Artificial Reconfigurable wireless network optimization based on battery levels

presented by

PETRUS VAN STADEN

Dissertation submitted in fulfilment of the requirements for the degree

Masters in Engineering: Electrical Engineering

in the

Department of Electrical, Electronic and Computer Engineering

of the

Faculty of Engineering, Built Environment and Information Technology

at the

Central University of Technology, Free State

Study Leader: Dr. BJ Kotze DTech (Eng.)

Bloemfontein
October 2019

Declaration

I, PETRUS VAN STADEN, identity number , and student number , do hereby declare that this research project which has been submitted to the Central University of Technology Free State, for the degree Masters in Engineering: Electrical, is my own independent work and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology, Free State, and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.

..... …16 October 2019…….

SIGNATURE OF STUDENT DATE

Acknowledgements

I would like to thank the following without whom this dissertation will not be a possibility:

Dr. BJ Kotze	Mr. CC Jackson	Mrs. CW Smith
Mr. AJ Smith	Mr. CF Swartz	Mrs. M Jackson
Mr. D Kouprianoff	Dr. H van Staden	NRF

In loving memory of Cornelius Christiaan Jackson…

Abstract

A Wireless Sensor Networks (WSN) was developed, along with a testbed that could evaluate the system under scrutiny on energy consumption parameters. The development of an artificial reconfigurable wireless network optimization routing protocol based on battery levels was created in a lab environment where certain variables were controllable. This study focused on measuring energy of a network based on different routing protocols. To capture data, physical batteries were replaced with a power source that was developed to capture the energy usage of each node in the system simultaneously.

The development of a system that could test the energy consumed by a WSN was one part of the study, whilst the important part was to be able to analyse energy consumed by a network as a whole.

Analysing a WSN by capturing data based on routing topology formed a conclusion based on data that was captured in the lab. Different packet sizes, TX power levels and routing protocols were tested in this dissertation on a custom developed testbed. Results did show a predictable outcome clearly indicating different energy consumptions per topology.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Preface

IoT is a group noun for devices connected to an internet-based server via any possible means. The term IoT has brought a new age of exploration into wireless radio networks, security protocols and a few more topics that can be investigated. Very often sensing devices have the issue of being out of range to an internet connection. For these cases a radio network can be implemented, relaying data from one location, to another location that can provide a link to the internet. Modern radio technology is becoming easier to implement, due to the increase of open source material [1] and it is more affordable for the average hobbyist/engineer to work with. When implementing IoT on a network of sensors that are a moderate distance apart from one another (100m – 500m) it is simpler to have a centre node that can be linked to the internet, due to Wi-Fi ranges that might not be in reach for all the nodes. Using GPRS on each device will require a SIM card for each device that leads to extra difficulty on installations. Based on topology algorithms this dissertation researched aspects of a network, before it can connect to the internet on a hardware level. Aspects like throughput battery energy consumption per node and battery energy consumption per network topology was simulated with custom developed physical hardware that can simulate a radio network. These parameters were

tested in lab conditions with open source development tools. The data was logged, pre-processed and analysed.

1.2 The Problem Statement

Certain nodes in a network might use more energy because of the radio topology being used. This is a problem if the radio network is dependable on batteries and does not have a routing algorithm that will adapt to nodes that are depleted.

1.3 Hypothesis

A wireless network sensor system will be able to last longer, allowing longer network up-time when an artificial reconfigurable wireless network optimization system, based on power consumption is used.

1.4 Project objectives

The objective of the study is to analyse a network topology that will be able to adapt based on battery levels of the wireless nodes.

- Obtain and develop a small scale Wireless Sensor Network (WSN);
- Develop a custom protocol to test different topologies;
- Design and build an energy monitoring system for a WSN;
- Analyse the functioning of the test bed;
- Analyse the power consumption of each node in the WSN; and

- Log the throughput of the WSN to compare with different types of topologies.

1.5 Research method

The research was divided into two methods with different outcomes also described in Fig. 1.1. The first objective was to build a system that could monitor the energy consumption of a WSN. Also known as a testbed [2]. A positive outcome is reached when the designed testbed can give the correct energy consumption readings and have a throughput calculation.

The second part of the research analyses different types of network topologies, allowing analysis based on readings of the previously designed testbed. The study will indicate a correlation between the physical layer based on the Open Systems Interconnection (OSI) model and physical hardware.

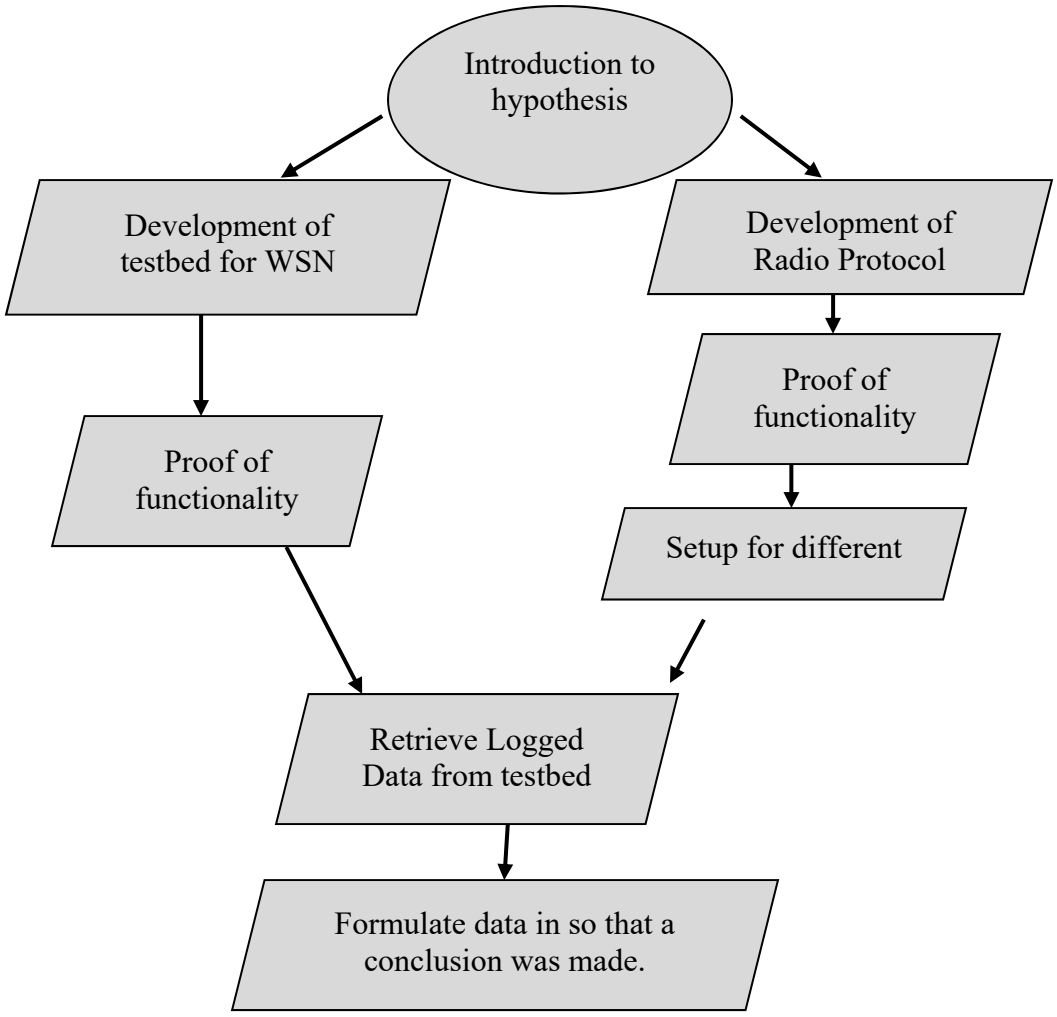

Fig. 1.1 A flow chart of the research method being used

1.6 Summary of Following Chapters

1.6.1 Chapter 2: Investigation into relevant theory for testing WSN's

This chapter considers a broad spectrum of theories related to radio engineering in the process of better understanding WSN's. After the basic radio terminology was discussed, other work concerning IoT research was referenced and discussed. Finally, components for experimental setup were reviewed and chosen.

1.6.2 Chapter 3: Development of a DAQ system for monitoring a WSN

This chapter is an extension of Appendix E [3], and focuses on the development of a testbed that does Data Acquisition (DAQ) on a WSN. The testbed used a Raspberry Pi that interfaced analogue to digital conversion circuits that measured the power of each node connected to the power circuit.

1.6.3 Chapter 4: WSN setup and throughput testing via MATLAB®

This chapter observed the development of a radio protocol that made it easier to test different types of network topologies. The gateway transmitted serial data which MATLAB® was able to analyse.

1.6.4 Chapter 5: Communication and power results

This chapter presents the data collected from the developed system from Chapter 3 (hardware) and 4 (software). Chapter 5 also compared different network topologies and specified if the system was able to predict the energy usage of a WSN.

1.6.5 Chapter 6: Conclusion

This chapter include the discussion of the results; relevance of the research; and future work.

Chapter 2

Investigation into relevant theory for testing WSN's

2.1 Introduction

This chapter discusses relevant theory to WSN's, considering the Open Systems Interconnection (OSI) model and different types of theory that helps in the development of a WSN testbed with a gateway that can be connected to the internet. Also discussed are the tools needed to collect the data; the components to create the data; and the software to save the data.

2.2 OSI model and Relative terms

The OSI model is an International Standard Organisation (ISO) stating that any WSN can be divided into seven layers [4], [5]. The OSI incorporate the physical layer; data link layer; network layer; transport layer; session layer; presentation layer; and application layer as shown in Fig. 2.1 [5]. The IEEE Std 802® also refers to a family of standards focusing on the physical layer and data link layer. It is important to know that IEEE 802 networks observe encryption, flow control and error correction, but it is not limited to the physical and data link layers [6] (layers concerning transmission of data and error detection).

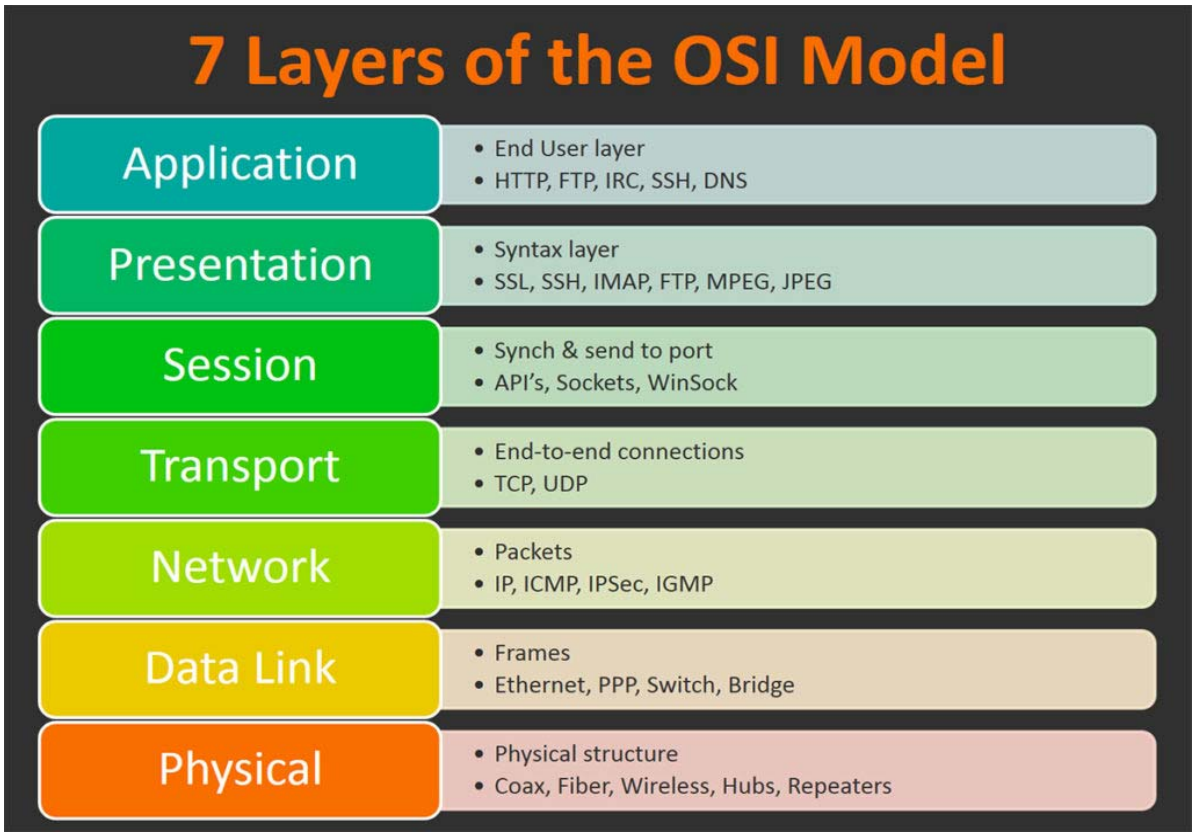

Fig. 2.1 The OSI model for networks

2.2.1 Physical Layer

The Physical Layer (PL) for a radio network system could be described as the process of a radio module that interfaces with the analogue circuit that controls and emits electromagnetic waves via the antenna. The PL for a radio talks to the hardware of an electromagnetic radiation circuit (radio). The PL can be described as a subset of 1's and 0's, but it produces hardware that takes the following into account:

- The frequency to transmit. Frequency can be related to data rate, but licensing should also be considered when looking at frequency

specifications. For this dissertation a radio module that operates at 868MHz is used. This is an unlicensed frequency band that complies with the Independent Communications Authority of South Africa (ICASA).

- Frequency modulations schemes are part of the physical layer. Different types are commonly known, but the popular types taught in general radio engineering subjects are Frequency Shift Keying (FSK); Amplitude Shift Keying (ASK); and Phase Shift Keying (PSK).
- The antenna designs. According to the Modern Dictionary of Electronics an antenna is an electrical device which converts electric power into radio waves and vice versa. Antennas come in different shapes, sizes and types. The antenna is designed for a specific frequency.

The PL used to describe the transmission part of any OSI model [7], the ArduRF1's (Fig. 2.2) was used to build the experiment platform for measuring energy usage during transmissions of data. It is developed around the Arduino framework and uses a RFM69HW radio transceiver, designed by HopeRF®, operating at 868MHz.

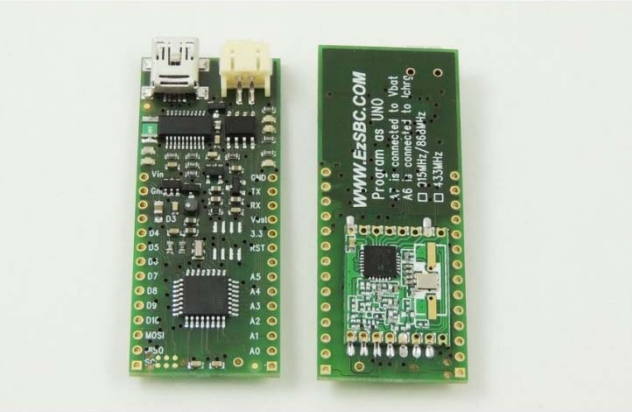

Fig. 2.2 Wireless nodes ArduRF1's Photo Credit: www.EzSBC.com

2.2.2 Data Link Layer

The Data Link Layer (DLL) [7] can be classified as the part of a radio module that describes the error correction and analysing of the data. Two sublayers, encryption and decryption can also happen in this layer, and is known as the Logical Link Control (LLC) sublayer and the Medium Access Control (MAC) sublayer.

2.2.3 Network Layer

A Network Layer (NL) [7] is responsible for packet forwarding. For example, node X sends a packet to node Z, but needs node Y to pass the message. Common platforms on this layer are called gateways or routers. They are connectors of communication [7]. In this dissertation a custom NL is designed to serve as an easy method of controlling the topology of the network.

2.2.4 Transport Layer

The Transport Layer (TL) [7] is commonly better known in high data rate networks, for example computer networks. This terminology describes the protocol being used for different node destinations to understand one another e.g. UDP and TCP/IP.

2.2.5 Session Layer

The Session Layer (SL) [7] describes the initiation of a communication link and the ending of the communication link. A good example will be a wireless webcam. If a link is requested and accepted, data will be able to flow from the webcam via the wireless link to the source that requested the link. When the link is disrupted or cancelled the session will be stopped and no further data streaming takes place.

2.2.6 Presentation Layer

The Presentation Layer (PL) [7] or syntax layer is the layer that would grab data and/or manipulate data to make it usable to the application layer, e.g. ASCII. Some types of encryption take place on this level. Testing of the username and password, to grant access to user data would be an example.

2.2.7 Application Layer

This layer (AL) is also known as an abstraction layer and can be described as a user interface [8]. A user interface can range from a LCD to a LED blinking. For this dissertation the Application Layer will be absent, but interfacing LED's are designed into the circuit to allow for better debugging purposes.

2.2.8 Throughput

Throughput means the amount of material or items passing through a system or process. When working with radio communication, throughput is an important factor in the evaluation of a system. In perfect conditions the throughput of a radio system must be 100%. For this thesis throughput was measured for the developed testbed system. The effects of directional antennas versus omni-directional antennas are discussed by Hong-Ning Dai et al. [9]. It was found that a directional antenna can improve the throughput significantly and reduce the need for multi-hop routing.

2.2.9 Latency

Latency is a common term used to describe the delay in data. High latencies refer to major delays in data. When sensing data for logging purposes, latency can be negligible comparing to wireless control systems. Latency will only be an issue for high data rate application e.g. video feeds or sound streaming. Om Jee Pandey et al. [9] researched low-latency and energy-balanced data transmission over cognitive small world WSN. A network was proposed that viewed hop-counts and geographical distances that acquired 44.74% reduction in latency, compared to Low Energy Adaptive Cluster Head routing (LEACH) [10].

2.3 Modulation of a carrier signal

Modulation refers to the way a signal needs to be carried through open space. When using wires between different devices for communication purposes, protocols like

UART and I^2C can be used, but when communicating wireless only through antennas the signals must be modulated with external electrical waves to enable the transmission of radio magnetic energy via an antenna. Modulation schemes were developed for different types of criteria. Popular modulation schemes are amplitude modulation and frequency modulation. These modulation schemes provide a service to analogue signals. Very common techniques taught at universities when working with digital data (binary values) are phase-shift keying (PSK); frequency-shift keying (FSK); and quadrature amplitude modulation (QAM). All this modulation can be sub-divided into different categories allowing for better application-based modulations.

Summarising modulation schemes can be divided into analogue; digital; hierarchical; and spread spectrum modulation. Further modulations described refer mostly to spread spectrum digital modulation schemes used on common breakout boards sold by electronic and robotic component suppliers [11].

2.4 SigFox™

SigFox™ is popular in the world of IoT. This is a radio network based on star topology, with a gateway node that handles the data and uploads it to the cloud. The significant part of this IoT design is the fact that there is no need to create a link to the internet. The link to the internet is provided by SigFox™. Fig. 2.3 [12] is a coverage map of SigFox™. The light blue area is where SigFox™ coverage is already deployed and purple is the area where SigFox™ is planning to provide coverage in the near future.

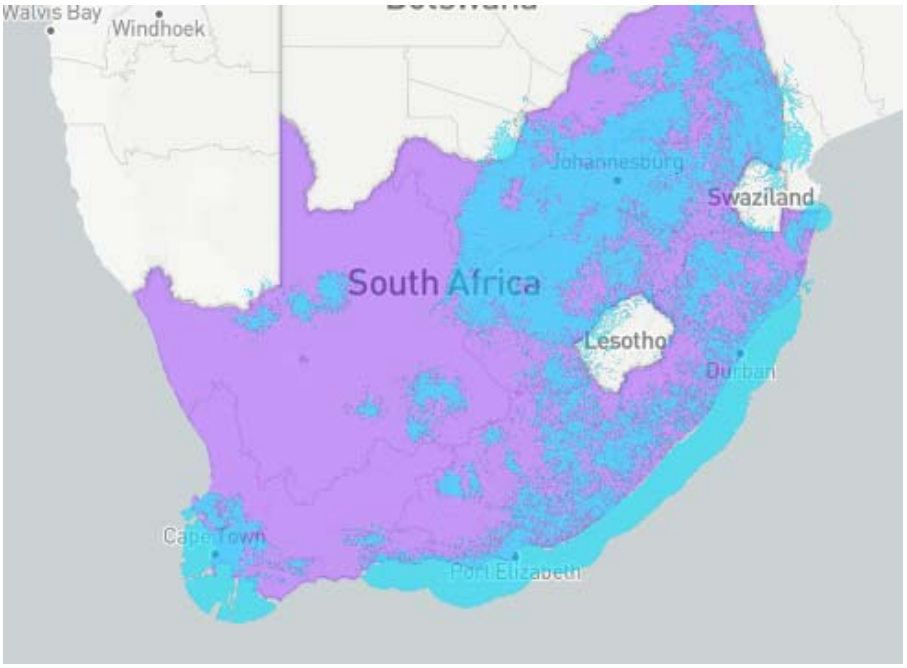

Fig. 2.3 Coverage provided for SigFox™ modems in South Africa on the 19th of March 2018 as provided by SigFox™

This service is significant in the sense that the coverage is very similar to mobile networks, but extra SIM cards are not needed for functionality. Each SigFox™ modem is provided with a unique ID. The user of the SigFox™ modem needs to register this unique ID and to pay a license fee. This license fee can also be considered as data. Using the unique ID on the SigFox™ server, the user can access the data provided by the SigFox™ modem.

The largest data package that can be obtained from SigFox™ is a package that transmits 12 bytes every 10 minutes. This system is dependent on being in range of gateway towers. It does not allow for a lot of data to be uploaded, therefore only uncomplicated sensing applications can make use of this system. Water usage

monitoring; rain fall measurements; or lightweight tracking systems would be an example.

SigFox™ differs from the experiments done in this dissertation because it only uses a star network topology.

2.5 WSN Topology Control and Algorithms

EZone is an energy efficient zone routing algorithm [13]. MATLAB® simulations analyses the effectiveness of this algorithm on network performances for single and multiple gateway scenarios. Comparing EZone to routing algorithms like minimum transmission energy (MTE), LEACH and zone clustering, a conclusion can be made that EZone has a better throughput rate. That also means a better network connectivity with more lifeless nodes.

EZone is only based on theoretical work, leaving the practical testing of this algorithm up to expensive testbeds for these radio networks. EZone algorithm creates a radio link between neighbouring nodes based on energy level.

During reduction of energy consumption in WSN, using the generalized Pythagorean theorem [14] a routing scheme was proposed on the physical location of the nodes used in the network. Using theoretical info of the power consumption used by a node simulation done in MATLAB®, an algorithm was tested based on the angles and distances between each node. Fig. 2.4 [14] illustrates that the angle between different nodes had a theoretical effect on the energy consumed if the distance between node A

and B and node A and C was constant, but the angle was different. The proposed algorithm chose an extra hopping step between nodes if the angle from node A's point of view was smaller than 90 degrees.

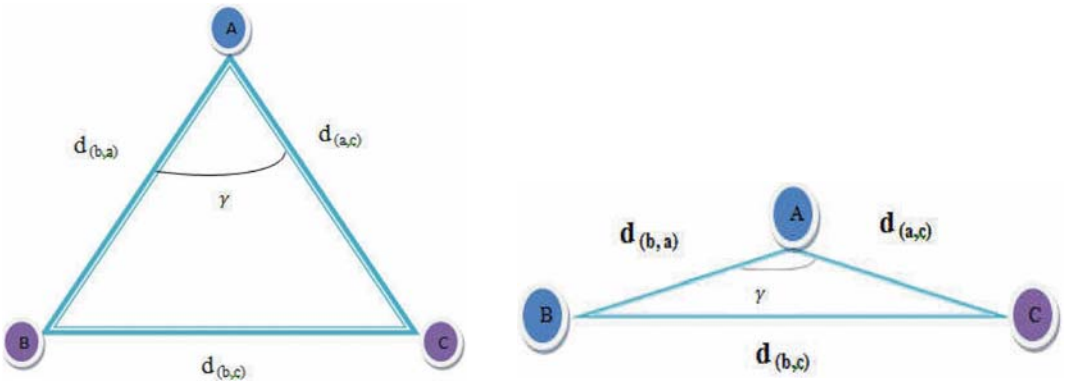

Fig. 2.4 A, B and C represented as nodes showing diffirent angle possibilities

Energy efficient reliable routing algorithm (EERA) for WSN[15] focused on throughput, reliability and latency. The EERA algorithm was developed from the basis of low earth orbiting (LEO) [16] satellite networks. LEO makes use of an algorithm where only the neighbouring node data was stored in a specific node.

