Long Distance, Low Data Rate Packet Transmission at 868 MHz: Protocol Design and Implementation

by

A.S.M. Samiul Saki Chowdhury

Supervisor
Frank Yong Li

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Masters of Science in Information and Communication Technology

Department of Information and Communication Technology
Faculty of Engineering and Science
University of Agder
Norway

Grimstad
May 15, 2016

Abstract

Data transmission over wireless networks can be achieved in various ways. Some of these technologies are feasible in many real life scenarios, but there are many consequences we have to consider in this field, such as packet loss, quality degradation of radio signal over distance incrementation, ambient noise, transmission delay and etc. For the low data rate packet transmission, to adapt and to address IoT is one of the most discussed topics nowadays, especially in terms of transmission protocol design and the implementation of that design in real life scenarios. Transmitting packets in low data rate over long distance efficiently is becoming an important issue in this fast-paced modern world. Although various work has been done individually on low data rate packet transmission and protocol to achieve longer distance. Our objective is to combine both options in the same protocol design with diverse form of services.

In this project, we design an embedded solution and implement our own transmission schemes based on Raspberry Pi micro-controller to send low date rate packets via radio modules that operating on 868 MHz band to achieve long distance. We embed and implement the design with low cost and low power components and send real-time services such as GPS co-ordinates, temperature, image and etc. over wireless network. Then we develop our design to relay the transmission from the source node to the destination node by using a forwarding node in between them, which helps us to achieve further distance. These transmission schemes have the ability to respond to each other with acknowledgements assuring the successful transactions between the nodes and for the request of retransmissions. In addition, the schemes are developed further to a new scheme where the retransmission will occur with an adaptive data rate at the end of the transmission process to ensure the successful transfer of a large file. At the end, we collect and examine the data on the sender and receiver ends and calculate parameters such as packet loss rate, transmission delay, etc., by running the tests in different scenarios and optimising the experimental results. The results show that with these transmission schemes the packets can be transmitted to long distance in low data rate and with a small amount of packet loss rate.

Preface

Entrance enthusiasm and revelation have always emerged as a fundamental part in the accomplishment of any task. This thesis is the result of the IKT590 Master's thesis project, which is corresponding to 30 ECTS points, at the Department of Information and Communication Technology (ICT), Faculty of Engineering and Science, University of Agder (UiA) in Grimstad, Norway. The work has started from January 4, 2016 and ended on May 16, 2016.

I would like to thank my thesis supervisor, Professor Frank Yong Li, for his encouragement, ideas, guidance and support from the initial to the final level throughout the entire project. With his supervision I have learned a great deal about mobile and wireless communication technology and the technical implementations of scientific concepts.

I am also thankful to my study co-ordinator Morten Goodwin and study consultant Sissel Marie Andreassen for all their support on my academic career in UiA. I would like to thank our head of the department Folke Haugland for providing me with the necessary equipment and tools required for this masters thesis. I am grateful to my wife Kari-Anette Nilsen for her contribution on helping me collecting experimental data for this master thesis and her devoted support.

A.S.M. Samiul Saki Chowdhury

Grimstad
May 15, 2016

Contents

List of Figures

List of Tables

List of Abbreviations

ARM	Advanced RISC Machine
CEPT	European Conference of Postal and Telecommunications Administrations
CRC	Cyclic Redundancy Check
DECT	Digital Enhanced Cordless Telecommunications
DSSS	Direct-Sequence Spread Spectrum
EOF	End of File
ERP	Effective Radiated Power
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
FHSS	Frequency-Hopping Spread Spectrum
FTDI	Future Technology Devices International
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
ITU	International Telecommunication Union
LAN	Local Area Network
LQI	Link Quality Indication
PLR	Packet Loss Rate
RF	Radio Frequency
RFID	Radio-Frequency Identification
RPi	Raspberry Pi
RSSI	Received Signal Strength Indication
SAW	Surface Acoustic Wave
SRD	Short Range Device
U-NII	Unlicensed National Information Infrastructure
WDCT	Worldwide Digital Cordless Telecommunications
WSN	Wireless Sensor Network
WuR	Wake-Up Radio

Chapter 1

Introduction

In the traditional wireless networks, the main focus of the researches are pointed at sense and report capabilities of components to acquire information of physical phenomenon and environmental parameters. This information is such as temperature, sound and pressure , etc. In present days wireless communication technologies, this information are highly demanded for various reasons to fill in the gaps in communication protocols. Wireless devices are nowadays are being used for services such as emergency responses, rescue missions, shipping, mining, monitoring systems and even for daily life activities. In daily life situations, wireless communications have achieved an indicative attention due to the development of wireless technologies. Since not only the components of wireless networks are being upgraded but also, the research has been done on different types of services that will be transmitted over these networks are also being researched on. The limited amount of frequency allocation creates a unique field of research work where researchers uses their scientific methods of finding suitable solutions which can properly utilise the allocated frequencies to the devices, to be exact the radio modules or sensors. The frequencies most sensor networks works on are the ISM bands. This chapter gives an overview of ISM frequency band as it is one of the most commonly researched field in present day information and communication technology.

