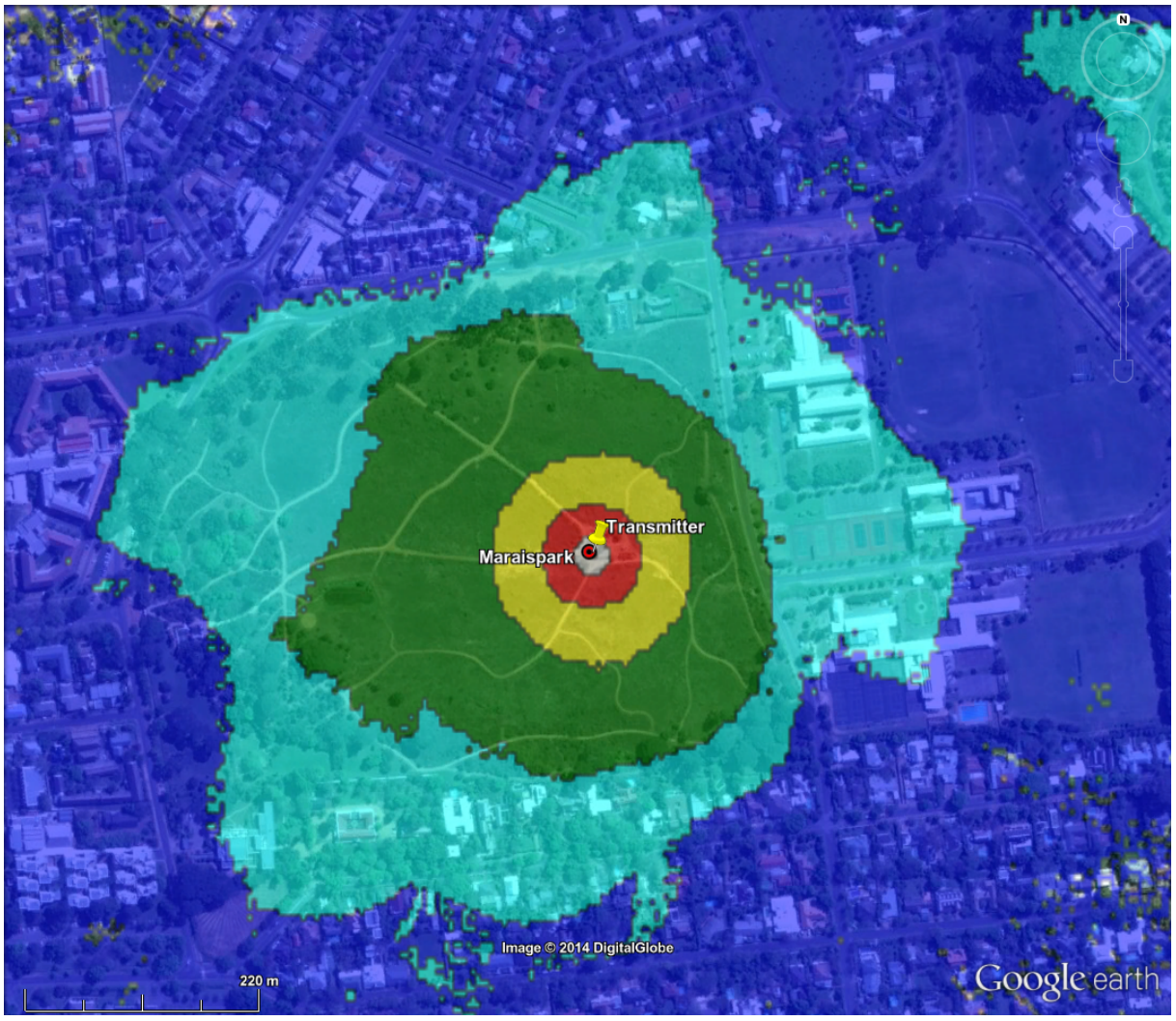


(a) Pathloss simulation at 7dBm transmit power using the NSMA algorithm.