For example, on analysis of node a, it will hold only data concerning its neighbouring nodes b and c, but none on the non-neighbouring nodes d and e. EERA implemented this strategy, but added data saved within the node, holding information on nodes that is on route to the gateway.

EERA protocol's header packet format contained 10 packages.

1. The source node;

2. Destination node;

3. Node between source and sink;

4. Neighbouring node;

5. Sequence number of the packet;

6. Packet start time of each node;

7. Time to reach base station;

8. Energy of node;

9. Node packet transmission ID;

10. Tuning factors.

2.6 WSN Power Consumption Testbeds

Section 2.6 discusses the relevant theory concerning testbeds. This topic became part of the dissertation because of its relevance to this project. The author of this dissertation designed and built a testbed specifically for testing the hypothesis of this research.

2.6.1 Introduction to Testbeds

Testbeds are designed and used to evaluate a WSN. Outputs usually contain data concerning the energy consumption, data rates and throughput capability. WSN testbeds are mostly deployed indoors for academic purposes [17] [18].

A. Kopke et al. measured the node energy consumption in a USB based WSN testbed [19] and he also researched more affordable, easy deployable and precise methods of estimating the actual energy consumption. The systems were developed to be independent from node type or testbed type. The developed circuit (Fig. 2.5 [19]) allowed the node being tested to switch between programming mode and energy consumption measuring mode. This operation was done by the node itself.

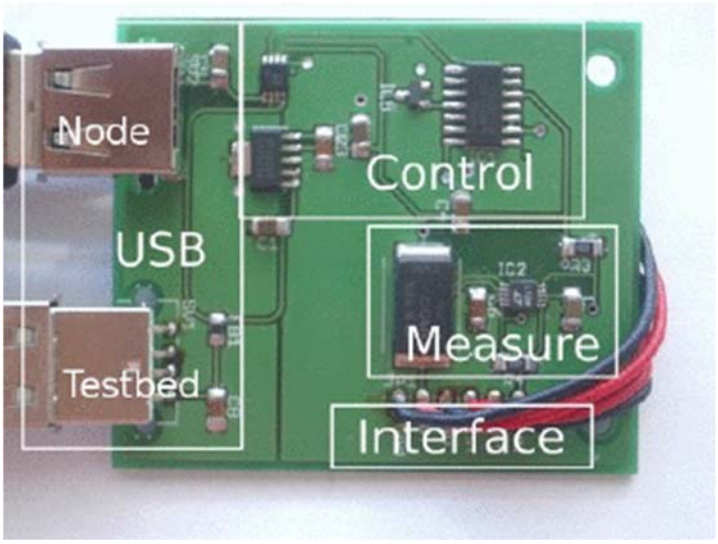

Fig. 2.5 Circuit for a USB based WSN energy monitor developed by Kopke and Walisz

The circuit shown in Fig. 2.5 simply explains that by the use of a data pin output to the node being measured, the resultant current measurements, interfaced with a LTC4150 coulomb counter's data, was transferred to the node itself, allowing the data to be accessed wirelessly. One of the main concerns in this research piece was cutting out the regulator that provided the system with 3V and replacing it directly with a battery. The problem that remains when using rechargeable batteries on a 3V node, is that additional charging circuits are needed.

2.6.2 NEWSBED: The Internet of Things Testbed Platform

NEWSBED [20] is a testbed theoretically designed for IoT. A few requirements were identified that are useful when creating a testbed.

It has to be neutral, easy making, web based and shareable.

Neutral indicates that the testbed is simulated, but with extra Quality of Service (QoS) as a Service. This implies that extra data is injected into the simulation conditions to make the simulation a better representation of reality. **Easy making** is the requirement that only concerns the programing environment of the system. NEWSBED intends to make the coding interface a graphical user interface (GUI). **Web based** refers to access of the collected data. Allowing the data to be presented on a web site. **Shareable** indicates that NEWSBED is synonym to open-source. If a researcher already set up an environment on the system, it is easily accessible to other researchers. Fig. 2.6 [20] illustrates that research on NEWSBED can be done from remote locations.

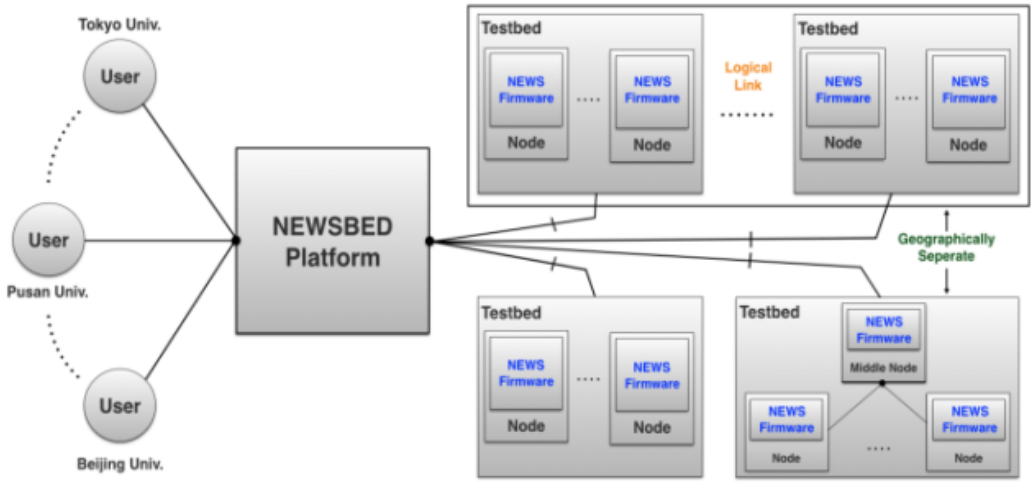

Fig. 2.6 The deployment of a NEWSBED testbed

In conclusion this article demonstrates that NEWSBED is merely an idea with no proof of concept yet. NEWSBED also doesn't consider cost effectiveness or any physical implementation of a testbed.

2.6.3 MoteLab: a wireless network testbed

MoteLab [21] (Fig. 2.7) was developed to help future researchers and teachers whilst researching or teaching WSNs. Quoting the author, "MoteLab is a set of software tools for managing a testbed of Ethernet-connected sensor network nodes." As shown in Fig. 2.7 the system makes use of a webserver, MySQL database, a data logger and hardware connected to the wireless nodes that interfaces with Ethernet. The radio nodes can be reprogrammed via the Ethernet interface board. The Ethernet interface board can also monitor the power consumption of the node that is plugged in and connects the node to extra sensory data.

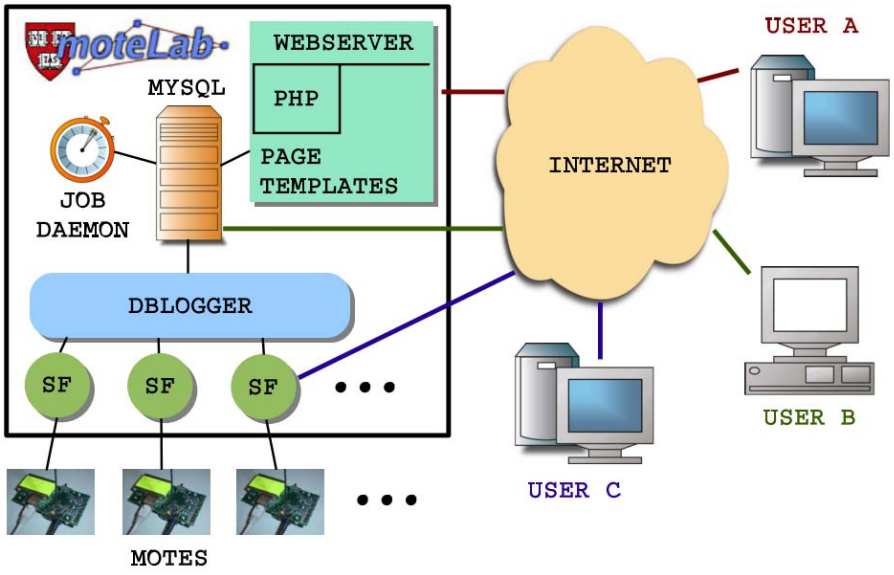

Fig. 2.7 Component model that illustrates a summary of MoteLab

In conclusion MoteLab is a spectacular testbed that implemented simple techniques like MySQL and a PHP webserver to interface the general system. The only negative aspect about using this testbed is that the type of wireless nodes used in this testbed can't be changed or adapted. Each wireless node also needs to be within range of an Ethernet port.

2.6.4 WISEBED: an open large-scale wireless sensor network testbed

WISEBED [22] focuses on a very large WSN testbed system placed all over Europe. WISEBED deployed over 500 nodes with different types of hardware platforms with pre-connected sensors. Most nodes work with the IEEE 802.15.4 wireless protocol as a standard. This is ideal for topology control testing at a radio frequency of 2.4GHz.

WISEBED's intended goal is to connect each testbed to larger heterogeneous networks e.g. IoT.

This testbed is well developed but has limited types of hardware platforms. The hardware is not interchangeable, and the testbed only focusses on research based in the physical layer of the OSI as described in section 2.2.1.

This testbed will not allow for research on the physical hardware of a node. The testbed that was developed in this thesis is very small in comparison, but focuses on the hardware's performance in relation with the physical layer setup. The physical layer also focuses only on the testing of different topologies with preconfigured firmware.

2.7 Appliance Load Monitoring (ALM)

Google defines the word appliance as a device or piece of equipment designed to perform a specific task. Identifying with the term appliance, a wireless sensing node can be defined as an appliance. As explained by a survey [23] there are two common types of Appliance Load Monitoring (ALM). Intrusive Load Monitoring (ILM) and Non-Intrusive Load Monitoring (NILM).

2.7.1 ILM

A good analogy for ILM would be to introduce water consumption sensing per tap in a household. ILM would refer to the sensing of water usage at each tap by literally putting a water flow sensor at each tap. Using ILM and the water analogy example, data can be presented for each tap. This is an uncomplicated method for ALM when data

collection is the main concern, but it is a more expensive and inconvenient method of water usage monitoring than the other method, NILM.

2.7.2 NILM

Using the same water analogy as used for ILM, the NILM introduced the single point sensing method. For this explanation a water analogy will work fine. Instead of placing a water flow sensor at each tap inside the house, one sensor can be placed outside on the main water supply. This is a cheaper method, but the problem comes down to knowing which tap gave what amount of water. When implementing what researchers have discovered [23], it is possible to specify which tap/appliance was opened/latched by evaluating the effect that it has on the total water flow/current measurement.

As proposed by [24] there are different types of consumer appliances. Ranging from power supply latching appliances (ON/OFF); multi-state appliances (appliances that consume different amounts of power during its operation); continuously variable devices (Light dimmers); and [25] permanent consumer devices (geyser and hardwired smoke detectors).

2.7.3 Conclusion for ALM

ALM is useful when it comes to monitoring energy. Comparing ILM and NILM they both have their advantages and disadvantages. ILM is a bit more expensive, but will be helpful when precise data must be logged. NILM is a predictive method, not usually 100% accurate, but it can be helpful for smaller scale measuring systems.

For example: measuring the power consumption of a residential house instead of a 5-story building. The testbed used for the radio network made use of ILM, because of better accuracy and higher resolutions when it comes to energy monitoring.

2.8 Radio Transceiver modules

Radio transceivers are very popular in a module format (with all the components on a small printed circuit board to manage the transmission and receiving of data bytes). A common term for modern radio modules are LoRa®. LoRa® falls under the topic Low Power Wide Area Network (LPWAN). The radio modulation scheme used is the chirp spread spectrum (CSS). Other modulation types can be the basic frequency shift keying; phase shift keying; or amplitude shift keying. Experimentation for this dissertation was done with the RFM69HW that can do FSK, GFSK, MSK, GMSK and OOK modulations.

The following sections 2.8.1 and 2.8.2 discusses popular types of transceiver modules operating at 868MHz and a review on the features that will apply to this dissertation. 868MHz was chosen as it is an open license band in South Africa and does not tend to interfere with other sub 1GHz open licenses, e.g. the 433MHz band.

2.8.1 RFM69HW from HopeRF™

Described as a low powered transceiver module (Fig. 2.8) with a wide frequency range of operation most of the communication parameters are programmable. High receiving sensitivity of -120dBm at 1.2 kbps and a programmable power output can be

programmed with ranges from -18 to +20 dBm in 1dBm steps. See appendix A for extra information.

Fig. 2.8 RFM69HW radio transceiver

2.8.2 RN2483 from Microchip®

Described as a low powered transceiver module with a wide frequency range of operation most of the communication parameters are programmable. It is a System on Chip (SoC) with a high receiving sensitivity of -120dBm at 1.2 kbps and a programmable power output can be programmed with ranges from -18 to +20 dBm in 1dBm steps. Fig. 2.9 [26] shows the module.

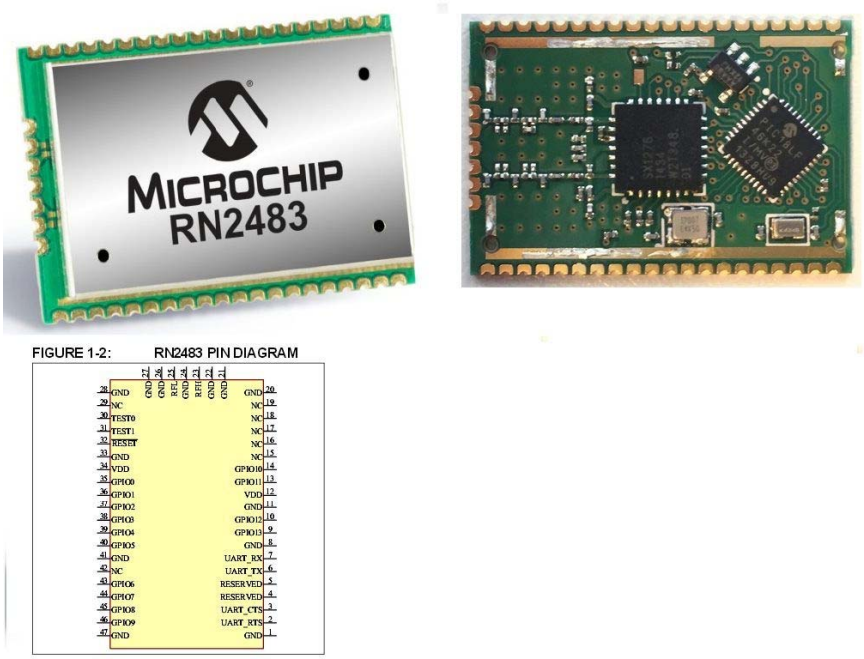

Fig. 2.9 RN2483 SMT chip developed by Microchip

2.9 Gateway between sensing grid and internet

A gateway is a hardware-based device that provides a connection between two or more different types of network protocols. In most cases the gateway is a node with an extra connection interface that can be connected to the internet or any other type of frequency that is different from the first or main network that the gateway is connected to. The gateway may also be referred to as a bridge between a network and a user interface [27].

2.10 Summary

All of these concepts were investigated, applied or altered in the incorporation and development of the research project. Research about testbeds for WSNs, power

optimisation protocols and relevant theory to radio networks was discussed and presented in Chapter 2.

Chapter 3

Development of a DAQ system for monitoring a WSN

3.1 Introduction

Exploratory work was done for chapter 3, to ensure that the obtained current measurements from a Raspberry Pi and an ADS115 16Bit I^2C ADC module are correct. This research is to conclude if a WSN will operate longer when the routing algorithm is based on the battery level of each node in the WSN. The network will be tested with 9 nodes; one being the sink node (gateway node) and the rest are sensor nodes. Measuring the power usage of the WSN will give viable data that won't need a simulation to speculate the performance of a WSN's routing algorithm. Using the Raspberry Pi and an ADS115 16Bit I^2C ADC module will provide an accessible method of obtaining energy measurements and logging the data. Before a testbed for monitoring energy consumption of a WSN can be build, the data acquisition (DAQ) should operate properly on a single node. This simulation can replace actual field work and long delays for collecting data.

3.2 Tools used to test DAQ credibility of the WSN testbed.

Section 3.2 discusses the tools and considerations of these tools used for the design of a testbed.

3.2.1 The Digilent Analog Discovery™

Developed in conjunction with Analog Devices Inc., replaced by the Analog Discovery 2, the Digilent Analog Discovery™ (Fig. 3.1) can be used as an oscilloscope; waveform generator; logic analyser; pattern generator; and a programmable power supply. Using the software WaveForms 2015 the device can log readings taken from the oscilloscope channels. The logged readings can then be used to create graphs for better analysis and evaluation of the transmission power of a specific node.

Fig. 3.1 Digilent Analog Discovery tool

3.2.2 NI myRIO™

As presented in previously mentioned work of the author of this dissertation[3], the myRIO was a useful tool when logging analogue data. It combines a Field Programmable Array (FPGA) chip with a real time processor. This allows a very fast analogue sampling rate.

Fig. 3.2 MyRIO from National Instruments

3.2.3 Python

Python is a high-level programming language for the programming of any type of application. In this dissertation's situation python is used on the Raspberry Pi. Usually Linux and python are compatible without problems. This observation comes from the fact that python comes preinstalled on most Linux distributions as said on the official python support web page [28].

3.2.4 MySQL

MySQL is a database platform. It is convenient for storing data, requesting data and just handling data. For this dissertation it was used to store throughput and energy usage data, recorded by the designed radio testbed.

3.3 The Node for testing DAQ tools

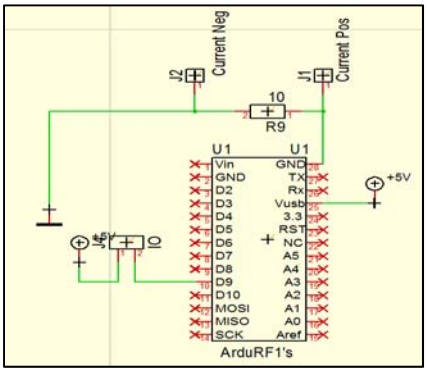

Fig. 3.3 Basic circuit for monitoring energy usage

The node that was tested is a single ArduRF1s placed in series with a 10Ω resistor. Measuring the voltage difference through the resistor will give a resultant value that is 10 times the value of the current usage. Using the Target3001!™ software the circuit board was designed to interface with tools mentioned in section 3.2.1, 3.2.2. Fig. 3.3 shows the schematic for a circuit that is used for energy consumption calculations. J1 on Fig. 3.3 was used to measure the positive voltage, where J2 was considered as 0V.

The Power Monitor circuit (Fig. 3.4) is designed for supplying a known voltage to the ArduRF1's and measuring the breakout pins (J1 and J2 from Fig. 3.3) to calculate the instantaneous current value and is equipped with an on board 5V/1A power supply. The instantaneous power is calculated by using the variable data collected from the power monitoring circuit. Results can be calculated by using equation (3.1 only by using a DAQ device energy consumption.

$$J = IV\Delta t \tag{3.1}$$

$J = Energy\ in\ Joules\ (J)$

$I = Current\ measured\ in\ Amsp(A)$

$V = Volts(V)$

$t = Tme\ measured\ in\ seconds\ (S)$

Fig. 3.4 Power monitor circuit for the ArduRF1's

3.4 Analog Discovery setup and test

Hooking up the power monitor circuit (Fig. 3.4) to the Analog Discovery tool on the prescribed channel 1 leads an external power source needs to be connected to the power monitoring circuit. Using software WaveForms™ 2015 the tool's logging function was used. This function logged analogue values, plotting it on a graph and also allowing the data to be exported to a .CSV file. A .CSV file is used for simplistic reasons. Fig. 3.5

shows the connection between the power monitor and the Analog Discovery device now used as a DAQ.

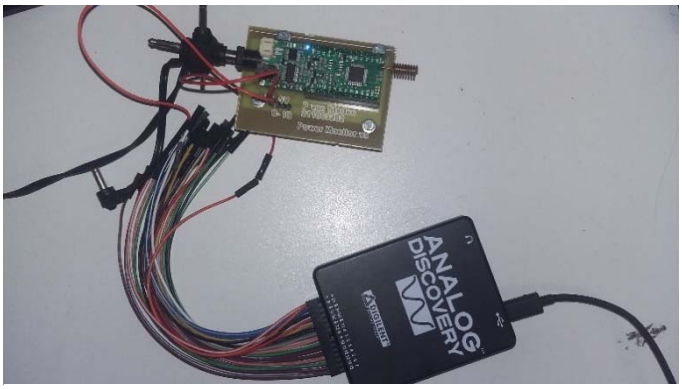

Fig. 3.5 ANALOG Discovery monitoring the current consumption of an ArduRF1's

Three experiments were done with this tool. The first experiment involved manual reprogramming of the ArduRF1's after each test, in order to maintain a constant power level. The power level is a value that is related to the transmission power of the wireless node in the testing of power consumption while changing the package size. The package size was varied between 60 packets and a minimum of 5. After each run the package size was changed manually on the firmware and the ArduRF1's was reprogrammed. Fig. 3.6 shows the programmed flowchart for the first tests.

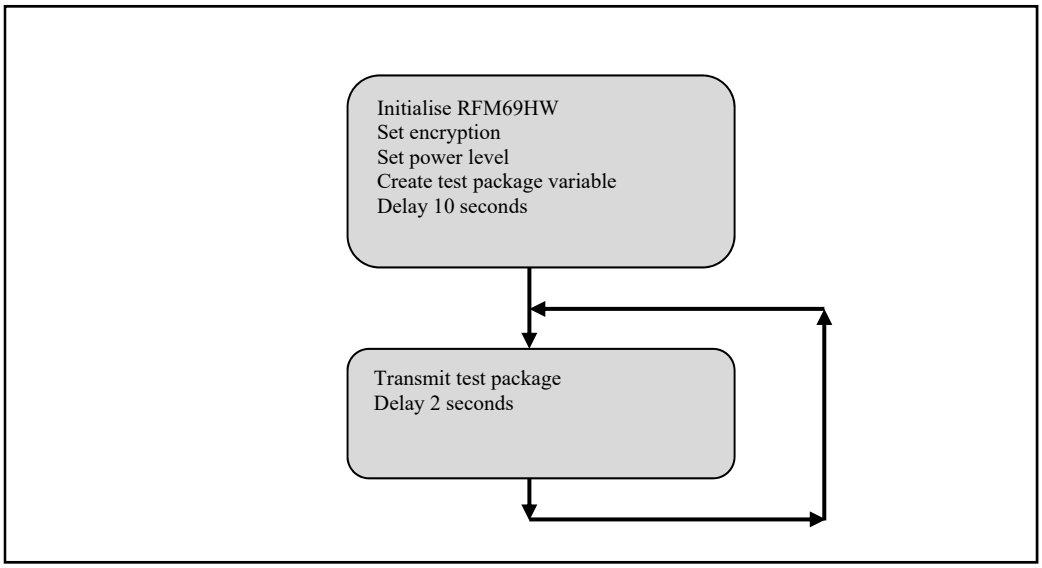

Fig. 3.6 Flowchart for firmware on the ArduRF1's when testing the power consumption related to package size.

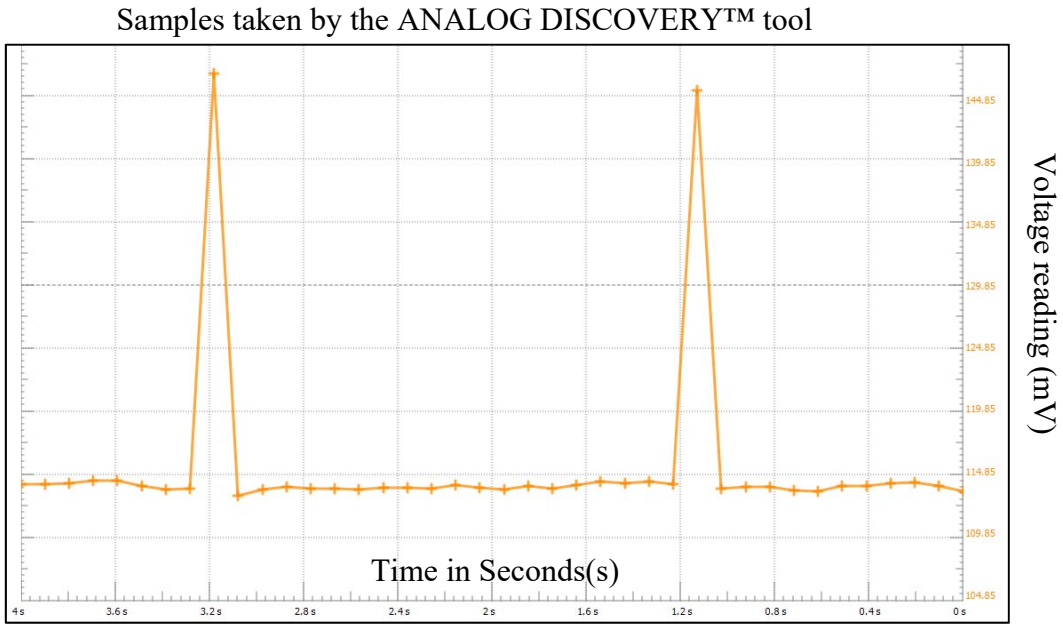

Fig. 3.7 Voltage level recordings when package size was set to 5 bytes

Using the Logger function provided by Waveform's™ (2015) data voltage readings across a 10-ohm resistor was taken, to determine the current at a specific time stamp. Fig. 3.7 displays logged data over a 4 second period allowing two transmissions during the logged session.

After saving all the data to a .CSV file, using Microsoft Office Excel, the total energy for the period of 4 seconds was calculated using the following function as represented by equation (3.2.

$$E = \left(\sum_{n=1}^{N} \frac{x(n)}{R}\right) \times V \times t \quad \textbf{(3.2)}$$

$E = Energy\ in\ Joules$

$N = total\ number\ of\ values\ recorded$

$x(n) = pointer\ to\ the\ data\ value\ collected$

$R = resister\ value\ to\ measure\ the\ current$

$V = the\ supply\ voltage\ to\ the\ node$

$t = time\ in\ seconds$

The N in equation (3.2 represents the values recorded in a table, divided by 10(value of the resistor) to determine the current value and multiplied by time and voltage to determine the energy used during the test. After using this formula on tests done with packet sizes of 60, 30, 15 and 5. Table 3-1 was constructed for a summary of all recorded data in the experiments.

Table 3-1 A summary of data collected from an experiment changing the package size of a transmission

Test number:	Package Size (Bytes)	Energy used for 2 transmissions(mJ):	Peak Current reading(mA):
1	60	238.627	20.96
2	30	235.673	18.936
3	15	234.384	16.883
4	5	230.992	14.656

Conclusion after the evaluation of Table 3-1. Package size has an effect on the power consumption, as is clearly indicated by the results of the energy consumption of a node during a 4 second evaluation period. The difference in the energy consumed by each test done is not linear, meaning a bigger package size's energy consumption can't be predicted using y=mx+c.

For the second experiment using the Analog Discovery an extra loop was inserted into the firmware. This time the experiment was done to see the influence of the power level register of the RFM69HW. Fig. 3.8 shows the flow chart of the firmware that the wireless node was operating on. The experiment was done 3 times with packet sizes of 60, 30 and 10 bytes. The ArduRF1's was reprogrammed after each successful session of logging data. The logging time was also adapted from 4 seconds to 20 seconds. After collecting data from WaveForms the data was exported to a .CSV file. Adding the data in 1 file the following chart was created as shown by Fig. 3.9.

The timing of the transmissions is not in sequence as the logging task was initiated manually. This was sufficient for evaluation of the different current levels,.

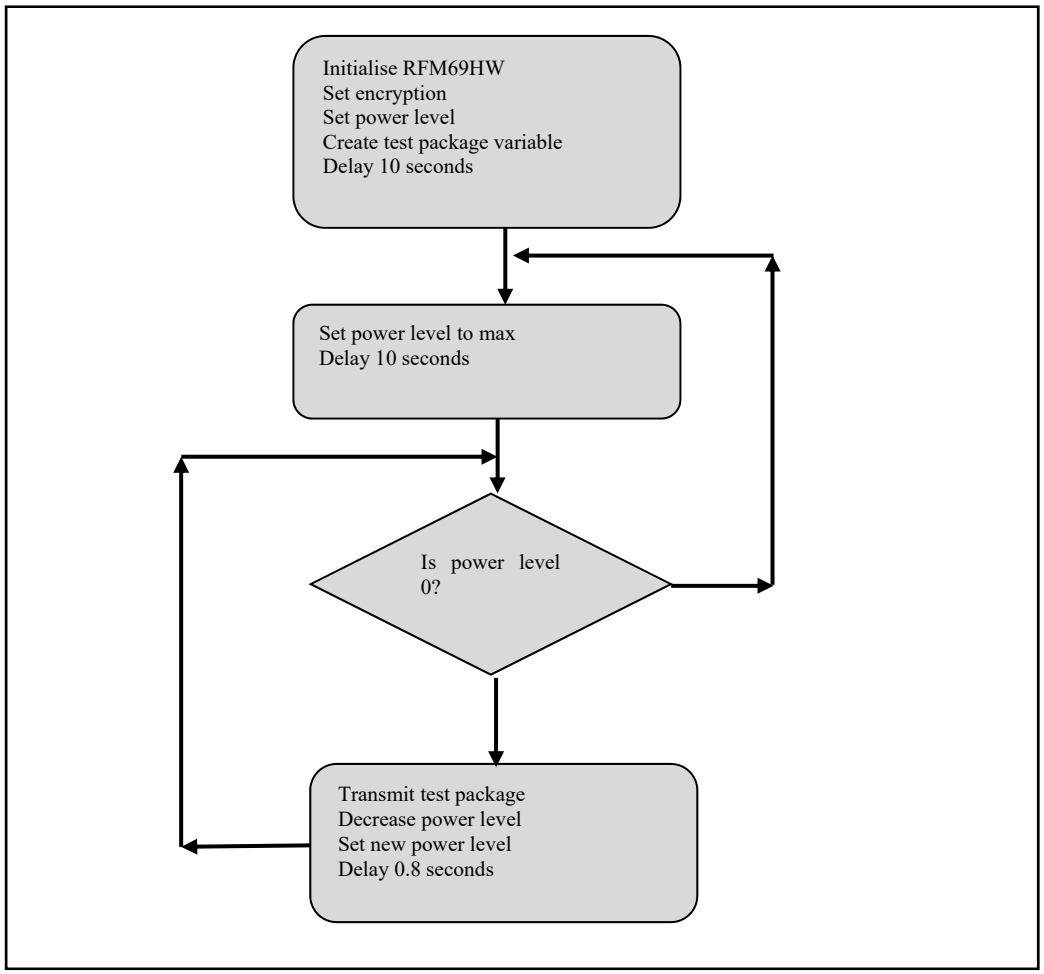

Fig. 3.8 Flowchart diagram for changing the transmission power level in a loop

Evaluation of Fig. 3.9 illustrates that changing the power level has a major impact on the peaks of the readings taken by the Analog Discovery. Comparing the number of bytes transmitted with the power level, it is clear that by using the minimum power level with 30 bytes, the current spike is almost the same as when using maximum power with

10 bytes as the package size, as shown in Fig. 3.10. In fact, when using 10 packets the power level has not that much effect on the wireless node.

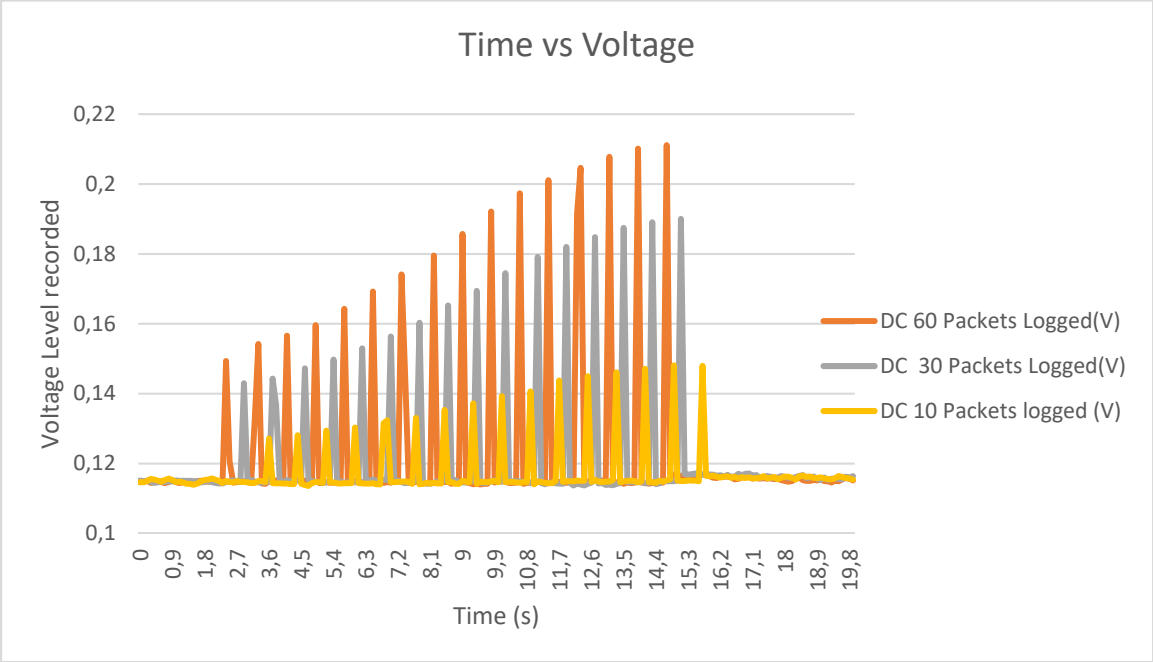

Fig. 3.9 Recorded values for different package sizes while decreasing the power level

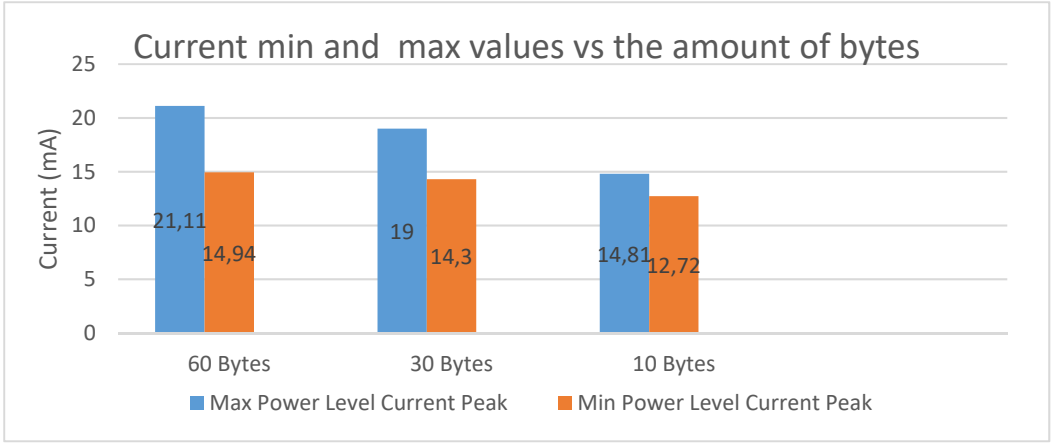

Fig. 3.10 Illustrates the power level current peaks when transmitting

After analysing Fig. 3.10 a conclusion was drawn that packet sizes of the ArduRF1's power level will not have that much effect on the radio, allowing for maximum range of the radio. Meaning that it will be pointless to add extra control bytes in a package to control the power level.

3.5 Raspberry Pi DAQ

The Raspberry Pi data acquisition system was developed to record analogue data from a scalable amount of analogue readings. For the testbed created for this dissertation the general concept was to have a few sensor nodes (eight nodes for this experiment) to be connected to a single power supply that will provide power to all the nodes via the energy monitoring circuit connected to the Raspberry Pi.

For this system to work the proof of concept was tested in previous work done as shown in Appendix E. The system performed well enough to operate with the Nyquist Theorem standards.

The system setup performed with a data rate of 3300 bps at maximum speed. The Raspberry Pi had to monitor 8 nodes with a maximum sample rate of 3300bps. Using I^2C to communicate to the module this slowed down the process. After testing the system, it derived at approximately 3 samples per 13ms. This then meant that the system settled at 307 samples of 12-bit digital data values that ended up only being 2769bps.

3.6 Raspberry Pi WSN testbed

There were a few projects concerning this dissertation in development of the WSN testbed.

- Hardware Setup;
- Package Design;
- Arduino Development;
- Raspberry Pi preparation;
- MySQL setup;
- Python Visual Display Setup; and
- MATLAB® software control.

3.6.1 Hardware Setup

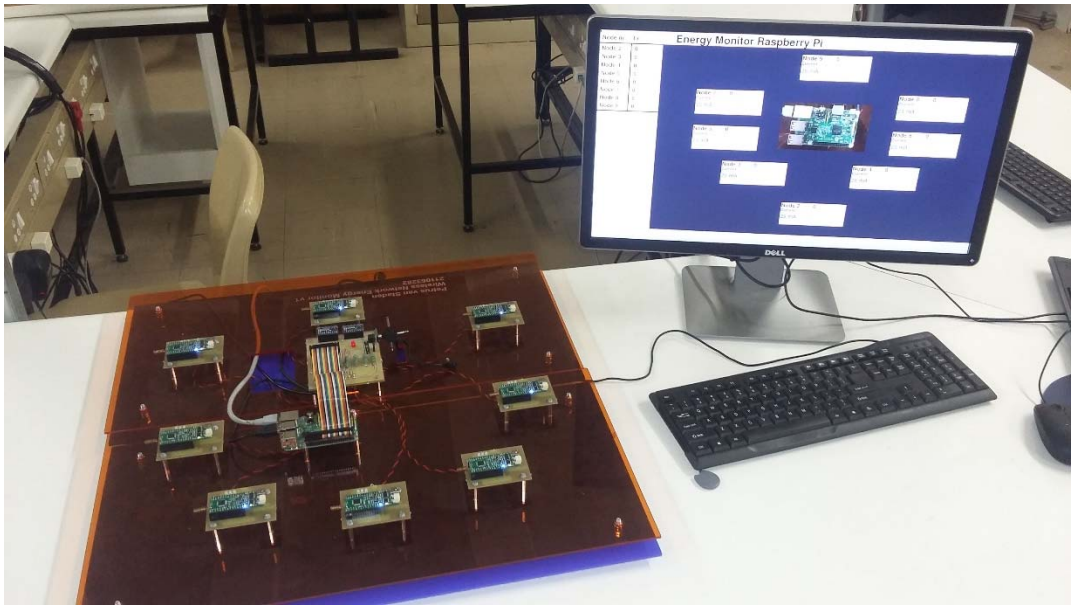

Fig. 3.11 WSN hardware Setup

Fig. 3.11 shows the entire project used in proof of this dissertation. Fig. 3.12 shows a closer image of the testbed purposefully used in this experiment with nodes numbered from 2 to 9, the Raspberry Pi 3 and the ADS115 breakout board. The design for the node breakout board was designed with 3 LEDs and an extra output pin with the purpose of indication to the Raspberry Pi when transmission is happening (Fig. 3.13).

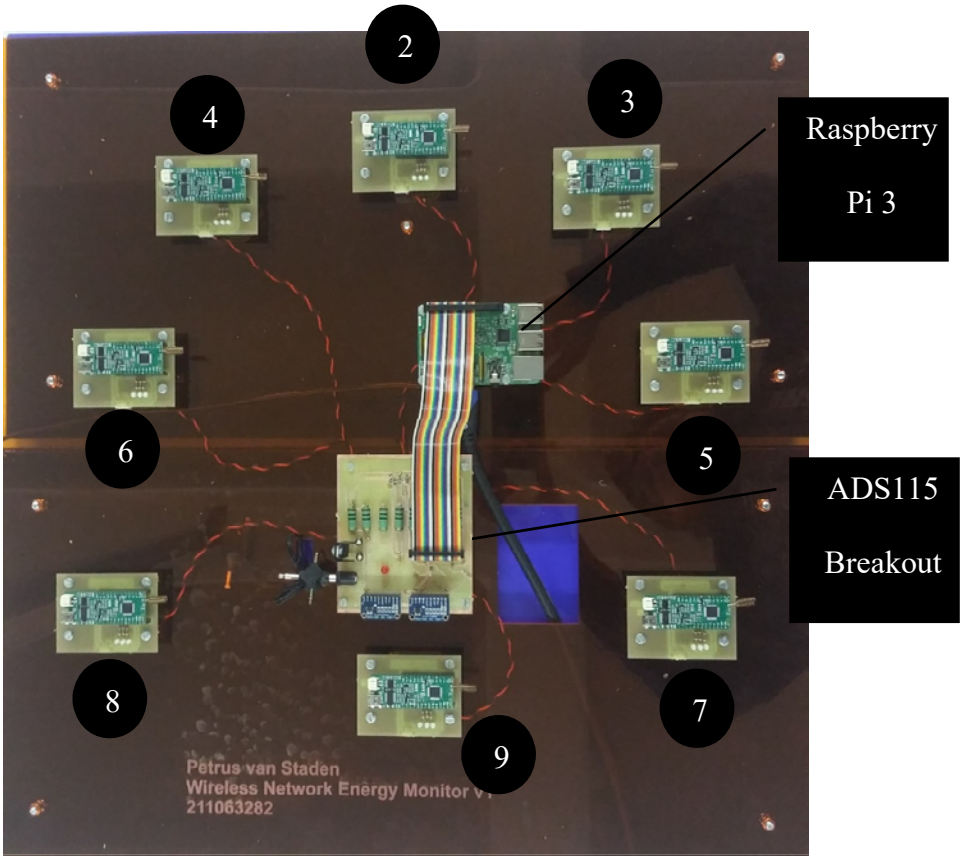

Fig. 3.12 Testbed main platform lasered from Perspex

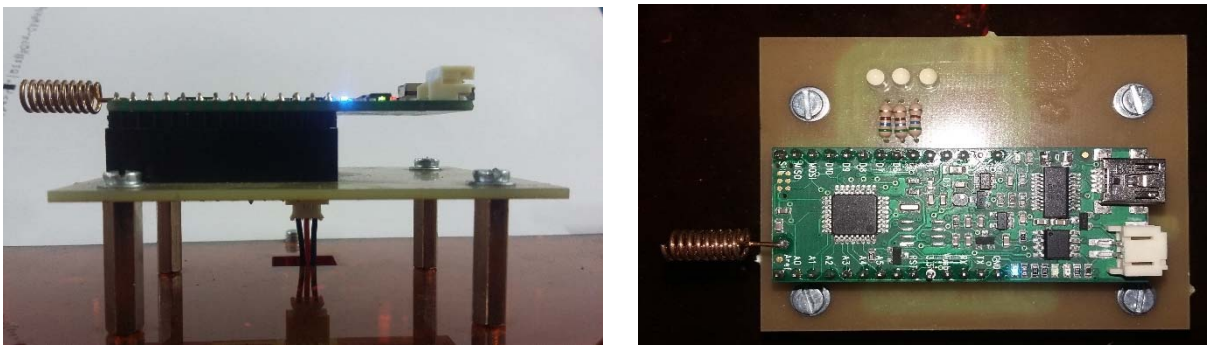

Fig. 3.13 Node platform for the ARDURF1s

3.6.2 Raspberry Pi Preparation

Raspbian Jessie is a Linux derivative, optimized to suit the Raspberry Pi Environment. The goal of the current section was to create an environment that could visually display power consumption data collected from the WSN testbed (Fig. 3.12). A 19inch LCD monitor with a custom build DAQ breakout board is attached to the Raspberry Pi.

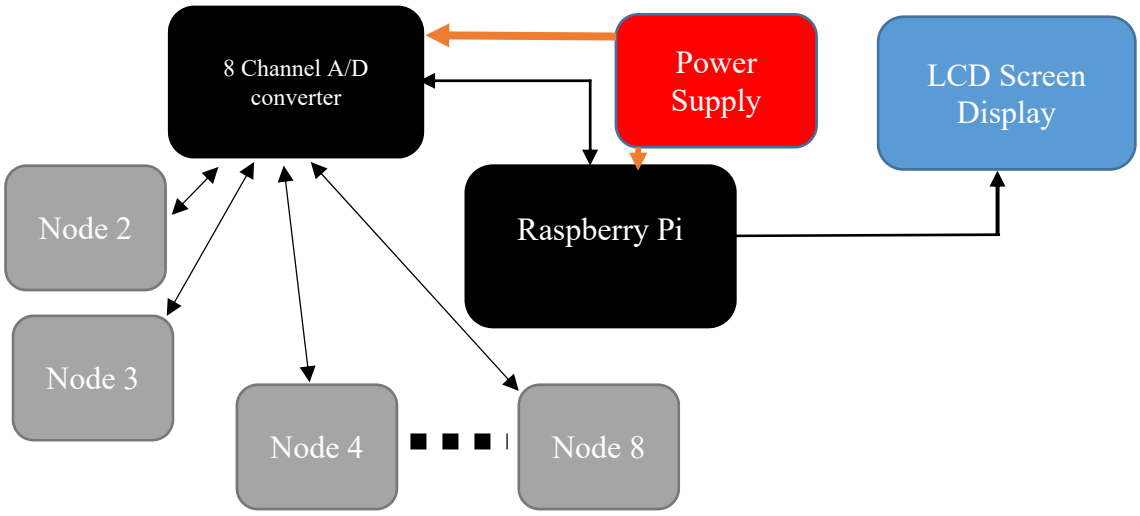

Fig. 3.14 Block diagram of the testbed developed around the Raspberry Pi

3.6.3 Python Data Logging Software Development

The current readings obtained from the testbed were logged into a MySQL database. This method was chosen to simplify the method of moving or converting data. The Raspberry Pi logged a general current reading from nodes each 5 second. This was to keep the data to a minimum. Each node changed the state of an IO pin when transmission began. This gave the Raspberry Pi a queue, initiating a high resolution of ADC sampling rate on the specific node that initiated the request (IO pin trigger event). Fig. 3.15 has a brief explanation of the flow of the python program developed for the testbed.

This was functional for determining the throughput. The problem that arose was that the collected data did not have a good start and stop functionality. When the testbed's program was not stopped extra data was collected. Further development made it possible to sync the GW node with the testbed. This meant that the GW node will only test a topology for a certain amount of time and then the testbed will terminate automatically.

The throughput data gained from the MATLAB® program could be modified by adjusting some delays on the nodes. Meaning that a 13ms delay was added on a node, before transmission changed the throughput percentage from a low number to a very high number (most of the times 100%).

This concluded the irrelevance of the throughput data being obtained from this experiment and it was thus left out.