Colour	RSSI (dBm)
	$-85 \leftrightarrow -110$
	$-75 \leftrightarrow -85$
	$-65 \leftrightarrow -75$
	$-55 \leftrightarrow -65$
	$-40 \leftrightarrow -55$
	> -40

(b) Legend

Figure E.8: Pathloss simulated coverage for a 7dBm transmitter in the Jan Marais Park using the NSMA algorithm.



(a) Pathloss simulation at 7dBm transmit power using the Pathloss algorithm.

Colour	RSSI (dBm)
	$-85 \leftrightarrow -110$
	$-75 \leftrightarrow -85$
	$-65 \leftrightarrow -75$
	$-55 \leftrightarrow -65$
	$-40 \leftrightarrow -55$
	> -40

(b) Legend

Figure E.9: Pathloss simulated coverage for a 7dBm transmitter in the Jan Marais Park using the Pathloss algorithm.

Bibliography

[1] D. J. Warnich, "Tracking collar and infrastructure for leopard research", Master's thesis, Stellenbosch University, 2012-08.

[2] A. Markham, "On a wildlife tracking and telemetry system: A wireless network approach", PhD thesis, University of Cape Town, 2008-08.

[3] Africa Wildlife Tracking cc. (unknown). Radio collars, [Online]. Available: `http://www.awt.co.za/awt_radio_collar/radio.htm` (visited on 2014-11-12).

[4] C. Luyt, personal communication, 2014-02-25.

[5] Calvert Amateur Radio Association and N3AE. (2009-04). Fox hunting, [Online]. Available: `http://www.k3cal.org/files/foxhunt.pdf` (visited on 2014-11-16).

[6] *Symposium on Animal Movement and the Environment - Combining animal tracks with remote sensing data about our planet*, 2014. [Online]. Available: `http://amovee2014.com/drones-for-animal-tracking/`.

[7] D. D. Gray, *Conservation drones protect wildlife, spot poachers and track forestloss*, The World Post (The Huffington Post), Ed., 2012-08-19. [Online]. Available: `http://www.huffingtonpost.com/2012/08/19/conservation-drones_n_1806592.html` (visited on 2014-11-12).

[8] C. Andrews, *Wildlife monitoring: should UAV drones be banned?*, Engineering and Technology Magazine, Ed., 2014-07-14. [Online]. Available: `http://eandt.theiet.org/magazine/2014/07/uavs-in-the-wild.cfm` (visited on 2014-11-12).

[9] The Open-Mesh project. (2014). B.A.T.M.A.N., [Online]. Available: `http://www.open-mesh.org/projects/open-mesh/wiki` (visited on 2014-11-12).

[10] Bluetooth SIG, Inc. (2014). The low energy technology behind bluetooth smart, [Online]. Available: `http://www.bluetooth.com/Pages/low-energy-tech-info.aspx` (visited on 2014-11-12).

[11] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet", *SIGOPS Oper. Syst. Rev.*, vol. 36, no. 5, pp. 96–107, 2002-10, ISSN: 0163-5980. DOI: `10.1145/635508.605408`. [Online]. Available: `http://doi.acm.org/10.1145/635508.605408`.

[12] S. Nikoletseas and J. Rolim, *Theoretical aspects of distributed computing in sensor networks*, ser. Monographs in theoretical computer science. Springer Berlin Heidelberg, 2011, ISBN: 9783642148491. [Online]. Available: `http://books.google.co.za/books?id=QGBGt6J5K4UC`.

[13] unknown. (unknown). Contiki: The open source OS for the internet of things, [Online]. Available: `http://www.contiki-os.org/` (visited on 2014-11-12).

[14] T Winter and P. Thubert. (2011-03-13). Internet-draft - RPL: IPv6 routing protocol for low power and lossy networks, [Online]. Available: `http://tools.ietf.org/html/draft-ietf-roll-rpl-19` (visited on 2014-11-12).

[15] B. Saadallah, A. Lahmadi, and O. Festor, "CCNx for Contiki: implementation details", INRIA, Tech. Rep. RT-0432, 2012-11. DOI: `hal-00755482`. [Online]. Available: `https://hal.inria.fr/hal-00755482`.