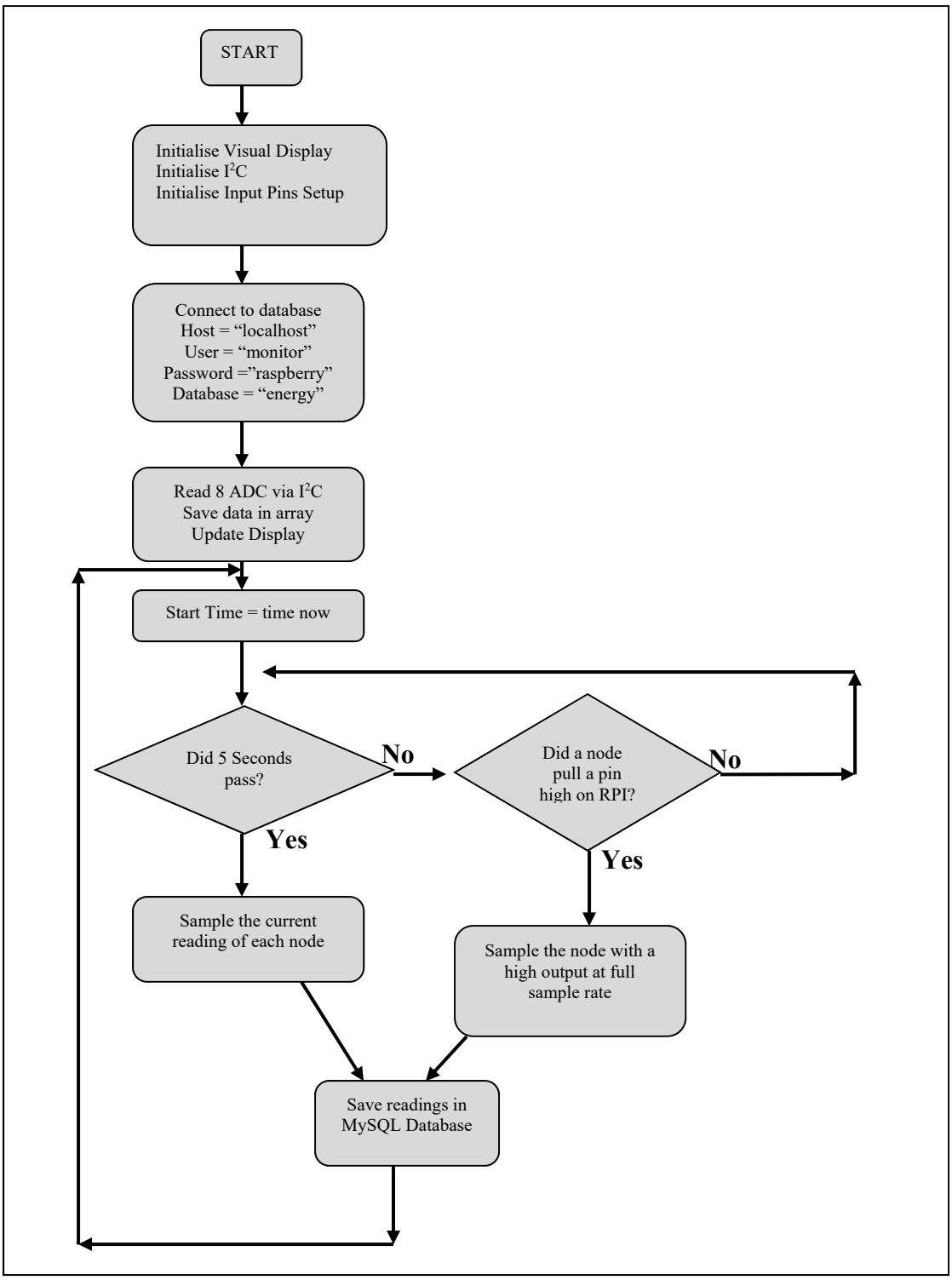

Fig. 3.15 Flow Chart of the python code running on the Testbed

3.6.4 Python Visual Interface

Fig. 3.16 shows the display that was developed on the raspberry pi using python with the open source imported pygame library (www.pygame.org). It displays the immediate current consumption at the node described. If Fig. 3.12 and Fig. 3.16 are compared it is visible that the layout of the graphical display and the layout of the physical testing bed are the same. The reason why all the nodes are not the same may be due to the nodes not all having the same amount of functional on-board LEDs.

Each time a node does a transmission, the specific node indicates with a high level GPIO pin set. This allows for the transmission count column to increase according to each transmission of each node. This allows for easy inspection of the throughput of a WSN.

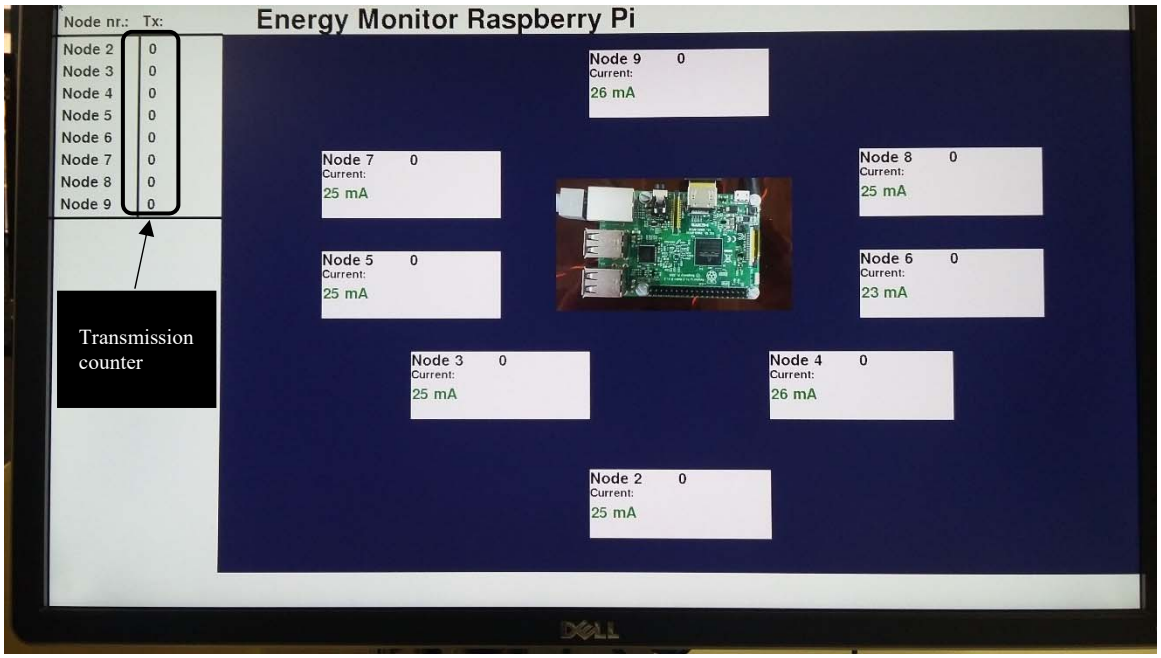

Fig. 3.16 The Raspberry Pi displays the records and the transmissions

3.6.5 Sensor Calibration

The sensor calibration was done using a resistor, to determine in an uncomplicated manner if the system is working. This resistor was measured at 217.4 Ω. When calculating the current with a voltage difference of 4.75V the theoretical current reading is 21.8mA. Fig. 3.17 shows how the wireless sensor node was disconnected to connect the resistor in series with the power source.

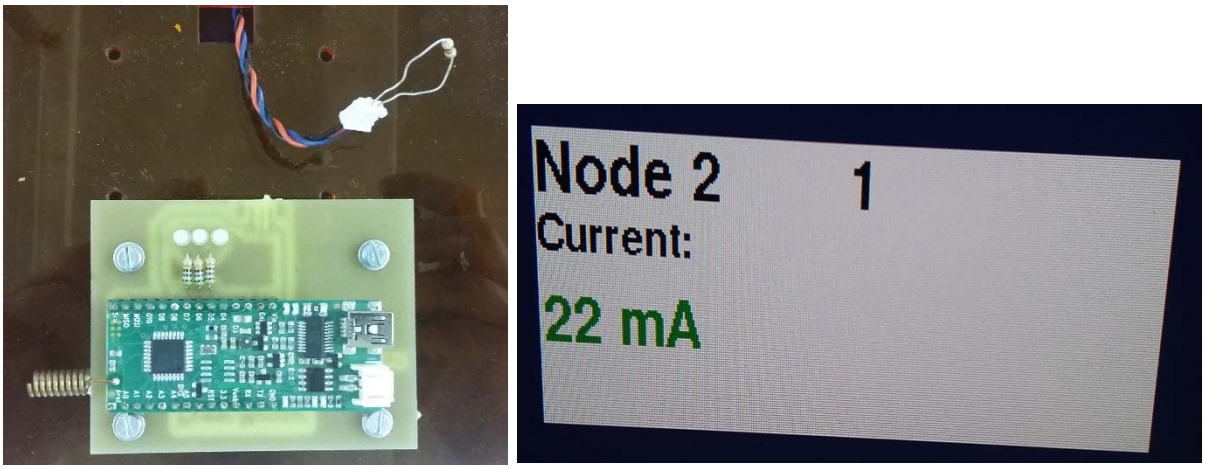

Fig. 3.17 Resistor placement with the result being displayed on the monitor

The test was done with node 2 to node 9, each showing a result of 22 mA. This means that the testbed measures current correctly, rounding the value to the nearest mA. As an extra precaution a random node was disconnected from power and replaced with a

322.4Ω resistor. The expected result was 14.75 mA. The result that showed on the screen was 15mA. This means the system is functioning correctly.

3.7 Summary

Exploratory work was done for chapter 3 to ensure that obtained current measurements from a Raspberry Pi and an ADS115 16Bit I2C ADC module are correct. This research is to conclude if a WSN will operate longer when the routing algorithm is based on the battery level of each node in the WSN. The network will be tested with 9 nodes, one being the sink node and the rest are sensor nodes. Measuring the power usage of the WSN gave viable data that didn't need a simulation to speculate the performance of a WSN's routing algorithm. Using the Raspberry Pi and an ADS115 16Bit I2C ADC module will provide an accessible method of obtaining energy measurements and logging the data. Before a testbed for monitoring energy consumption of a WSN can be build, the DAQ should operate properly on a single node. This simulation can replace actual field work and long delays for collecting data. The testbed will not measure the received signal strength indicator (RSSI) values of the nodes as the testbed is confined to limited space.

Chapter 4

WSN setup and throughput testing via MATLAB®

4.1 Introduction

This chapter observes the firmware that was developed to test the WSN and to measure the output as a result with MATLAB®. The protocol was designed to be part of the transmitted package.

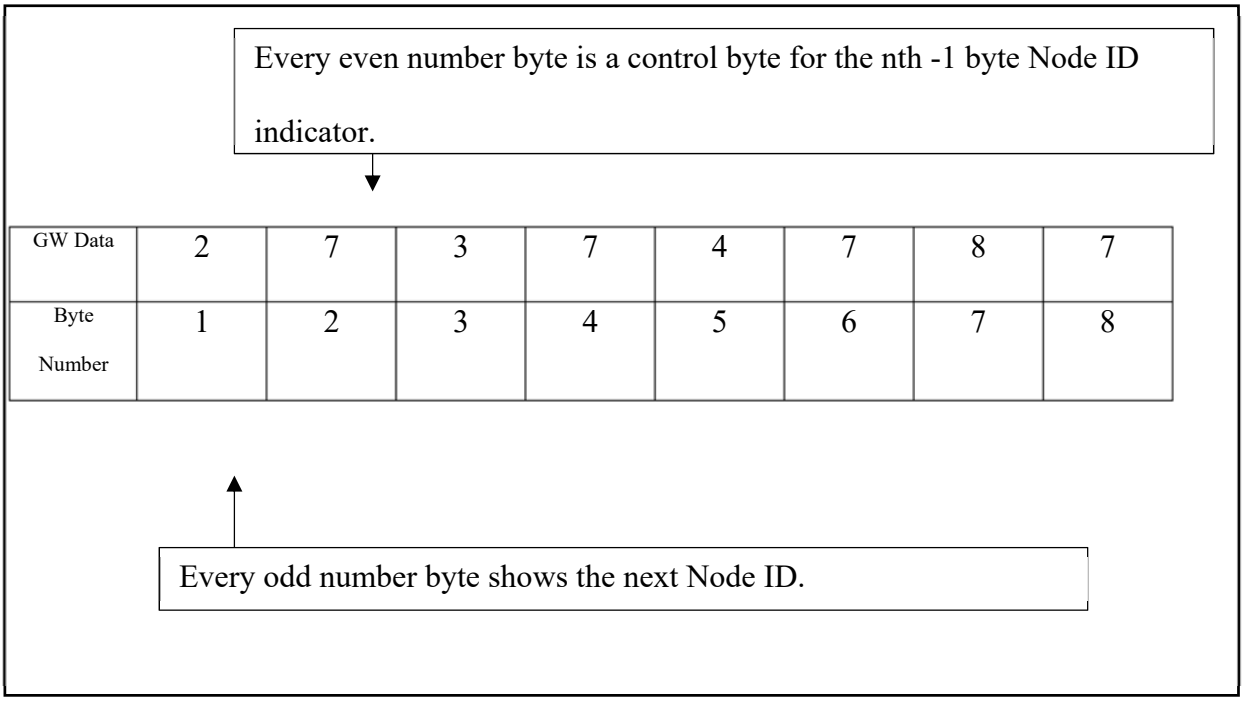

Fig. 4.1 Example of the Package Design for the Custom protocol, as inserted into the GW

4.2 Package Design for a custom protocol

When sending data from a Node to the Gateway or vice versa, a standard benchmark of 64 bytes was decided on for working in the Arduino environment. 64 Bytes is a big enough package size to transmit 51 ten bit values that is a common resolution for a 0-5V analogue reading converted to a digital value. For this dissertation a custom protocol was developed, enabling the Gateway Node to determine the route of data flow.

Fig. 4.1 illustrates a package size of 8 bytes. Looking at the values transmitted via serial to the gateway node, the first byte indicates the node that the GW transmits the data to. The second byte indicates how to switch the LEDs on the node test board. For example the 7 indicated on byte 2 will light up three LEDs on the second node test board, showing the binary value of 111.

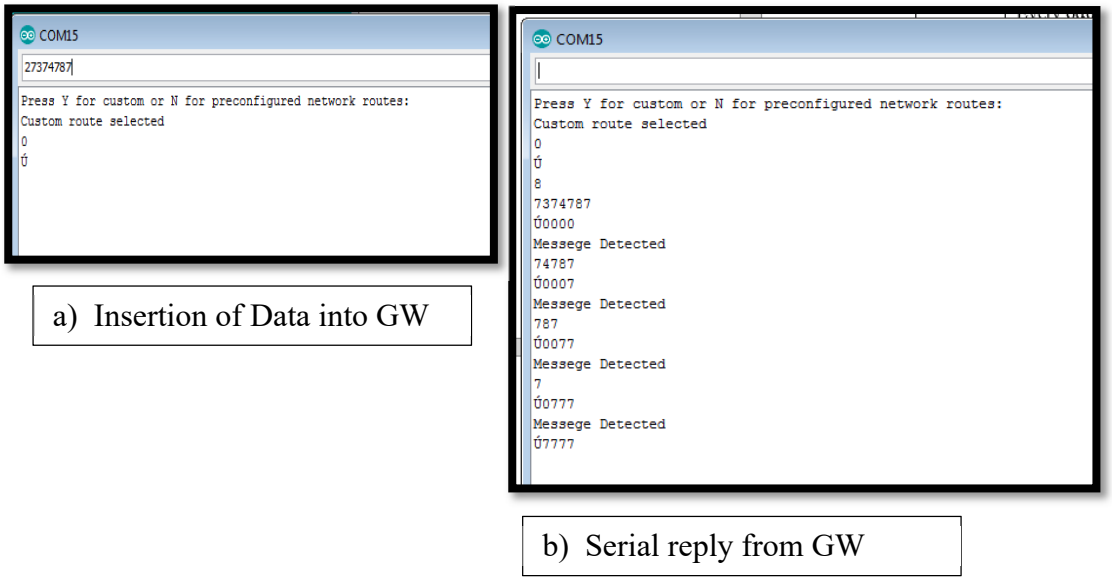

Fig. 4.2 Serial data to and from the GW node

Fig. 4.2 a) illustrates how to insert data into the Arduino IDE serial monitor. This process is also shown in Table 4-1, where row 1 is the data given to the GW via the serial monitor as shown in Fig. 4.2 a), and the last row in Table 4-1shows data that can also be obtained from Fig. 4.2 b) in the case of 100% throughput transmission.

Table 4-1 Package transmission by each node after receiving package from previous node

Node No.	Package String Value	Number of Bytes:
1	"20335077978760401"	17
2	"3350779787604010"	16
3	"5077978760401 03"	15
5	"7797876040103 0"	14
7	"97876040103 07"	13
9	"876040103 077"	12
8	"6040103 0777"	11
6	"4010307770"	10
4	"103077700"	9

Fig. 4.3 displays how data that was inserted into the serial monitor can translate to the physical testbed. The reason for developing the package in this way is to have a

package with a bit more data than only the routing information. This data can either be used for the control of the system, or for response of the sensor data.

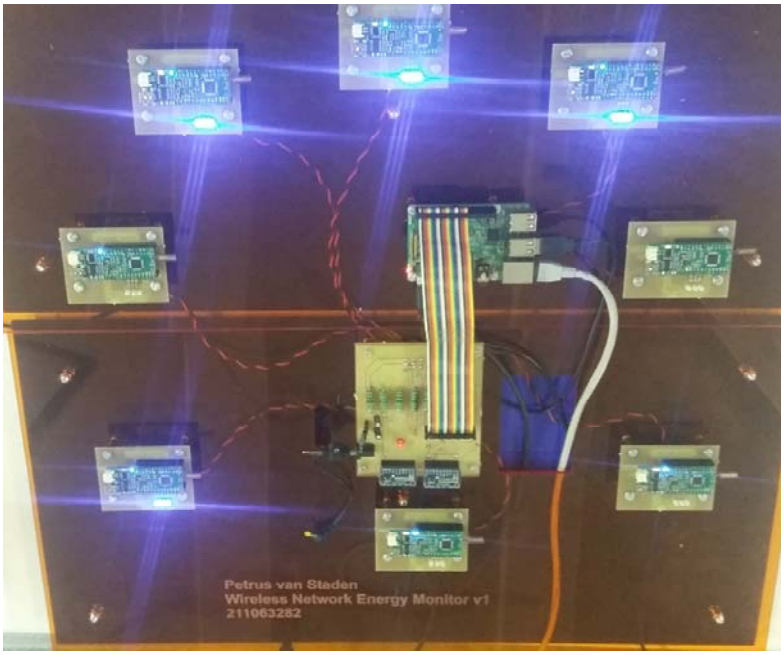

Fig. 4.3 An Example of LED's switched on wirelessly

4.3 Arduino Firmware development on the Node

At first there were three sketches for the Arduino's used in this project. There were two sketches for the gateway, one for the custom route paths, and the other for predefined route paths. The node sketch on the Arduino nodes that was only analysing packets it received. The first objective when listening to a packet was to identify if the address matched the specific node. If the node ID was matched, the Arduino then processed the rest of the packet.

The firmware developed for the node needed to be adaptable in accordance to the data it receives. The topology of the radio system is determined by the packet that originated from the gateway.

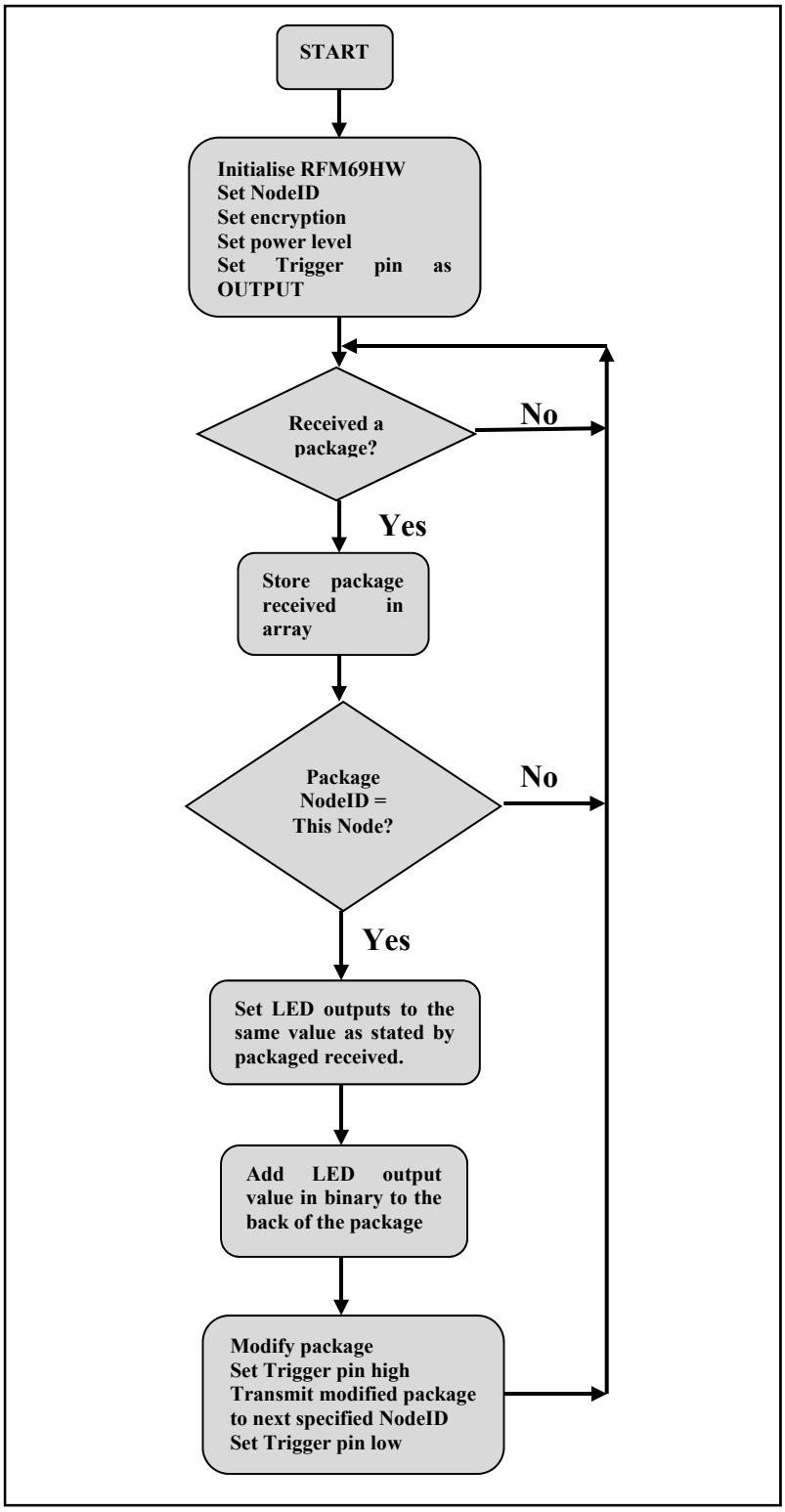

Fig. 4.4 Node firmware flow chart

4.4 Arduino Firmware development on the GW

```
switch (NodeTo){
  case 2:
 StringToChar((char*)"201");
  Transmit();
  break;
  case 3:
  StringToChar((char*)"301");
  Transmit();
  break;
  case 4:
  StringToChar((char*)"401");
  Transmit();
  break;
  case 5:
  StringToChar((char*)"3050301");
  Transmit();
  break;
  case 6:
  StringToChar((char*)"4060401");
  Transmit();
  break;
  case 7:
  StringToChar((char*)"3070301");
  Transmit();
  break;
  case 8:
  StringToChar((char*)"4080401");
  Transmit();
  break;
  case 9:
  StringToChar((char*)"2090201");
  Transmit();
  break;
  default:
  Receive();
  break;
}
Receive();
```

Fig. 4.5 Tree-network preconfigure firmware

Version two of the GW firmware was adapted to have the option of a preconfigure route for transmission, or have custom route, provided by the serial input.

The packet designed was discussed in section 4.1. As shown in Fig. 4.6 the GW waits for a serial response. The first serial response identifies if the GW will go into custom route mode, or pre-determined packet mode. The pre-determined mode needs to be edited by tree network as shown in Fig. 4.5. The “NodeTo” is inserted into the function with data received from the serial interface and will be determined by the end node of the tree network.

When changing the Tree network topology to a Ring network topology the NodeTo value was constantly 2, and the “StringToChar” variable was changed to “20305070908060401”.

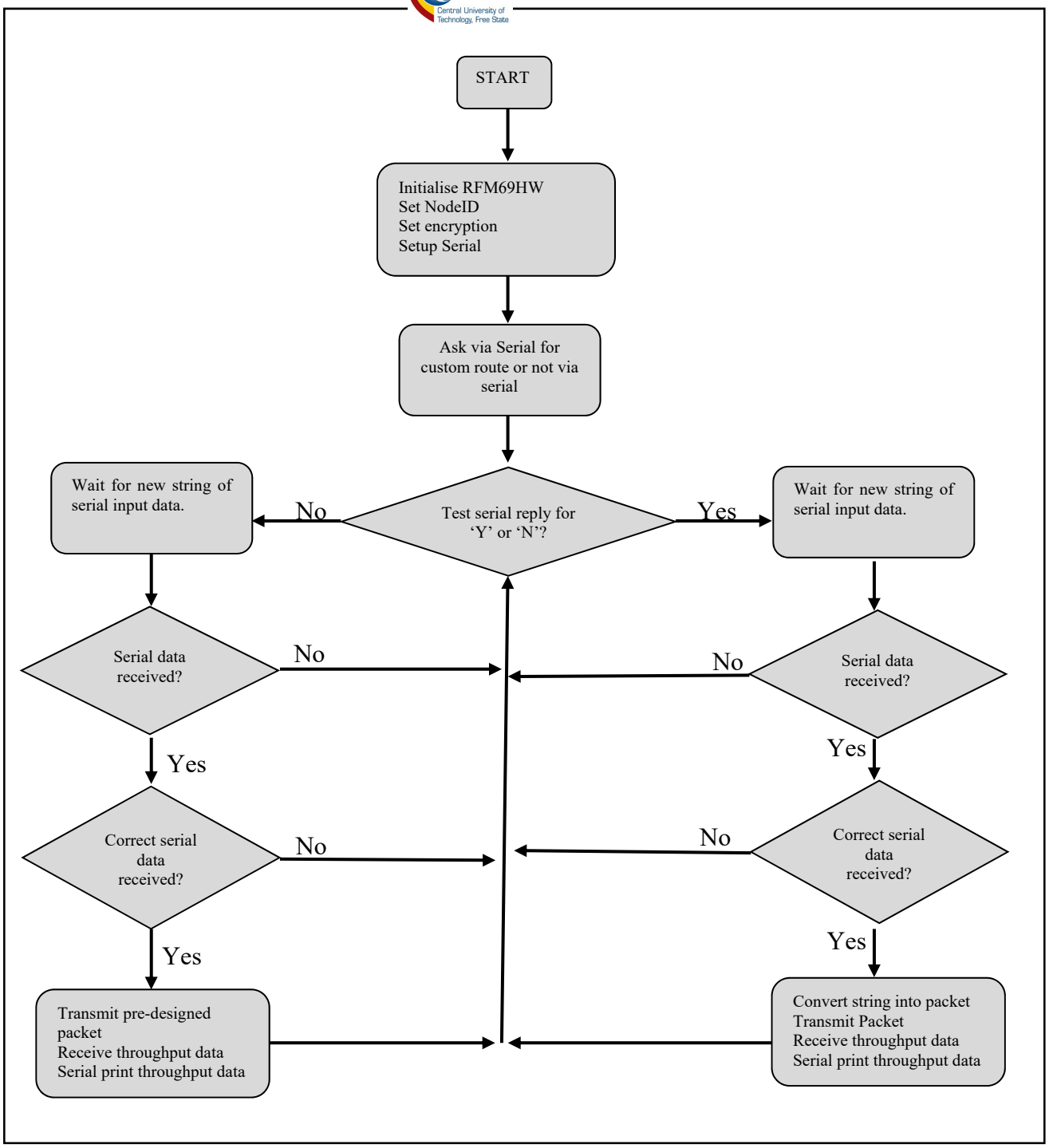

Fig. 4.6 Gateway Arduino firmware flow chart

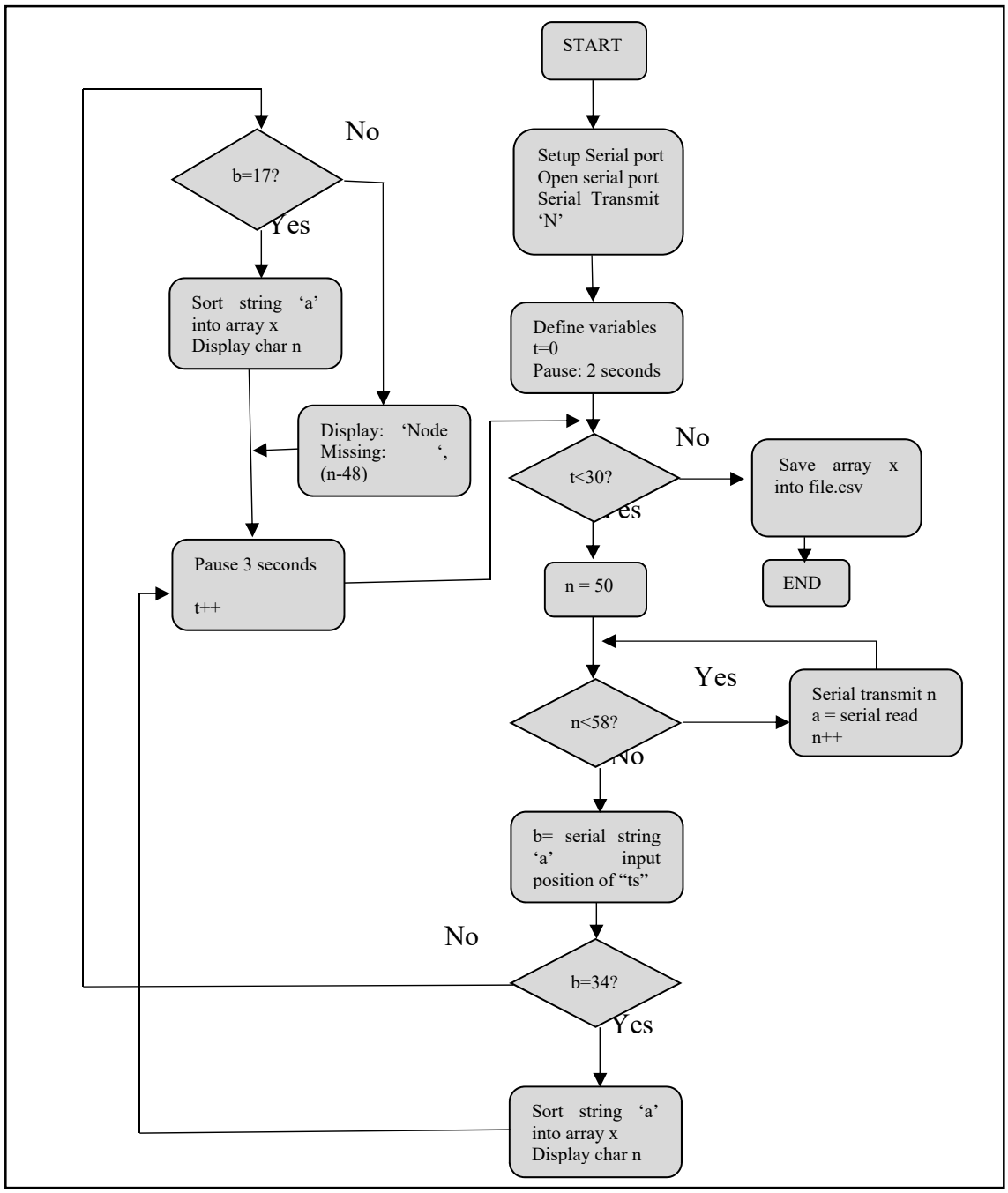

Fig. 4.7 MATLAB® Flow chart

4.5 MATLAB® Program

Fig. 4.7 is a summary of the MATLAB® code written to analyse the throughput of the data, also requesting a transmission. This specific example does the loop 30 times and then saves the data on a .CSV file. 30 loops seemed sufficient for obtaining enough results. The computer talks via serial to the ArduRF1s Gateway module. Giving it an ASCII command valued between '0' and '8'. The gateway will reply to the computer via serial with the throughput results of the requested transmission. This only works with the preconfigured routes already programmed into the gateway, as explained in section 4.3.

4.6 Testing Procedure for basic topology test

For a basic topology test the system was setup to do two methods. The topology is setup on the Gateway processor and the number of runs is setup on the MATLAB® program that interfaces the throughput data of the Gateway.

Step 1:

Decide which topology needs to be used. For scenario 1 the gateway was setup to do the tree network topology. This is changed by manipulating a variable in the firmware for the gateway. Upload the firmware once the variable is changed.

Step 2:

Start the GUI on the Raspberry Pi. This initiates the DAQ where the data is saved in the MySQL database.

Step 3:

Decide on the amount of runs that the testbed needs to do and edit the variable on the prepared MATLAB® code. This value must be kept consistent with all topology tests that have to be done to get comparable data. Connect the gateway node to the computer that runs the MATLAB® script to ensure that the com ports are correct in the script and run it. The physical testbed also needs to be within range.

Step 4:

After the MATLAB® script has finished its routine, the throughput data will be displayed in the command window. This data is then copied and saved into a text file.

Step 5:

In this step data is extracted from the MySQL database and saved in .CSV files. Each node has a .CSV file that contains timestamps and current readings.

Step 6:

Import the data into MATLAB® and draw the figures necessary for evaluation of data.

4.7 MATLAB® visual data created with .CSV files

The code that turned all the necessary data into graphs is shown in Appendix B. The logged data of each node was used to generate a graph that can be placed in the results. Because there was a lot of data only one of the topology graphs was inserted in the results chapter.

Chapter 5

Communication and power results obtained by experimental setup

5.1 Introduction

In this chapter results that was retrieved from the DAQ system is displayed and explained.

5.2 Tree network:

The basics of a tree network topology are for bi-directional communication to the previous or next node. The route the packages follow to the end node is the same route the packages will follow in reverse. E.g. if the gateway wants to transmit info to node number 9 the route for this example is from GW to node 2-3-9. When node 9 wants to transmit data, it follows the same route, but only in reverse. Fig. 5.1 displays how the

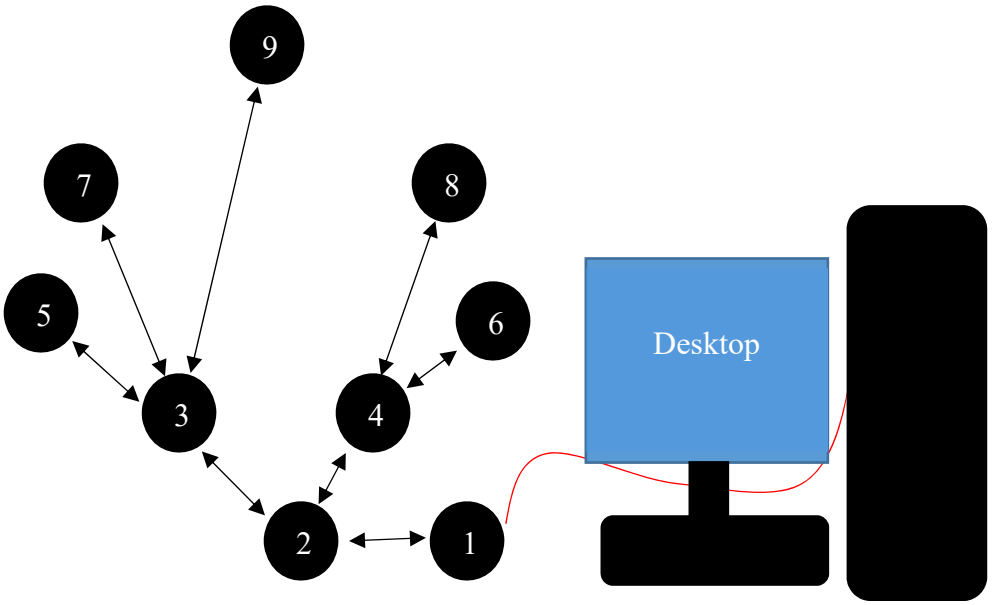

Fig. 5.1 Tree Network Topology with Packet paths indicated by arrows

setup of the current testbed looks like.

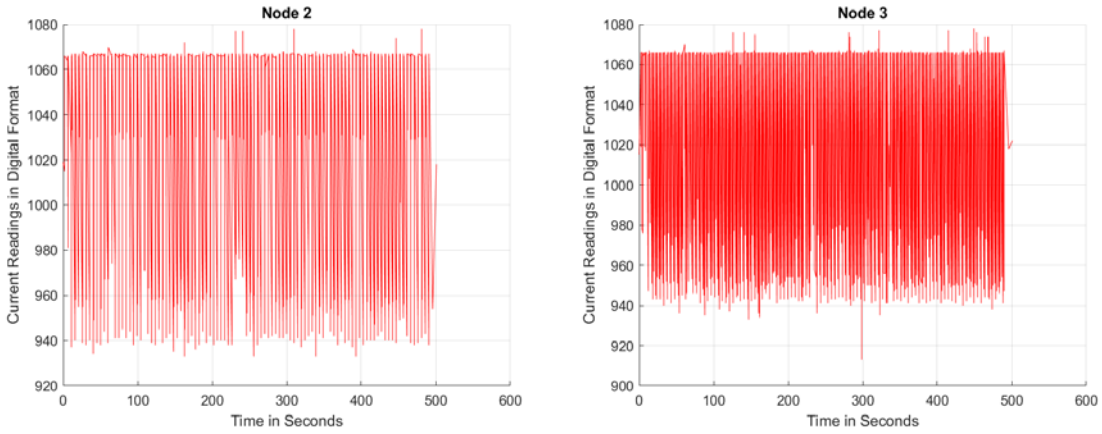

Fig. 5.2 Node 2 and 3 current reading in digital format

5.2.1 Data of the Tree Network measured and displayed:

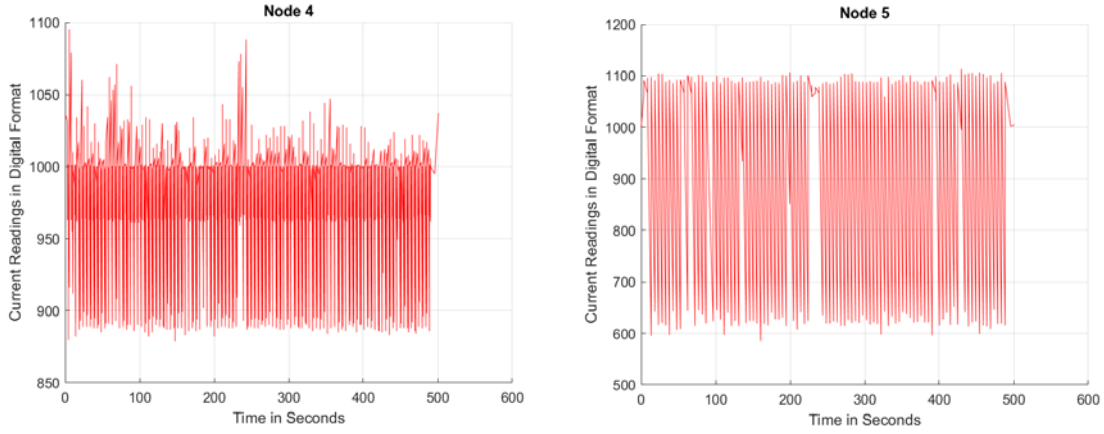

Fig. 5.3 Node 4 and 5 current readings in digital format

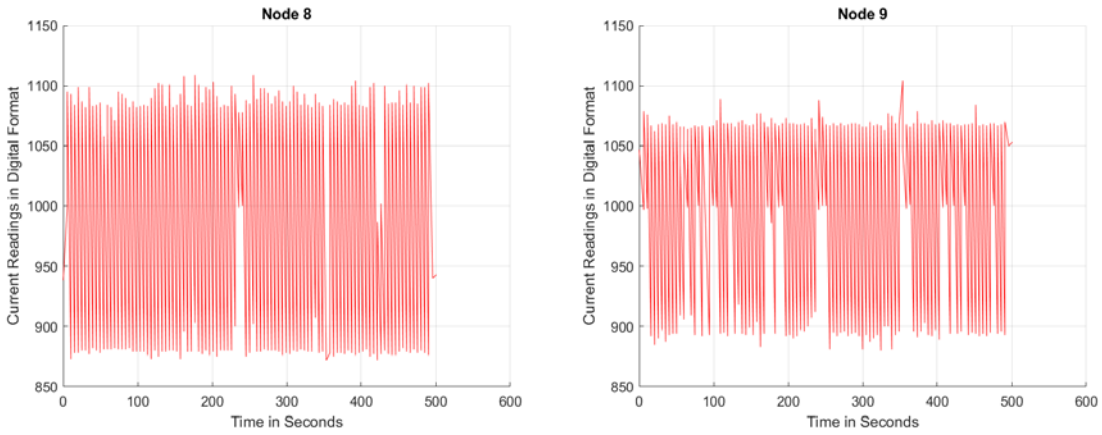

Fig. 5.5 Node 8 and 9 current readings in digital format

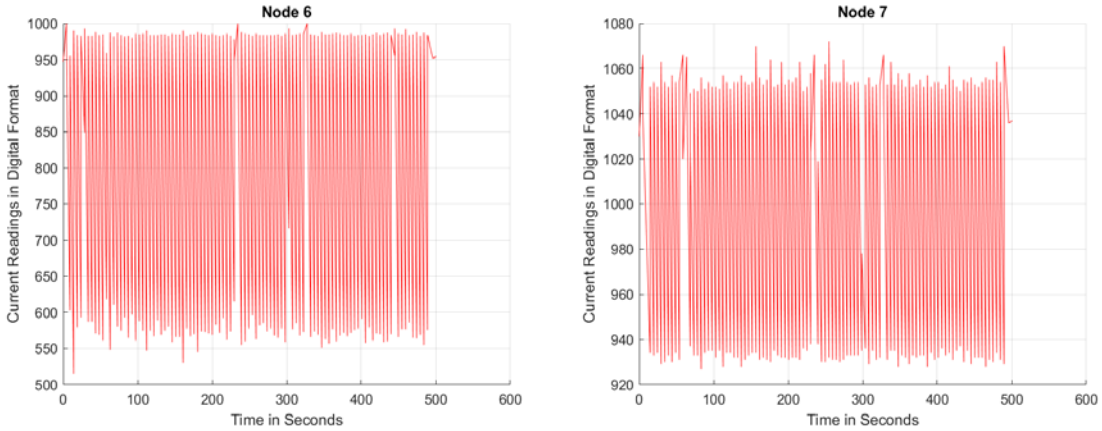

Fig. 5.4 Node 6 and 7 current readings in digital format

Fig. 5.2; Fig. 5.3; Fig. 5.4; and Fig. 5.5 display the raw data collected from the testbed (Fig. 3.13). Using the instructions setup in section 4.6 this data was retrievable. The raw data is too much to get a clear picture of the energy consumption. Instructions described in section 4.7 were used to create the visual data used in the results.

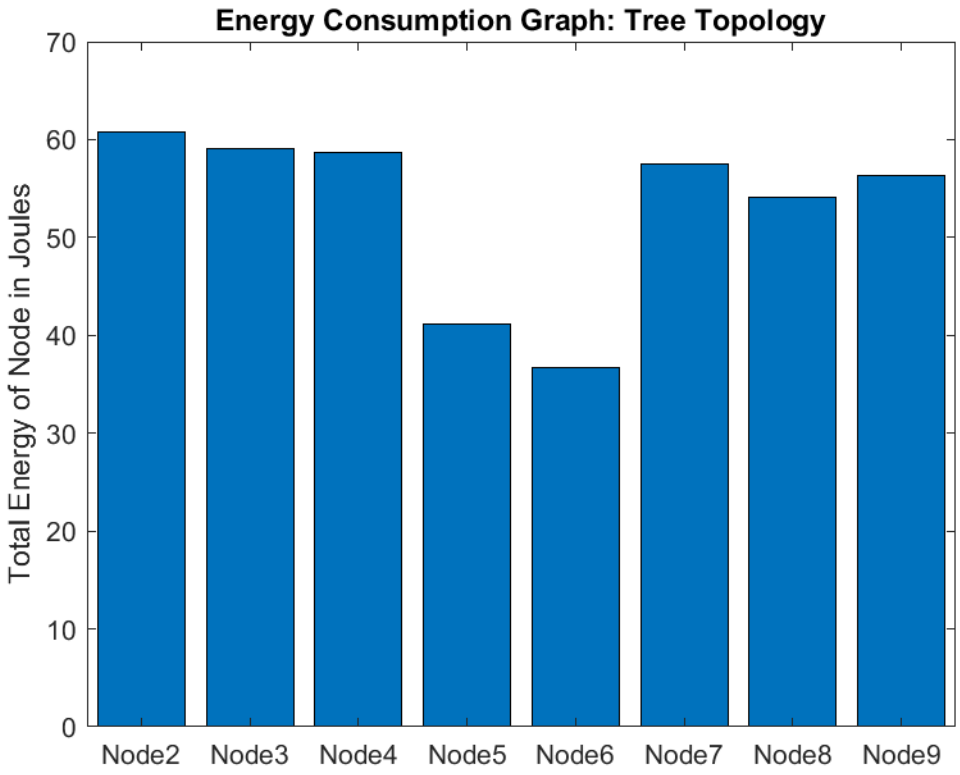

Fig. 5.6 Total Energy bar graph for a tree topology

Observing Fig. 5.6 in accordance with Fig. 5.1 shows that node 5 and node 6 uses the least amount of energy during the test duration. This is a clear indication of inconsistency in the nodes that are used on the testbed, theoretically node 5,6,7,8 and 9 should use the same amount of energy, that should be a little bit less the what node 3 and

4 uses for the same test. Node 2 should use the most energy (as also indicated in Fig. 5.6), because it handles the most data.

5.3 Ring Network

The ring network topology is a transmission of data across the entire network. The path was from node 1 to node 2, 3,5,7,9,8,6,4 and back to one. The first test was to test the ring network 240 times. Fig. 5.7 shows the ring network topology on the self-developed radio network testbed.

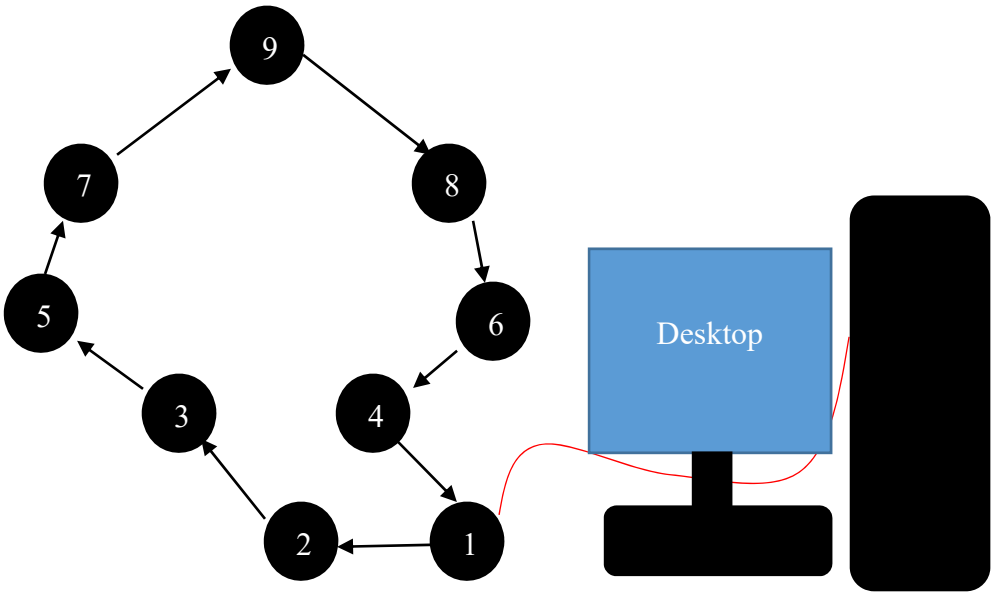

Fig. 5.7 Ring Network Topology with Packet paths indicated by arrows

The ring network topology is better when nodes 3 and 5 to 9 are out of reach of the gateway node. Even though the distance gain might be a positive feature, if one node malfunctions, the integrity of the network is compromised.