[16] The APRS Working Group. (2000-08-29). Automatic position reporting system - APRS protocol reference - protocol version 1.0. Ian Wade, G3NRW, Ed., [Online]. Available: `http://www.aprs.org/doc/APRS101.PDF` (visited on 2014-11-12).

[17] W. Beech, D. Nielsen, J Taylor, L Knoper, and G. Jones. (1998). AX.25 link access protocol for amateur packet radio v2.2. G Jones, Ed., [Online]. Available: `http://www.tapr.org/pdf/AX25.2.2.pdf` (visited on 2014-11-12).

[18] J. P. Meijers, M. Amadeo, C. Campolo, A. Molinaro, S. Paratore, G. Ruggeri, and M. Booysen, "A two-tier content-centric architecture for wireless sensor networks", in *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, 2013-10, pp. 1–2. DOI: `10.1109/ICNP.2013.6733621`.

[19] A. M. Wilson, J. Lowe, K Roskilly, P. E. Hudson, K. Golabek, and J. McNutt, "Locomotion dynamics of hunting in wild cheetahs", *Nature*, vol. 498, no. 7453, pp. 185–189, 2013.

[20] Biotrack Ltd. Specialists in Animal Radio Monitoring, *Mortality sensing*, 52 Furzebrook Road Wareham Dorset BH20 5AX United Kingdom. [Online]. Available: `www.biotrack.co.uk`.

[21] Freescale Semiconducktor, *MMA7455L - three axis low-g digital output accelerometer*, 10th ed., 2009-12.

[22] J.P. Meijers, "Low cost web enabled environmental telemetry system", Stellenbosch University, Skripsie, 2011-10. [Online]. Available: `http://jpmeijers.com/skripsie/Skripsie%20Verslag%2015312704.pdf`.

[23] Rod Dinkins (AC6V) [SK] and Jeff Dinkins (AC6V). (2012). Amateur radio prefixes, [Online]. Available: `http://www.ac6v.com/prefixes.htm` (visited on 2014-11-03).

[24] Roger Coudé (VE2DBE). (2014). Radio mobile web site, [Online]. Available: `http://www.cplus.org/rmw/english1.html` (visited on 2014-11-03).

[25] Ian D. Brown (G3TVU). (2014). Radio mobile installer website, [Online]. Available: `http://www.g3tvu.co.uk/Radio_Mobile.htm` (visited on 2014-11-03).

[26] EMSS, EM Software & Systems–S.A. (2014). Overview of FEKO, [Online]. Available: `https://www.feko.info/product-detail/overview-of-feko` (visited on 2014-11-04).

[27] Prislal IA Dobrohotov (UN7GM). (unknown). The normal-mode helix antenna, [Online]. Available: `http://www.cqham.ru/spiral.htm` (visited on 2014-11-03).

[28] Nordic Semiconductor, *nRF905 single chip 433/868/915MHz transceiver product specification*, Otto Nielsens vei 12 7004 Trondheim, 2008-04. [Online]. Available: `http://www.nordicsemi.no`.

[29] AeroComm, *AC4868 the fastest way to wireless*, Laird Technologies, 3481 Rider Trail South Earth City, MO 63045. [Online]. Available: `http://datasheet.octopart.com/AC4868-250M-AeroComm-datasheet-35300.pdf`.

[30] Radiometrix, *BiM3A 869/914MHz high speed FM radio transceiver module*, 2nd ed., Hartcran House, 231 Kenton Lane, Harrow, HA3 8RP, England, 2005-10. [Online]. Available: `http://www.radiometrix.com/files/additional/bim3a.pdf`.

[31] Axsem AG, *AX5042 narrow-band single chip RF transceiver IC*, Oskar-Bider-Str. 1, CH-8600 Dübendorf, Switzerland.

[32] Radiometrix, *Shx1 500mW multichannel VHF transceiver*, 1st ed., Hartcran House, 231 Kenton Lane, Harrow, Middlesex, HA3 8RP, England, 2012-08-17.