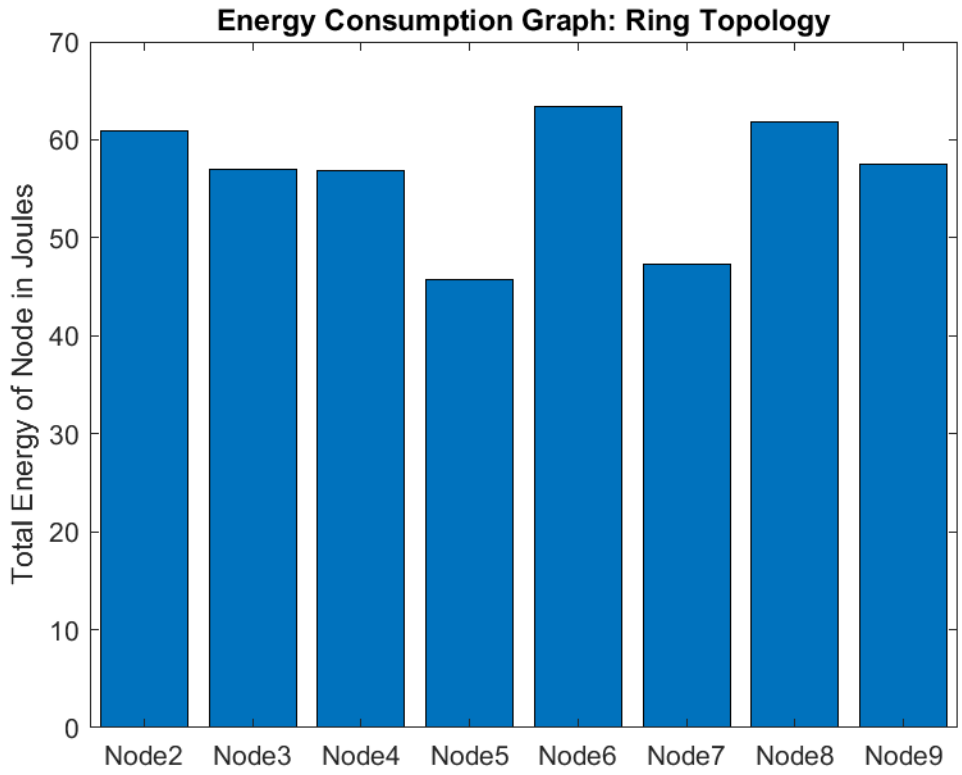

Fig. 5.8 Total Energy bar graph for a ring topology

5.4 Star Topology

The star network topology (Fig. 5.9) can only be plausible when all the nodes can transfer data to the central gateway node. All the nodes should also not transmit at the same time because a single node will not be able to handle all the data at the same time, or the data can be scrambled. On the positive side, none of the nodes are dependent on one another, if one node is malfunctioning, the system will still work.

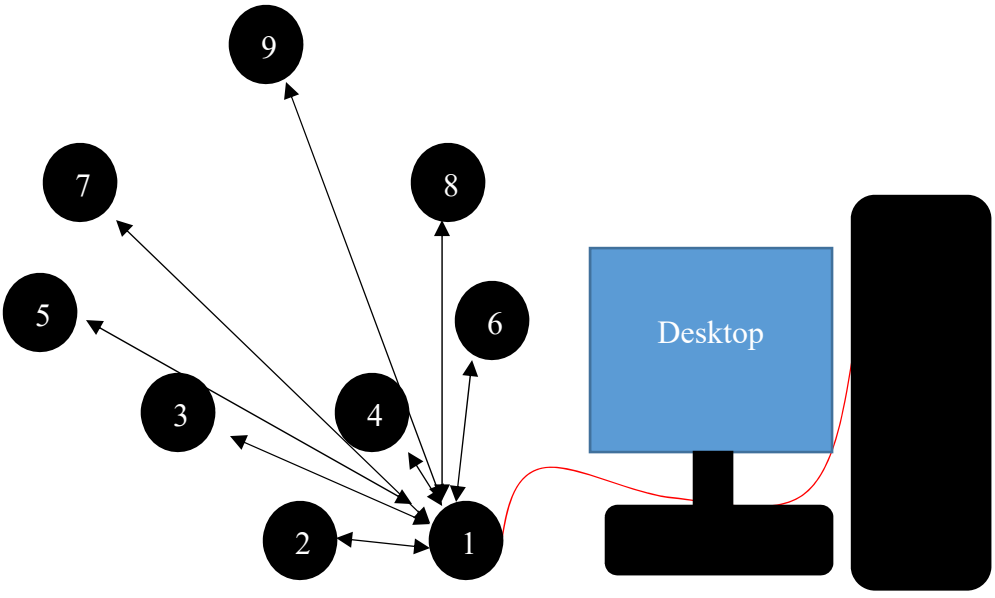

Fig. 5.9 Star Network Topology with Packet paths indicated by arrows

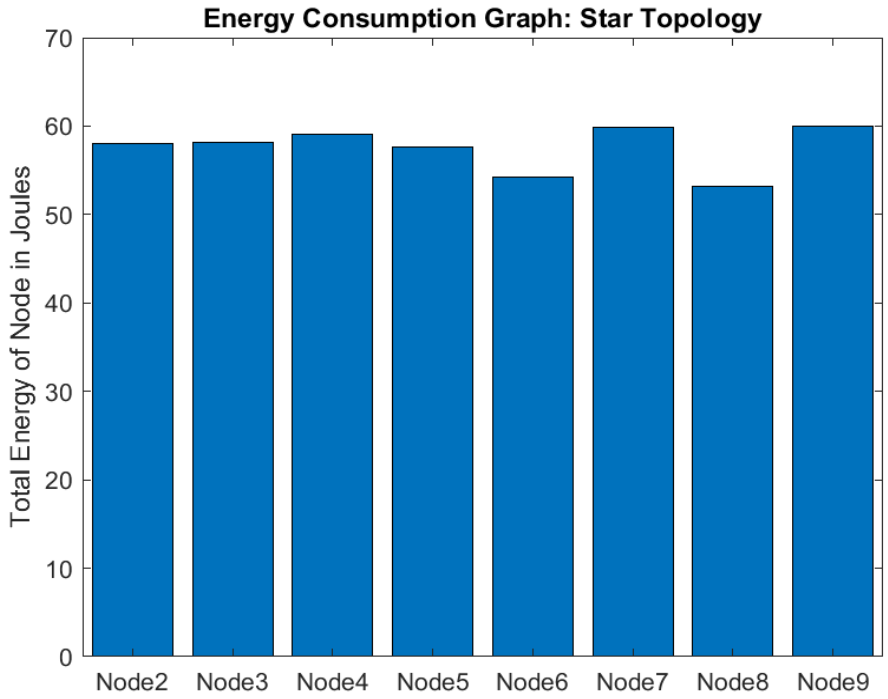

Fig. 5.10 Total Energy bar graph for a star topology

From the theoretical data of Fig. 5.Fig. 5.10 it could be expected that each node would use the same amount of energy, but the bar graph indicates that 6 and 8 use a bit less energy and the others a bit more than the expected average.

5.5 Star Tree Topology

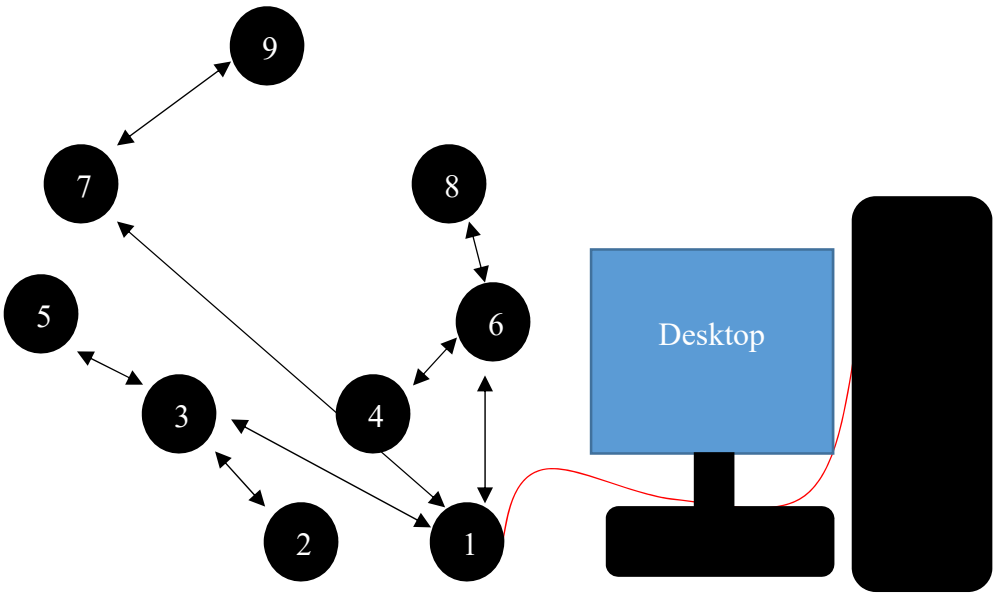

Fig. 5.11 Star Tree Network Topology with Packet paths indicated by arrows

Fig. 5.11 shows the configuration of a star tree network topology. This test was done to combine two topologies and measure the outcome based on energy consumed. Fig. 5.13 gives the energy per node produced by the test that was run on the testbed.

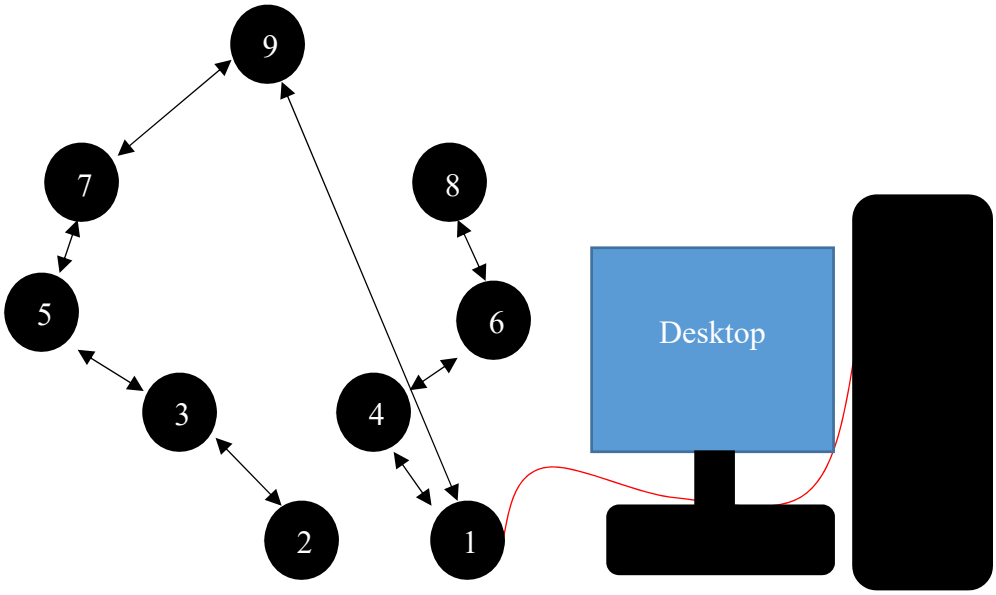

Fig. 5.12 Random Network Topology with Packet paths indicated by arrows

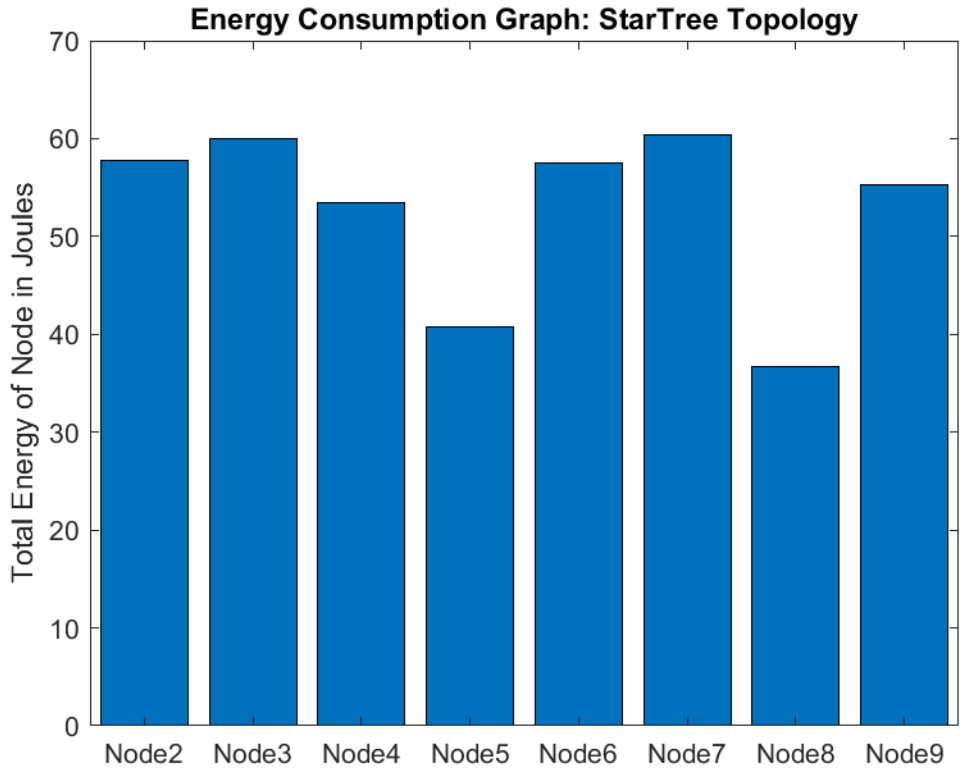

Fig. 5.13Total Energy bar graph for a star tree topology

5.6 Random Topology

This topology (Fig. 5.12) was added to the testbed to have a benchmark to test if the other topologies have a better advantage over a logically impotent topology. Fig. 5.14 shows that most of the nodes used a bit more energy than 60 Joules, but with abnormal low energy usage by node 6.

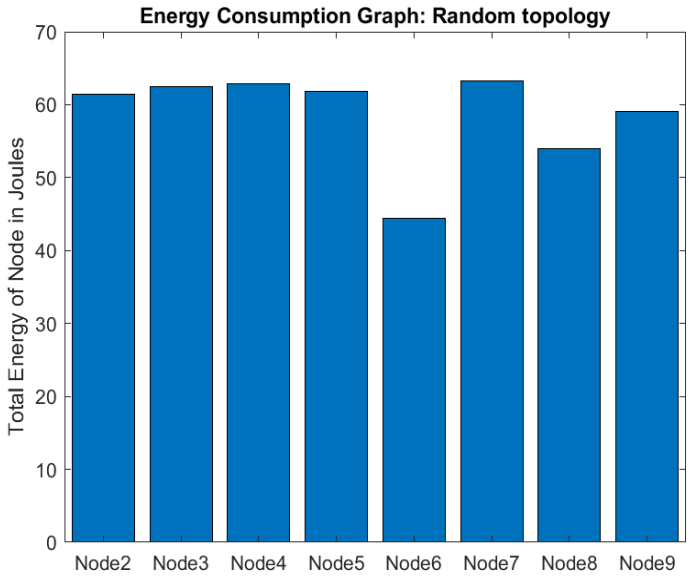

Fig. 5.14 Total Energy bar graph for a self-designed topology

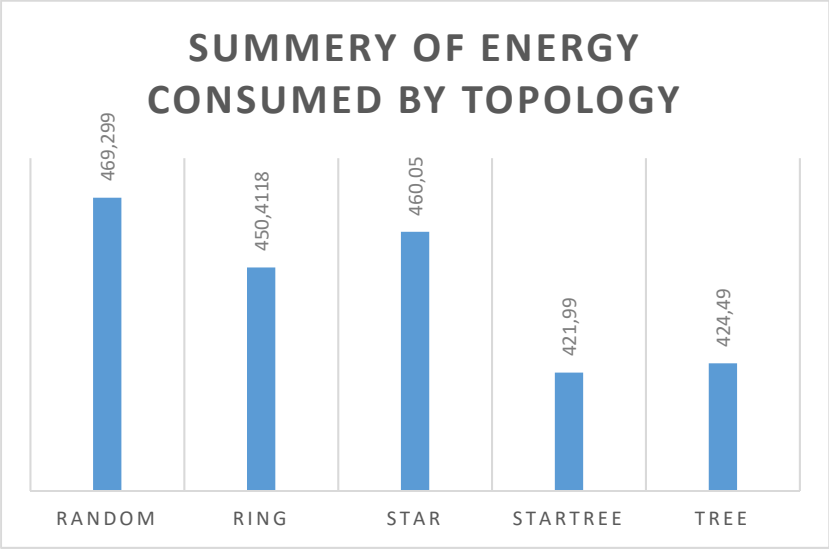

Fig. 5.15 Total energy consumed by each topology test

5.7 Total energy consumed by topologies

Adding the energy of each node categorized by each topology, the total amount of joules is displayed in Fig. 5.15. This info is useful specifically to the nodes that were used on this circuit. The star tree topology used the least amount of energy, 421.99J.

Fig. 5.16 shows the maximum energy consumed by a node in the specified topology. This data is crucial in determining the lifetime of a specific topology as a chain is most weak at the weakest link.

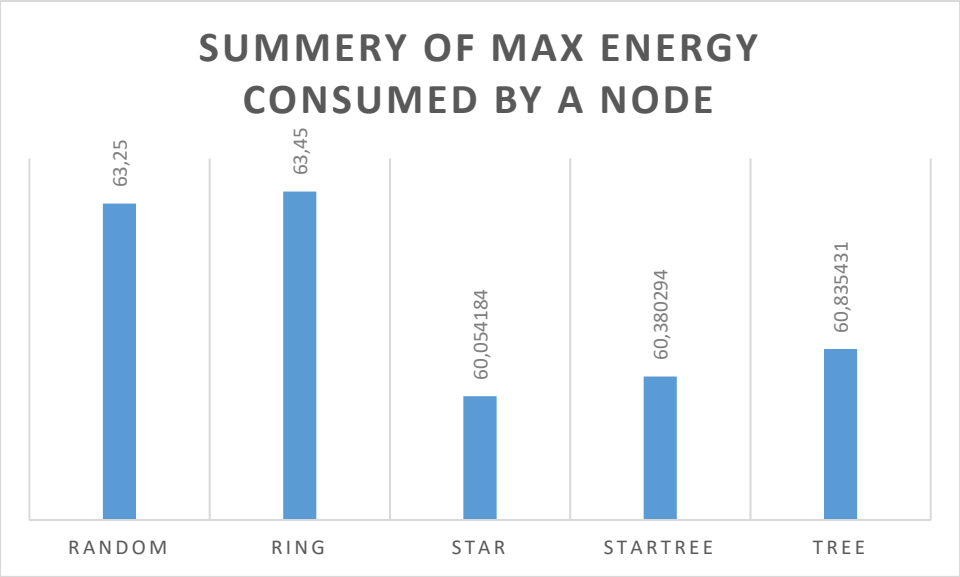

Fig. 5.16 Maximum energy consumed by a node in a certain topology configuration

5.8 Results evaluation

Using prediction software and simulation software for testing the longevity of a wireless network it can be complicated and difficult to do everything in theory. Sometimes the issue might be that all the variables are unknown. Radio modules operating on the open licenses are very popular in the era of IoT. With a lot of open source codes available to program these modules it can be easily utilised by using Arduino hardware implementation.

5.8.1 Single Node Evaluation of Nodes in a Topology

Fig. 5.6, Fig. 5.8, Fig. 5.10, Fig. 5.13 and Fig. 5.14 looks at the total energy consumption of a node inside a topology. On viewing these bar graphs a few abnormalities is visible, that is not explainable just by evaluating the data. If this test was done in theory the star topology analysis would have resorted in each node using the same amount of energy.

5.9 Conclusion based on the results

The goal was to analyse energy consumption with different topologies to discern which topology would be more efficient when it comes to depleting the energy sources of each node in the network. The results have shown that a star-tree configuration used the least amount of energy in total (Fig. 5.15). This is a bad conclusion taking into account the duration of time that this topology will be working. A chain is only as strong as its weakest link. This implies that if one node is down, the network is down. Fig. 5.16

demonstrates the theory that a chain is as strong as its weakest link and revealed that a star topology optimised the network to have a node that consumed the maximum energy to be 60.05 Joules. The crux of a star topology is that it limits the range of nodes. Each node only knew the route to the next node once upon the reception of a viable package. The more complicated the route that a packet needed to travel, the bigger the package size was to hold the route information.

Chapter 6

Conclusion

6.1 Introduction

After developing a basic wireless node network that could be reconfigured to test different topologies, it was equally important to find a way to capture data produced by the network. In this case the energy consumption data was important in order to evaluate the duration of a network's possible life time. An Artificial Reconfigurable wireless network optimization based on battery levels, was semi-simulated in the lab. The wireless network was artificially reconfigurable, but instead of using physical batteries, a testbed was developed to provide and monitor the power of each node in the network.

6.2 Summary

In this research the following results were achieved:

1. A reconfigurable wireless network was developed, acting as a real network;
2. Testing methods for measuring and evaluating energy of a node was introduced;
3. Development of a testbed based on tested methods;
4. Proof of functionality of the testbed;
5. Proof of which topology used the least amount of energy; and

6. Conclusion about the longevity of a radio network.

Evaluation of the throughput data was also briefly analysed, but the results were not constant and therefore it was left out of this study.

6.3 Final Remark

This study observed wireless sensor networks and the energy consumption of a wireless sensor network. In industry the artificial reconfigurable network based on energy consumption can be used in some applications, but it might not be a popular option as it is customary to develop a network sensor with a low power source. The industry is currently focussing on developing low energy consumption radio networks of which Bluetooth Low Energy (BLE) is a good example.

6.4 Future studies

By using available electronics it is possible to measure the power consumption of a Wireless Sensor Network (WSN) to determine which topology might have an improvement on battery life and how to set up an efficient protocol. This action simplifies the prediction calculation of the radio network's lifetime. If the power source is only dependant on a battery, in the case of a deployed WSN, this is useful in the prediction of site visits. Improvements of this study can be done by adding extra environmental tests to the testbed. The testbed designed and used for this study could not test throughput thoroughly, as each node was near the next one and the system could not,

unless further development was initiated on the testbed, identify where the link broke in the communication between the nodes,.

References

[1] T. Warger, “The Open-Source Movement,” *Educause Quarterly*, pp. 18–20, 2002.

[2] L. P. Steyn and G. P. Hancke, “A survey of Wireless Sensor Network testbeds,” in *IEEE Africon ’11*, 2011, pp. 1–6.

[3] P. van Staden and B. Kotze, “Wireless node energy monitor using common development platforms: Using a Raspberry Pi to monitor energy consumption by a WSN,” in *2017 IEEE AFRICON*, 2017, pp. 1514–1519.

[4] CertBros, *OSI Model Explained | Real World Example - YouTube*. .

[5] “ISO/IEC 7498-1:1994 - Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model.” [Online]. Available: https://www.iso.org/standard/20269.html. [Accessed: 20-Nov-2018].

[6] IEEE Computer Society, *IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture*, vol. 2014. 2014.

[7] N. Heap, *An Intoduction to OSI*. Blackwell Scientific Publications, 1993.

[8] W. A. Melendez and E. L. Petersen, “The upper layers of the ISO/OSI reference model (Part II) 1The production of this document was financed under contract by the Commission of the European Communities. All property rights belong to the Commission of the European Communities. Part I of the original publication of this paper appeared in volume 5, issue 1 of Computers Standards & Interfaces. 1 2Originally published in Computer Standards and Interfaces, volume 5 (1986), pages 65–77. 2,” *Comput. Stand. Interfaces*, vol. 20, no. 4–5, pp. 185–

199, Feb. 1999.

[9] H.-N. Dai, "Throughput and Delay in Wireless Sensor Networks using Directional Antennas," in *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2009.

[10] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *J. Netw. Comput. Appl.*, vol. 60, pp. 192–219, Dec. 2015.

[11] J. Fagot, P. Magne, D. W. Fry, and W. Higinbotham, *Frequency Modulation Theory : Application to Microwave Links.* Elsevier Science, 2014.

[12] SigFox, "Coverage | Sigfox." [Online]. Available: https://www.sigfox.com/en/coverage. [Accessed: 20-Nov-2018].

[13] P. Thulasiraman and K. A. White, "Topology control of tactical wireless sensor networks using energy efficient zone routing," *Digit. Commun. Networks*, vol. 2, no. 1, pp. 1–14, 2016.

[14] S. Bouarafa, R. Saadane, and D. Aboutajdine, "Reduction of energy consumption in WSN using the Generalized Pythagorean Theorem," in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, 2016, pp. 720–725.

[15] N. F. Shah, Amritanjali, S. Gautam, and D. Gosain, "EERA: Energy efficient reliable routing algorithm for WSN," in *2016 1st India International Conference on Information Processing (IICIP)*, 2016, pp. 1–5.

[16] Yun Sik Kim and Chul Hye Park, "Traffic adaptive handover management in ATM-based LEO satellite networks," in *1999 IEEE 49th Vehicular Technology*

Conference (Cat. No.99CH36363), vol. 3, pp. 2466–2470.

[17] P. Dawei, P. Yu, and P. Xiyuan, "A testbed for the evaluation of low power protocols in wireless sensor networks," in *2011 IEEE International Instrumentation and Measurement Technology Conference*, 2011, pp. 1–4.

[18] G. Girban and M. Popa, "WSN testing environment with energy consumption monitoring and simulation of sensed data," in *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, 2011, pp. 842–847.

[19] A. Kopke and A. Wolisz, "Measuring the Node Energy Consumption in USB Based WSN Testbeds," in *2008 The 28th International Conference on Distributed Computing Systems Workshops*, 2008, pp. 333–338.

[20] Jun Hwan Huh, Dong Hyun Kim, and Jong-Deok Kim, "NEWSBED: The Internet of Things testbed platform," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 492–494.

[21] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: a wireless sensor network testbed," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pp. 483–488.

[22] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, "WISEBED: An Open Large-Scale Wireless Sensor Network Testbed," Springer, Berlin, Heidelberg, 2010, pp. 68–87.

[23] A. Zoha, A. Gluhak, M. Imran, and S. Rajasegarar, "Non-Intrusive Load

Monitoring Approaches for Disaggregated Energy Sensing: A Survey," *Sensors*, vol. 12, no. 12, pp. 16838–16866, Dec. 2012.

[24] G. W. Hart, "Nonintrusive appliance load monitoring," *Proc. IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[25] M. Baranski and J. Voss, "Non-intrusive appliance load monitoring based on an optical sensor," in *2003 IEEE Bologna Power Tech Conference Proceedings,* vol. 4, pp. 267–274.

[26] Microchip, "Low-Power Long Range LoRa® Technology Transceiver Module." 2017.

[27] I. F. Akyildiz and M. C. Vuran, *Wireless sensor networks*. John Wiley & Sons, 2010.

[28] "2. Using Python on Unix platforms — Python 2.7.14 documentation." [Online]. Available: https://docs.python.org/2/using/unix.html. [Accessed: 15-Jan-2018].

Appendix A Python Code

```
#       prevLOG[i][1]=curLOG[i][1]
#       prevLOG[i][0]=curLOG[i][0]

#       powerDisplay[i]= powerDisplay[i]+ ((((displayLOG[i][1]*0.000025)*displayLOG[i][0])*5))
for event in pygame.event.get():
        if event.type == pygame.KEYDOWN():
                time.sleep(20)
                gameExit = True

gD.fill(white)
#font = pygame.font.Font(None, 36)
Title =header.render("Energy Monitor Raspberry Pi", False, black)
a=col.render(("Node 2         %d"%(TXcounter[0])),False,black)
b=col.render(("Node 3         %d"%(TXcounter[1])),False,black)
c=col.render(("Node 4         %d"%(TXcounter[2])),False,black)
d=col.render(("Node 5         %d"%(TXcounter[3])),False,black)
e=col.render(("Node 6         %d"%(TXcounter[4])),False,black)
f=col.render(("Node 7         %d"%(TXcounter[5])),False,black)
g=col.render(("Node 8         %d"%(TXcounter[6])),False,black)
h=col.render(("Node 9         %d"%(TXcounter[7])),False,black)
Joules=subtxt.render("Current: ",False,black)
#JValue=col.render("2345218 J",False,green)
nodetitle=col.render("Node nr.:    Tx:",False,black)
#gD.blit(prent,(900,500))
gD.blit(nodetitle,(10,20))

gD.blit(a,(10,70))
gD.blit(b,(10,110))
gD.blit(c,(10,150))
gD.blit(d,(10,190))
gD.blit(e,(10,230))
gD.blit(f,(10,270))
gD.blit(g,(10,310))
gD.blit(h,(10,350))

gD.blit(Title,(360,5))
#background rectangle

pygame.draw.rect(gD, peet, [300,60,1720,960])
for x in range(0,8):
        pygame.draw.rect(gD, w2, NodeRect[x])


gD.blit(a,node2)
gD.blit(b,node3)
gD.blit(c,node4)
gD.blit(d,node5)
gD.blit(e,node6)
gD.blit(f,node7)
gD.blit(g,node8)
gD.blit(h,node9)
gD.blit(Joules,node91)
gD.blit(Joules,node81)
gD.blit(Joules,node71)
gD.blit(Joules,node61)
gD.blit(Joules,node51)
gD.blit(Joules,node41)
```

```
        gD.blit(Joules,node31)
        gD.blit(Joules,node21)

        for x in range(8):
                gD.blit((col.render((str("%d"%powerDisplay[x])+" mA"),False,green)),NodeResultsDisplay[x])

        #gD.blit(JValue,node92)
        pygame.draw.line(gD, black,[150,60],[150,390],5)
        pygame.draw.line(gD, black,[0,60],[300,60],5)
        pygame.draw.line(gD, black,[0,390],[300,390],5)
        gD.blit(prent,(900,320))
#        pygame.draw.rect(gD, white, [760,116,400,200])
#pygame.display.update()

        #pygame.display.update()
        start = time.time()
        end = time.time()
        timeout = end - start

        while timeout < 5:
                for x in range(0,8):
                        if GPIO.input(NodeArray[x][0]) == 1:
                                TXcounter[x] = TXcounter[x]+1
                                print("Node number %d tx count %d"%(x+2,TXcounter[x]))
                                start2=time.time()
                                end2=time.time()
                                while GPIO.input(NodeArray[x][0])==1:
                                        if NodeArray[x][1]==0:
                                                curLOG[x][1]=adc.read_adc(NodeArray[x][2], gain=GAIN, data_rate=3300)
                                        else:
                                                curLOG[x][1]=adc2.read_adc(NodeArray[x][2], gain=GAIN, data_rate=3300)
                                        curLOG[x][0]=time.time()-beginNow
                                        curs.execute("INSERT INTO %s VALUES(%s,%s,1)"%(TableName[x],str(curLOG[x][1]),str(curLOG[x][0])))
                                        end2=time.time()
                                start=time.time()

                end = time.time()
                timeout=end-start
        db.commit()
        pygame.display.update()
db.close()
pygame.quit()
quit()
```

Appendix B MATLAB® code

```
clear all;
close all;
clc

N2 = fopen('node2.csv');
N3 = fopen('node3.csv');
N4 = fopen('node4.csv');
N5 = fopen('node5.csv');
N6 = fopen('node6.csv');
N7 = fopen('node7.csv');
N8 = fopen('node8.csv');
N9 = fopen('node9.csv');

N2Data = textscan(N2,'%f %f %d','HeaderLines',0,'Delimiter',',');
N3Data = textscan(N3,'%f %f %d','HeaderLines',0,'Delimiter',',');
N4Data = textscan(N4,'%f %f %d','HeaderLines',0,'Delimiter',',');
N5Data = textscan(N5,'%f %f %d','HeaderLines',0,'Delimiter',',');
N6Data = textscan(N6,'%f %f %d','HeaderLines',0,'Delimiter',',');
N7Data = textscan(N7,'%f %f %d','HeaderLines',0,'Delimiter',',');
N8Data = textscan(N8,'%f %f %d','HeaderLines',0,'Delimiter',',');
N9Data = textscan(N9,'%f %f %d','HeaderLines',0,'Delimiter',',');


 Node2y = N2Data{1,1}(:,1);
 Node3y = N3Data{1,1}(:,1);
 Node4y = N4Data{1,1}(:,1);
 Node5y = N5Data{1,1}(:,1);
 Node6y = N6Data{1,1}(:,1);
 Node7y = N7Data{1,1}(:,1);
 Node8y = N8Data{1,1}(:,1);
 Node9y = N9Data{1,1}(:,1);

 Node2x = N2Data {1,2}(:,1);
 Node3x = N3Data {1,2}(:,1);
 Node4x = N4Data {1,2}(:,1);
 Node5x = N5Data {1,2}(:,1);
 Node6x = N6Data {1,2}(:,1);
 Node7x = N7Data {1,2}(:,1);
 Node8x = N8Data {1,2}(:,1);
 Node9x = N9Data {1,2}(:,1);


f1 = figure(1);
cla; hold on; grid on;
```

```
plot(Node2x,Node2y,'r-');
title('Node 2');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f1,'Node2.png');

f2 = figure(2);
cla; hold on; grid on;
plot(Node3x,Node3y,'r-');
title('Node 3');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f2,'Node3.png');


f3 = figure(3);
cla; hold on; grid on;
plot(Node4x,Node4y,'r-');
title('Node 4');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f3,'Node4.png');

f4 = figure(4);
cla; hold on; grid on;
plot(Node5x,Node5y,'r-');
title('Node 5');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f4,'Node5.png');

f5 = figure(5);
cla; hold on; grid on;
plot(Node6x,Node6y,'r-');
title('Node 6');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f5,'Node6.png');

f6 = figure(6);
cla; hold on; grid on;
plot(Node7x,Node7y,'r-');
title('Node 7');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f6,'Node7.png');

f7 = figure(7);
cla; hold on; grid on;
plot(Node8x,Node8y,'r-');
```

```
title('Node 8');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f7,'Node8.png');

f8 = figure(8);
cla; hold on; grid on;
plot(Node9x,Node9y,'r-');
title('Node 9');
ylabel('Current Readings in Digital Format');
xlabel('Time in Seconds');
saveas(f8,'Node9.png');



TotalEnergy = zeros(8,1);
for ii = 2:length(Node2y)
    CurrentReading = Node2y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node2x(ii)-Node2x(ii-1);
    Energy = PowerReading*TimeInterval;
       TotalEnergy(1) = TotalEnergy(1)+Energy;
end

for ii = 2:length(Node3y)
    CurrentReading = Node3y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node3x(ii)-Node3x(ii-1);
    Energy = PowerReading*TimeInterval;
    TotalEnergy(2) = TotalEnergy(2)+Energy;
end
for ii = 2:length(Node4y)
    CurrentReading = Node4y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node4x(ii)-Node4x(ii-1);
    Energy = PowerReading*TimeInterval;
       TotalEnergy(3) = TotalEnergy(3)+Energy;
end
for ii = 2:length(Node5y)
    CurrentReading = Node5y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node5x(ii)-Node5x(ii-1);
    Energy = PowerReading*TimeInterval;
       TotalEnergy(4) = TotalEnergy(4)+Energy;
end
for ii = 2:length(Node6y)
    CurrentReading = Node6y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node6x(ii)-Node6x(ii-1);
    Energy = PowerReading*TimeInterval;
```

```
       TotalEnergy(5) = TotalEnergy(5)+Energy;
end
for ii = 2:length(Node7y)
    CurrentReading = Node7y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node7x(ii)-Node7x(ii-1);
    Energy = PowerReading*TimeInterval;
       TotalEnergy(6) = TotalEnergy(6)+Energy;
end
for ii = 2:length(Node8y)
    CurrentReading = Node8y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node8x(ii)-Node8x(ii-1);
    Energy = PowerReading*TimeInterval;
       TotalEnergy(7) = TotalEnergy(7)+Energy;
end
for ii = 2:length(Node9y)
    CurrentReading = Node9y(ii)*0.0244;
    PowerReading = CurrentReading*0.005;
    TimeInterval = Node9x(ii)-Node9x(ii-1);
    Energy = PowerReading*TimeInterval;
       TotalEnergy(8) = TotalEnergy(8)+Energy;
end
f9=figure(9);
x =
categorical({'Node2','Node3','Node4','Node5','Node6','Node7','Node8','N
ode9'});
str = pwd ;
idx = strfind(str,'\') ;
foldername = str(idx(end)+1:end) ;
figurename = strcat(foldername,'.png') ;
%titel(figurename);
bar(x,TotalEnergy);
title(['Energy Consumption Graph: ',foldername]);
xlabel('');
ylabel('Total Energy of Node in Joules');
saveas(f9,figurename);
DataSave = strcat(foldername,'.txt');
save(DataSave,'TotalEnergy','-ascii');
```

Appendix C Arduino Code GW

```
#include <RFM69.h>    //get it here: https://www.github.com/lowpowerlab/rfm69
#include <RFM69_ATC.h>//get it here: https://www.github.com/lowpowerlab/rfm69
#include <SPI.h>
#include <SPIFlash.h> //get it here: https://www.github.com/lowpowerlab/spiflash
//21 August 2018 insert Star topology

//*****************************************************************************
**********************
//************ IMPORTANT SETTINGS - YOU MUST CHANGE/CONFIGURE TO
FIT YOUR HARDWARE *************
//*****************************************************************************
**********************
#define NODEID        1    //must be unique for each node on same network (range up to
254, 255 is used for broadcast)
#define NETWORKID     100  //the same on all nodes that talk to each other (range up to
255)
#define GATEWAYID     1
//Match frequency to the hardware version of the radio on your Moteino (uncomment
one):
#define  RESETPIN     5
#define LED1    5
#define LED2  6
#define LED3  7
 #define TRIGGERPIN  4
#define FREQUENCY   RF69_868MHZ
#define ENCRYPTKEY    "crosspollination" //exactly the same 16 characters/bytes on all
nodes!
#define IS_RFM69HW
//#define ENABLE_ATC
#define SERIAL_BAUD   115200
int TRANSMITPERIOD = 1500; //transmit a packet to gateway so often (in ms)
char Package[30];
int PackageCount[10]={0,0,0,0,0,0,0,0,0,0};
int PackageWrongCount[10]={0,0,0,0,0,0,0,0,0,0};
bool DidReceive=false;
int count = 60;
int PackageSize;
int NodeTo=0;
int RecieveRouteByte = 0;
bool RecieveOn = false;
```

```
bool CustomRoute = false;
int NextNode = 0;
char Input[30];
int loopcount=0;
int receivecount=0;
RFM69 radio;
void setup() {
  Serial.begin(SERIAL_BAUD);
  radio.promiscuous(true);
  radio.initialize(FREQUENCY,NODEID,NETWORKID);
radio.setHighPower();
  radio.encrypt(ENCRYPTKEY);
  pinMode(TRIGGERPIN,OUTPUT);
  radio.setPowerLevel(31);
  while(Serial.available()==0);
  Serial.println("Starting Simulation");
}
void Receive()
{
if(radio.receiveDone())
   {
    PackageCount[radio.SENDERID]                                          =
PackageCount[radio.SENDERID]+radio.DATALEN;
    DidReceive = true;
    if (radio.TARGETID == NODEID)
    {
     for (byte i = 1; i < radio.DATALEN; i++)
     {
       Package[i] = radio.DATA[i];
     //  Serial.write(Package[i] + '0');
     }
    //  Serial.write(radio.DATA[0]+'0');
    //  Serial.println();
    }
    else
    {
    //  Serial.println("Messege Detected");
     for (byte i = 0; i < radio.DATALEN; i++)
      {
       radio.DATA[i];
     //   Serial.write(radio.DATA[i]+'0');
      }
```

```
PackageCount[radio.TARGETID]=PackageCount[radio.TARGETID]+radio.DATALEN;
   //  Serial.println();
    }
   }
}
void Transmit()
{
     NextNode = Input[0];
   for (int i = 0; i < PackageSize; i++)
    {
     Package[i] = Input[i + 1];
     //Serial.print(Package[i]+'0');
    }
   Package[PackageSize] = 0x11;
   for (int i = 1; i < ((PackageSize/2)+2); i++)
    {
     Package[PackageSize + i] = 0x00;
     //Serial.print(Package[PackageSize + i + 1]+'0');
    }
   PackageSize=PackageSize+((PackageSize/2)+1);
   for(int i = 0; i < PackageSize;i++)
{
    // Serial.write(Package[i]+'0');
    }
  radio.send(NextNode, Package, PackageSize, false);
  // Serial.println("B");
}
void CustomProtocol(){
   if (Serial.available() > 0)
  {
  delay(30);
   int i = 0;
   while(Serial.available())
    {
     delay(20);
     Input[i] = Serial.read();
    // Serial.print(Input[i]);
     Input[i] = Input[i] - '0';
     PackageSize = i;
     i++;
    }
   Serial.print(PackageSize, DEC);
```

```
    Serial.println();
    Transmit();
   }
  Receive();
}
void StringToChar(char *Test)
{
  int i = 0;
  while(Test[i]!='\0')
  {
    Input[i] = Test[i]-'0';
    i++;
  }
  PackageSize = i-1;
}
void TreeNetwork(int j)
{
    NodeTo = j;
    switch (NodeTo){
      case 2:
      StringToChar((char*)"201");
      Transmit();
      break;
      case 3:
          StringToChar((char*)"301");
      Transmit();
      break;
      case 4:
          StringToChar((char*)"401");
      Transmit();
      break;
      case 5:
          StringToChar((char*)"3050301");
      Transmit();
     break;
      case 6:
          StringToChar((char*)"4060401");
      Transmit();
      break;
      case 7:
          StringToChar((char*)"3070301");
      Transmit();
```

```
      break;
      case 8:
           StringToChar((char*)"4080401");
      Transmit();
      break;
      case 9:
           StringToChar((char*)"2090201");
      Transmit();
      break;
      default:
      break;
    }
}
void StarNetwork(int j)
{
    NodeTo = j;
    switch (NodeTo){
      case 2:
      StringToChar((char*)"201");
      Transmit();
      break;
      case 3:
           StringToChar((char*)"301");
      Transmit();
      break;
      case 4:
           StringToChar((char*)"401");
      Transmit();
      break;
      case 5:
           StringToChar((char*)"501");
      Transmit();
     break;
      case 6:
           StringToChar((char*)"601");
      Transmit();
      break;
      case 7:
           StringToChar((char*)"701");
      Transmit();
      break;
      case 8:
```

```
        StringToChar((char*)"801");
    Transmit();
    break;
    case 9:
        StringToChar((char*)"901");
    Transmit();
    break;
    default:
    break;
   }

}
void loop() {

 int Total = 0;

 while(loopcount<100)
 {
 for(int i = 2;i<10;i++)
 {
  // TreeNetwork(i);
  StarNetwork(i);
   int j = 0;
   while(j<600)
   {
    Receive();
    delay(1);
    j++;
   }
   if(!DidReceive)
   {
    PackageWrongCount[i]++;
   }
   else
   {
    DidReceive=false;
   }
 }
   loopcount++;
 }
 for(int i = 2;i<10;i++)
 {
```

```
    Serial.print(i,DEC);
    Serial.print(" - ");
    Serial.println(PackageCount[i],DEC);
    Total = Total + PackageCount[i];
  }
  Serial.println(Total,DEC);
  while(1);
//   if (Serial.available() > 0)
//  {
//
//      int NodeTo = Serial.read();
//     // Serial.print(Input[i]);
//      NodeTo = NodeTo - '0';
//      TreeNetwork(NodeTo);
//
//
//    }

 // CustomProtocol();
}
void LEDController(bool a, bool b, bool c)
{
  digitalWrite(LED1, a);
  digitalWrite(LED2, b);
  digitalWrite(LED3, c);
}
```

Appendix D Arduino Code Node

```
/*Beginning of Auto generated code by Atmel studio */

#include <Arduino.h>


/*End of auto generated code by Atmel studio */


// Sample RFM69 sender/node sketch, with ACK and optional encryption, and Automatic
Transmission Control

// Sends periodic messages of increasing length to gateway (id=1)

// It also looks for an on-board FLASH chip, if present

// RFM69 library and sample code by Felix Rusu - http://LowPowerLab.com/contact

// Copyright Felix Rusu (2015)


#include <RFM69.h>    //get it here: https://www.github.com/lowpowerlab/rfm69

#include <RFM69_ATC.h>//get it here: https://www.github.com/lowpowerlab/rfm69

#include <SPI.h>

#include <SPIFlash.h> //get it here: https://www.github.com/lowpowerlab/spiflash

//Beginning of Auto generated function prototypes by Atmel Studio

//End of Auto generated function prototypes by Atmel Studio
```

```
//*************************************************************************
**********************
//************ IMPORTANT SETTINGS - YOU MUST CHANGE/CONFIGURE TO
FIT YOUR HARDWARE *************
//*************************************************************************
**********************
#define NODEID        2//9    //must be unique for each node on same network (range up
to 254, 255 is used for broadcast)
#define NETWORKID     100  //the same on all nodes that talk to each other (range up to
255)
#define GATEWAYID     1
//Match frequency to the hardware version of the radio on your Moteino (uncomment
one):
#define  RESETPIN     5
#define LED1    5
#define LED2  6
#define LED3  7
 #define TRIGGERPIN  9
#define FREQUENCY   RF69_868MHZ
```

```
#define ENCRYPTKEY    "crosspollination" //exactly the same 16 characters/bytes on all
nodes!
#define IS_RFM69HW
//#define ENABLE_ATC    //comment out this line to disable AUTO TRANSMISSION
CONTROL
//*****************************************************************************
**********************
#define SERIAL_BAUD   115200
int TRANSMITPERIOD = 1500; //transmit a packet to gateway so often (in ms)
char Package[30];
int count = 60;
int RecieveRouteByte = 0;
bool RecieveOn = false;
int NextNode = 0;
RFM69 radio;
void setup() {
  Serial.begin(SERIAL_BAUD);
  radio.promiscuous(true);
  radio.initialize(FREQUENCY,NODEID,NETWORKID);
radio.setHighPower();
  radio.encrypt(ENCRYPTKEY);
```

```
 pinMode(TRIGGERPIN,OUTPUT);
 radio.setPowerLevel(31);
  for (int i = 5; i < 8; i++)
  {
   pinMode(i, OUTPUT);
   digitalWrite(i, LOW);
  }
}
void loop() {
 //process any serial input
 if (radio.receiveDone())
 {
  int RXSize = 0;
  for (byte i = 0; i < radio.DATALEN; i++)
  {
   Package[i] = radio.DATA[i];
   RXSize = i;
   if (radio.DATA[i] == 0xAA)
   {
    RecieveRouteByte = i;
   }
```

```
}
char TXPackage[30];
if (radio.TARGETID == NODEID)
{
 int i = 0;
 while (i < (RXSize-1))
 {
  TXPackage[i] = Package[i + 2];
  i++;
 }
 int LogLED = (RecieveRouteByte*(1.5))-2;
 TXPackage[LogLED] = Package[0];
 NextNode = Package[1];
 switch (Package[0]) {
 case 0:
  LEDController(0, 0, 0);
  break;
 case 1:
  LEDController(0, 0, 1);
  break;
 case 2:
```

```
  LEDController(0, 1, 0);
  break;
case 3:
  LEDController(0, 1, 1);
  break;
case 4:
  LEDController(1, 0, 0);
  break;
case 5:
  LEDController(1, 0, 1);
  break;
case 6:
  LEDController(1, 1, 0);
  break;
case 7:
  LEDController(1, 1, 1);
  break;
default:
  break;
}
digitalWrite(TRIGGERPIN,HIGH);
```

```
      radio.send(NextNode, TXPackage, RXSize - 1, false);
      digitalWrite(TRIGGERPIN,LOW);
      //  Serial.print((char)radio.DATA[i]);
      // Serial.print("   [RX_RSSI:");Serial.print(radio.RSSI);Serial.print("]");
    }
 }
}
void LEDController(bool a, bool b, bool c)
{
  digitalWrite(LED1, a);
  digitalWrite(LED2, b);
  digitalWrite(LED3, c);
}
```

Appendix E Article for Africon 2017

IEEE Africon 2017 Proceedings

Wireless Node Energy Monitor using common Development Platforms

Using a Raspberry Pi to monitor energy consumption by a WSN

Petrus van Staden and Ben Kotze
Department of Electrical, Electronic and Computer Engineering
Central University of Technology, Free State
Bloemfontein, South Africa
vanstadenp@cut.ac.za, bkotze@cut.ac.za

***Abstract*— this article presents a simple method of acquiring data in order to experiment on the energy consumption of wireless sensor networks. This data allows further work in designing an affordable WSN testbed. Development platforms used in this article was the Raspberry Pi 3 connected to an ADC and the myRIO from National Instruments utilizing a FPGA processor.**

Keywords—WSN topology, Energy, DAQ, Testbeds

I. INTRODUCTION

This paper proposes a method for building a testbed that is measuring energy usage on a Wireless Sensor Network (WSN) with popular development platforms. Experimenting on the topology of a WSN the power usage is mostly calculated in theory or advanced testbeds can be used for hire as discussed in a survey done on WSN testbeds [1]. The theoretical calculations will work on calculating the energy used on all WSN configurations, but the results will mostly give a comparative scale factor that in conclusion can be compared to other WSN topologies. Due to many variables, the prediction of battery life using only theory will give an unreliable result. By measuring the power usage of each node during an experiment on WSN's the battery life time can be predicted. An experiment is done using a high-speed data acquiring device and comparing it to a more affordable Linux based computer (Raspberry Pi with Raspbian Jessie installed) in the process of developing a WSN testbed that focuses on power consumption of a WSN.