[33] K. W. Finnegan and B. Benson, "Clarifying the amateur Bell 202 modem", California Polytechnic State University, San Luis Obispo, Tech. Rep., 2014-07. [Online]. Available: `http://www.tapr.org/pdf/DCC2014-Amateur-Bell-202-Modem-W6KWF-and-Bridget-Benson.pdf`.

[34] G. Dion. (2005-07-18). The WhereAVR, [Online]. Available: `http://garydion.com/projects/whereavr/` (visited on 2014-11-19).

[35] BeRTOS. (). Arduino APRS [AVR], [Online]. Available: `http://www.bertos.org/use/examples-projects/arduino-aprs/` (visited on 2014-11-19).

[36] CML Microcircuits, *MX614 Bell 202 compatible modem*, Oval Park, Langford, Maldon, Essex, CM9 6WG, England, 2000. [Online]. Available: `http://www.cmlmicro.com/Download.aspx?file=MX614_ds.pdf`.

[37] South African Radio League. (2011-10-01). 2m band plan, [Online]. Available: `http://www.sarl.org.za/Documents/Bandplan_SA_2m_20111001.pdf` (visited on 2014-11-19).

[38] Duracell, *Alkaline-manganese dioxide - performance characteristics*, unknown. [Online]. Available: `http://docs-europe.electrocomponents.com/webdocs/0215/0900766b802155d9.pdf`.

[39] WB2OSZ. (2014-05). Raspberry Pi APRS, [Online]. Available: `http://home.comcast.net/~wb2osz/Version%201.0/Raspberry-Pi-APRS.pdf` (visited on 2014-11-21).

[40] ——, (2014). Dire wolf user guide, [Online]. Available: `http://home.comcast.net/~wb2osz/Version%201.0/User-Guide.pdf` (visited on 2014-11-05).

[41] DARPA Internet Program. (1981). RFC 793 - transmission control protocol, [Online]. Available: `http://www.faqs.org/rfcs/rfc793.html` (visited on 2014-11-07).

[42] Snakefoot. (2003). The TCPIP nagle algorithm can slow down network, [Online]. Available: `http://smallvoid.com/article/winnt-nagle-algorithm.html` (visited on 2014-11-07).

[43] M Allman, V Paxson, and W Stevens. (1999). RFC 2581 - TCP congestion control, [Online]. Available: `http://www.faqs.org/rfcs/rfc2581.html` (visited on 2014-11-07).

[44] J. Nagle. (1984). RFC 896 - congestion control in IP/TCP internetworks, [Online]. Available: `http://www.faqs.org/rfcs/rfc896.html` (visited on 2014-11-07).

[45] R. Stevens. (unknown). How can i force a socket to send the data in its buffer?, [Online]. Available: `http://www.unixguide.net/network/socketfaq/2.11.shtml` (visited on 2014-11-07).

[46] Plan-My-Power, (PTY) ltd. (unknown). Solar power calculator, [Online]. Available: `http://www.solarpanel.co.za/solar-power-calculator.htm` (visited on 2014-11-10).

[47] Jim McGuire, KB3MPL, Ivan Galysh, KD4HBO, Kevin Doherty, Hank Heidt, N4AFL, and Dave Neimi, "FX.25 forward error correction extension to AX.25 link protocol for amateur packet radio", Stensat Group LLC, Tech. Rep. v 0.01.06 DRAFT, 2006-09. [Online]. Available: `http://www.stensat.org/docs/FX-25_01_06.pdf` (visited on 2014-11-26).

[48] J. M. G3RUH, *9600 baud packet radio modem design*, Papers of ARRL 7th Computer Networking Conference (US) Oct 1988. pps 135-140 Proceedings of the 1st RSGB Data Symposium, Harrow, England July 1988. (12 pps) Packet: Speed, More Speed and Applications, ARRL 1995. ISBN 0-87259-495-5, 1988-07. [Online]. Available: `http://www.amsat.org/amsat/articles/g3ruh/109.html` (visited on 2014-11-26).

[49] Atmel, *8-bit avr microcontroller with 4/8/16/32k bytes in-system programmable flash.*