Relevant theory for this article presents already developed testbeds, methods of acquiring current measurements at high sampling rates, logging current measurements, converting analog to digital data and determining power usage. For the method of this article acquiring data using the myRIO from National Instruments for a very high precision data measurement at a very high data rate provided a perfect benchmark to test the Raspberry Pi data acquisition (DAQ) system. The performance of the Raspberry Pi DAQ will allow a conclusion that will state if a Raspberry Pi DAQ can be used to log power consumption of multiple wireless nodes allowing the deployment of a WSN testbed that measures the power consumption of a WSN. Fig. 1 shows a basic block diagram of the proposed solution to a conclusion made by this article.

Fig. 1 Proposed DAQ for multiple device energy monitoring

II. THEORY OF MAKING A DATA ACQUISITION SYSTEM AND PREVIOUS WORK

A. Related Works on the Development of Simple Testbeds

Motelab: a wireless sensor network testbed [2], known to be one of the first WSN testbeds [1] it consists of a set permanently deployed sensor network nodes. The system is web-based and can be controlled with a central server. The central server handles data gathering and application deployment. The basis of this system is deployed at Harvard University. Motelab uses MicaZ node attached to an Ethernet interface platform operating at 433MHz.

Article [3] actually used 5 Raspberry Pi's as a part of a wireless mesh network. This is very similar as the proposed outcome of this article, but with the main difference that the conclusion resulted in 1 Raspberry Pi monitoring 8 wireless sensor nodes.

B. Related Works for acquiring data from analog devices

M. Ambrož used a Raspberry Pi to acquire data from a human powered vehicle to determine physical power requirements. The proposed system logged data from an accelerometer using analog pins and saving the data to a file [4].

FPGA-based wireless sensor nodes for vibration monitoring system and fault diagnosis proposed a solution for synchronous data acquisition of systems that vibrate [5]. The tests by Bengherbia et al. was performed on a rotating machine test bench with an unbalance fault.

Fig. 2 ADS1115 ADC breakout board by Adafruit® [12]

High speed FPGA-based data acquisition system achieved sampling rates of up to 80 mega samples per second [6]. The system is utilized for measuring high speed signal changes at the input side of a specific system. Khedkar ed al. discusses FPGA design architecture programed by VHDL and using extra methods for storing data in a way that no samples will go missing.

C. Methods for acquiring energy readings

1) Hall Effect sensor

Physicist Edwin H Hall discovered that current passing through a perpendicular magnetic field will be moved with a force proportional to the product of the intensity of the magnetic field and the velocity of the charge carrier [7]. A force created through electric and magnetic fields cause charged particles such as electrons [8]

2) Analog-to-Digital converters

Analog-to-Digital converters (ADCs) translate analog values to a digital format, enabling information to be processed by a processor. One of the most common methods of ADC is Delta-sigma $\Delta\sum$ modulation [6]. An analog signal applied to the input of the converter needs to be relatively slow so the converter can sample it multiple times, a technique known as oversampling. The sampling rate is hundreds of times faster than the digital results at the output ports according to Texas Instruments Incorporated [7]. For the proposed solution, an ADS1115 (Fig. 2) was selected due to the high availability and already preprogrammed examples in Python. At 16-bit operation the sample rate increases up to 868 samples per second (SPS), but when using the 12-bit operation mode it can reach a maximum of 3300 SPS.

D. Hardware devices used for experiments

1) ArduRF1's

The ArduRF1's shown in Fig. 3 is based on the Arduino framework. It is a PCB containing an ATMEGA328 microcontroller. Preprogrammed with the Arduino Uno boot loader the board can be programmed via a USB cable. Onboard is a FTDI chip that handles USB to serial communication.

Fig. 3 Wireless nodes ArduRF1's [13]

2) Logging Data

Logging data can be done utilizing different methods. This article focuses on capturing data and saving it so that analysis can be made with the documented data. Noteworthy is the possibility of obtaining a too low resolution when saving data, buffers of the processor being used may fill, resulting in inconclusive logged data. A class 10 SD card can be written to at 10 Mega Bytes per second [9].

3) myRIO

The myRIO, Fig. 4, device produced by National Instruments features multiple I/O's and includes analog output and input pins. The myRIO interfaces onboard LEDs, a push button, an onboard accelerometer, a Xilinx FPGA and a dual-core ARM Cortex-A9 processor.

Fig. 4 myRIO device used for high speed data acquisition [10]

It includes Wi-Fi support. The myRIO can be programed with LabVIEW™ or C.

4) Raspberry Pi 3 Model B

Fig. 5 Raspberry Pi 3 model B [11]

The Raspberry Pi 3 model B is equipped with a 1.2GHz 64bit quad-core ARMv8 CPU, 802.11n Wireless LAN, Bluetooth 4.1 other than its predecessor, the Raspberry Pi 2. Same as the Pi 2 it also has 1GB RAM and a Micro SD card holder [11]. The Raspberry Pi has a very big supporting community, allowing easier product/project development and design.

III. METHODOLOGY

First objective was programming the myRIO using LabVIEW. The aim of this program was to take current reading and voltage readings. After taking the readings the system calculated the power and then the total energy used. After knowing the precise energy usage, the sample rate was set lower constantly until a benchmark result was captured.

Second objective was to do the same with the Raspberry Pi. Firstly, only using the same method and circuitry used for the myRIO and then secondly, using an added current sensor. Both systems were triggered using the chip select pin from the ArduRF1's. This triggered the processors when the wireless node was starting transmission.

A. Basic Node Circuit

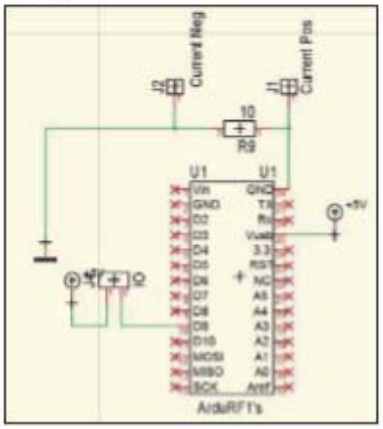

Fig. 6 Basic circuit for monitoring Energy usage

The ArduRF1's has an Atmega328P microcontroller preloaded with an Arduino UNO bootloader enabling programming via the Arduino IDE and USB cable. Fig. 6 presents the basic schematic, designed for monitoring energy usage. The R9 resistor is for measuring the current. Fig. 8 is a flow chart of the nodes firmware with the main objective to transmit a test package at full power once every two seconds. Before a transmission, the trigger pin is set high, letting the DAQ system know when to start acquiring analog readings. The trigger pin changes to a low level to initiate a stop. The circuit is designed with an extra header pin connected to the 5V power supply to monitor the voltage at any given moment. Fig. 7 is a printed circuit with an onboard 5V/1A power supply.

Fig. 7 Power monitor circuit for the ArduRF1's

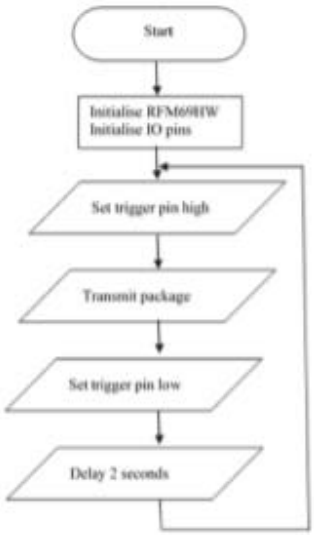

Fig. 8 Flow chart for the wireless node

B. myRIO setup

1) FPGA Program

Fig. 9 is the LabVIEW code running on the FPGA. The sampling rate is set manually and the DIO connected to the circuit (Fig. 6) initiates a start sampling sequence. The FPGA ADC is inherently limited to a sample rate of 10us/sample.

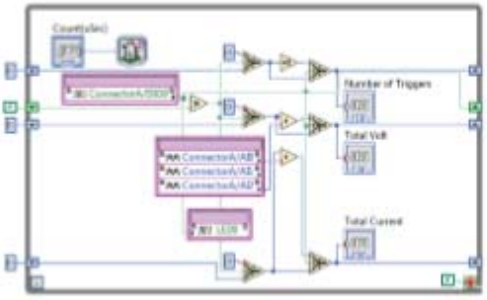

Fig. 9 Code running on the FPGA of the myRIO

In Fig. 10 the myRIO is connected to the energy monitor. Implementing variables of unsigned integers, manipulated equations utilizing float variables, granting less data manipulation.

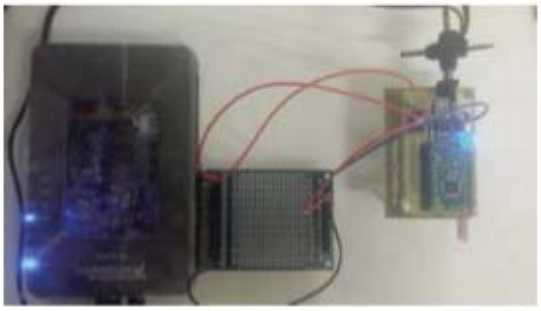

Fig. 10 myRIO connected to the wireless node

The value passed from the FPGA to the Real-Time Processor (RTP) is presented by the following equation:

$$F_{OUT} = \sum_{i=0}^{n} (A(i)_0 - A(i)_1)A(i)_2 \quad (1)$$

$F_{out} = FPGA\ output\ value$

$A_x = Analog\ pin\ being\ read$

$n = Number\ of\ samples\ during\ triggered\ interval$

2) Real Time Processor

As stated by Equation 1 the RTP receives a value from the FPGA. The value is modified by the RTP to produce proper output readings:

$$E = Pt \quad (2)$$

$E = energy\ in\ Joules\ (J)$

$P = power\ in\ Watts\ (W)$

$t = time\ per\ each\ interval\ in\ seconds\ (s)$

$$P = \frac{1}{n}\sum_{i=0}^{n} V(i)I(i) \quad (3)$$

$n = number\ of\ samples\ taken$

$$V(i) = DA(i) \quad (4)$$

$D = constant\ \ step\ voltage\ 1.221mV$

$A(i) = Analog\ reading\ in\ digital\ format$

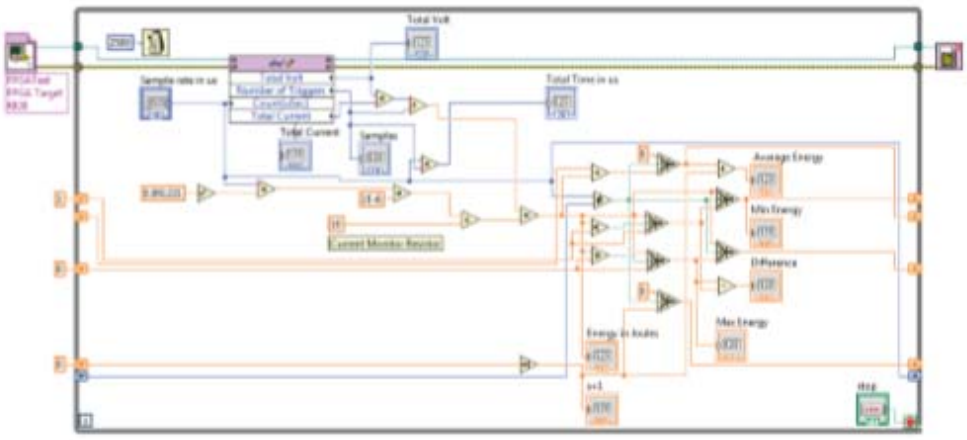

Fig. 11 LabVIEW code running on the RTP

$$I(i) = \frac{DA(i)_0 - DA(i)_1}{R} \quad (5)$$

$R = resistance\ of\ 10\ ohm$

$$E = \frac{1}{n}\sum_{i=0}^{n} DA(i)_2 \left(\frac{DA(i)_0 - DA(i)_1}{R}\right) tn \quad (6)$$

$$E = \frac{tD^2}{R} F_{OUT} \quad (7)$$

By implementing equations (2), (3), (4), (5) and (6) a final result is achieved represented by (7). Fig. 11 shows the implementation of (7) in LabVIEW.

The RTP takes a reading from the FPGA each 2.1 seconds that grants a reading of a variable that does not change during the duration of the reading cycle. The results are very small and for better accuracy an average was calculated on each transmission of a test package. Different transmission times was tested obtaining the results using this method. The ideal outcome was to compare the amount of energy used on a transmission kept constant after making the sampling rate changes.(4(5(4(3(2(5(3(2

C. *Raspberry Pi DAQ*

1) Python setup:

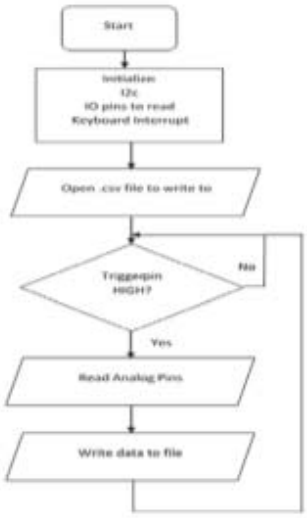

Fig. 12 Python code for DAQ on the Raspberry Pi

Fig. 12 is the program flow chart to program the Raspberry Pi to interface with the ADS1115. A few different tests were done to experiment on the difference between fast and slow sampling. Fig. 13 indicates the interface attached to the Raspberry Pi, enabling the Raspberry Pi to be a DAQ device.

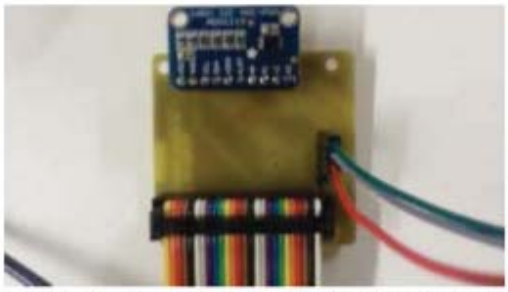

Fig. 13 The ADS1115 breakout board connected to the Raspberry Pi and Energy Monitor circuit

IV. RESULTS

A. *Energy Reading's myRIO:*

Table 1 Results for reading energy with myRIO

Sampling Time (µs)	Samples taken:	Joule Readings (mJ)
100	132	0.266
200	66	1.338
500	27	7.19
1000	13	7.18
1500	9	7.18
2000	6	7.22
4000	3	7.20
8500	1-2	7.25
9500	1	6.78

Table 1 is logged values where all the conditions were kept the same for each reading. The *samples taken* value is the number of samples taken during a transmission. The transmission time was +/- 13ms allowing 132 readings at a sample rate of 100 µs.

B. *Energy Reading's Raspberry Pi DAQ:*

Table 2 is a sample of data logged during a transmission. There were 5 samples taking during transmission. The energy column in Table 2 was calculated using the values in the columns to the left with a constant 10 Ohm resistor.

Table 2 Data logged with the Raspberry Pi during a transmission

Current Reading(mA):	Voltage Reading(V):	Duration(ms):	Energy (mJ):
16.01	5	5.6	0.451
127.24	5	5.45	3.429
75.17	5	5.33	2.006
10.76	5	5.318	0.288
Total:			6.221

Table 3 recorded the energy used for each transmission individually using the Raspberry Pi DAQ system.

Table 3 Total energy of each consecutive transmission

Transmission number:	Energy (mJ):
1	5.25
2	5.99
3	6.22
4	6.14
5	6.27
6	6.22
7	6.22
8	6.16

C. Further Development

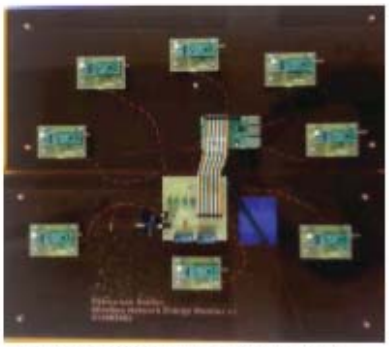

Fig. 14 WSN testbed to record power consumption

Fig. 14 shows a picture of a developed testbed using the tested results. This way the WSN can do its programmed application while the Raspberry Pi is recording current measurements.

V. Conclusion

The readings obtained by the myRIO and the readings obtained by the ADS1115 module differ with a constant value. The myRIO DAQ system starts to stabilize once a sampling rate of 500µs was implemented. This result was obtained since the variables of the FPGA used were unsigned integers of 16 bits. The expected result was, that by an increase in the sample rate time using the myRIO, the energy reading was to remain constant. Looking at Table 1 this trend is visible between 500µs and 4000µs sample rates.

Comparing results between the Raspberry Pi DAQ and the myRIO, the Raspberry Pi DAQ measured ~1mJ less in each transmission. This could be contributed to a small variance not detected in the test being done resulting in a 14% lower power usage in the wireless node transmission. The conclusion is that using the ADS115 with a Raspberry Pi it is possible to measure up to 8 WSN devices allowing for the development of a cheaper testbed. Provided that the voltage power source is kept constant 5V and each analog pin is connected to the Current + header pin and the Current – header pin equates to an absolute value of 0V.

References

[1] L. P. Steyn and G. P. Hancke, "A survey of Wireless Sensor Network testbeds," in *IEEE Africon '11*, 2011, pp. 1–6.

[2] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: a wireless sensor network testbed," in *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pp. 483–488.

[3] T. Oda, M. Yamada, R. Obukata, L. Barolli, I. Woungang, and M. Takizawa, "Experimental Results of a Raspberry Pi Based Wireless Mesh Network Testbed Considering TCP and LoS Scenario," in *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, 2016, pp. 175–179.

[4] M. Ambrož, "Raspberry Pi as a low-cost data acquisition system for human powered vehicles," *Measurement*, vol. 100, pp. 7–18, 2017.

[5] B. Bengherbia, M. Ould Zmirli, A. Toubal, and A. Guessoum, "FPGA-based wireless sensor nodes for vibration monitoring system and fault diagnosis," *Measurement*, vol. 101, pp. 81–92, Apr. 2017.

[6] A. A. Khedkar and R. H. Khade, "High speed FPGA-based data acquisition system," *Microprocess. Microsyst.*, vol. 49, pp. 87–94, Mar. 2017.

[7] R. S. Popovic, *Hall effect devices*. CRC Press, 2003.

[8] M. Espitia, "MEMS Hall Effect Sensor," *J. Microelectron. Eng. Conf.*, vol. 20, no. 1, 2014.

[9] "Speed Class - SD Association." [Online]. Available: https://www.sdcard.org/developers/overview/speed_class/. [Accessed: 21-Feb-2017].

[10] "myRIO - Student Embedded Device - National Instruments." [Online]. Available: http://www.ni.com/en-za/shop/select/myrio-student-embedded-device. [Accessed: 22-Feb-2017].

[11] "Raspberry Pi 3 Model B - Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 22-Feb-2017].

[12] "ADS1115 16-Bit ADC - 4 Channel with Programmable Gain Amplifier ID: 1085 - $14.95 : Adafruit Industries, Unique & fun DIY electronics and kits." [Online]. Available: https://www.adafruit.com/product/1085. [Accessed: 04-May-2017].

[13] "ArduRF1s Arduino with RF-link and Battery Power from ddebeer on Tindie." [Online]. Available: https://www.tindie.com/products/ddebeer/ardurf1s-arduino-with-rf-link-and-battery-power/. [Accessed: 22-Feb-2017].

Appendix F RFM69HW Spec sheet

HOPERF ELECTRONIC

RFM69HW

RFM69HW ISM TRANSCEIVER MODULE V1.3

GENERAL DESCRIPTION

The RFM69HW is a transceiver module capable of operation over a wide frequency range, including the 315,433,868 and 915MHz license-free ISM (Industry Scientific and Medical) frequency bands. All major RF communication parameters are programmable and most of them can be dynamically set. The RFM69HW offers the unique advantage of programmable narrow-band and wide- band communication modes. The RFM69HW is optimized for low power consumption while offering high RF output power and channelized operation. Compliance ETSI and FCC regulations.

In order to better use RFM69HW modules, this specification also involves a large number of the parameters and functions of its core chip RF69H's,including those IC pins which are not leaded out. All of these can help customers gain a better understanding of the performance of RFM69HW modules, and enhance the application skills.

RFM69HW

KEY PRODUCT FEATURES

- +20 dBm - 100 mW Power Output Capability
- High Sensitivity: down to -120 dBm at 1.2 kbps
- High Selectivity: 16-tap FIR Channel Filter
- Bullet-proof front end: IIP3 = -18 dBm, IIP2 = +35 dBm,80 dB Blocking Immunity, no Image Frequency response
- Low current: Rx = 16 mA, 100nA register retention
- Programmable Pout: -18 to +20 dBm in 1dB steps
- Constant RF performance over voltage range of module
- FSK Bit rates up to 300 kb/s
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK and OOK modulations
- Built-in Bit Synchronizer performing Clock Recovery
- Incoming Sync Word Recognition
- 115 dB+ Dynamic Range RSSI
- Automatic RF Sense with ultra-fast AFC
- Packet engine with CRC-16, AES-128, 66-byte FIFO Built-in temperature sensor
- Module Size:19.7X16mm

APPLICATIONS

- Automated Meter Reading
- Wireless Sensor Networks
- Home and Building Automation
- Wireless Alarm and Security Systems
- Industrial Monitoring and Control
- Wireless M-BUS