1.1 Background and Motivation

Before the discussion of the ISM bands we like give a brief overview of how the RF frequencies works. RF is the electromagnetic wave frequencies that operates in between the range of 3 kHz to 300 GHz. Frequency in the RF refers to the rate of oscillation. These frequencies include the communication frequencies and radar signals. It is usually the electrical oscillations rather than mechanical. We are focusing on the electrical RF in this project. Some of the special properties of RF current (which is different than direct current or alternating current of lower frequencies) are as follows:

- As the basis of the radio technology, the energy of an RF current can radiate off a conductor into space as radio waves.
- RF current tends to flow along electrical conductors surfaces instead of penetrate deeply into them (skin effect).
- RF current can create a conductive path through air by ionising it.

- RF can flow through paths that contain insulating material, like the dielectric insulator of a capacitor.

In radio communication fields, to receive the radio signals an antenna must be used. The antenna will pick up lots of other signals in the environment. These usually handled by including a radio tuner into desired frequencies. This radio tuner consists of via a resonator, a simple circuit with a capacitor and an inductor to form a tuned circuit. This resonator used to amplify oscillations within a specific frequency band and reduce the oscillations at other frequencies outside the band. In the software defined radios, isolation of the specific RF is done by oversampling and selecting the desired frequency. The wavelength of the RF in radio-communications are an important parameter to choose over the distance. In addition, other parameters need to be considered are receiver quality,type,size,height of the antenna, transmitter power, mode of transmission, surrounding noise and other signals which can interfere in while transmitting or receiving. Greater ranges can be achieved by using waves such as ground waves and skywaves other than using line-of-sight propagation. Our project will focus specifically on the ISM bands which lies within 300 MHz to 3 GHz of frequencies.

Short range device

SRDs have low capabilities of causing harmful interference to other radio equipments. These devices are the recommendations of ECC which is represents the RF transmitters used in telecommunications for information transmissions. These devices are typically low power transmitter which has the limitations of 25 to 100 mW of ERP or less. This ensure the use of such products in few 100 meters range and thus makes it licenses free from its users. SRD applications are on wireless devices with short range such as RFID application, radio-controlled models, security, alarms, wireless microphone, radars on vehicles, fire, traffic signs and signals (for control), remote for garage doors, motion detection and hundreds of application similar to that. The allocation of usage provided by European Commissions through CEPT and ETSI restricts the parameter of their use to avoid interference with other radios. In Europe, 863 to 870 MHz band has been allocated for license-free operation using FHSS, DSSS, or analog modulation with either a transmission duty cycle of 0.1%, 1% or 10% depending on the band, or listen before talk with adaptive frequency agility. The following table (Table 3.1)represents briefly the duty cycle and ERP allocated to the SRD860 frequency bands [1].

Unrestricted voice communications are allowed in the 869.7-870.0 MHz band with channel spacing of 25 kHz from Dec 2011. The maximum power output if this is 5 mW ERP. It is a common misunderstanding that ISM equipment includes telecommunication devices such as WiFi and cordless phones. This misconception stems from the fact that many unlicensed communication devices are permitted to operate in ISM bands; however, a device is not ISM just because it operates in an ISM band. Unlicensed communication devices legally operate under separate rules from ISM.

Benefits of 868 MHz over 2.4 or 5 GHz:

The 868 MHz is less congested than other frequencies, for example 2.4 or 5 GHz where signal is competing among all complicated designs, protocol stacks, controls and all sort of applications which can lead to control signal being wiped out or distorted. Sub-GHz proprietary protocols are often the best choice for low power, low data rate wireless networks due to their extended range, ease of implementation, and small code size. Limited duty cycle is a feature in 868 MHz which

helps controlling of the interferences. This means that no device that using this band frequency is allowed to transmit for more than a few milliseconds every second. This ensure that fewer devices will transmit at same time. According to many manufactures of 868 MHz transmitter chips, the frequency has at least more interference immunity than other existing frequencies. As the usage of frequencies getting more regular, the 2.4 or 5 GHz is getting more crowded since they are mostly used for WiFi and there are more user for the bandwidth than of 868 MHz. It is sometimes advised that for a long distance radio link, using a radio operating at 868 MHz is better than at 2.4 GHz . This assumption is based on the fact that, for any given distance, the attenuation suffered by the radio waves (free space loss) increases with the operating frequency. The free space loss is not the only factor that affects the link performance. The attenuation over distance favours 868 MHz over 2.4 or 5 GHz because the free space loss at those higher bands are around 8.5 dB larger than at 868 MHz. It combines the penetration of normal MHz systems and the reliability of most new 2.4 GHz systems. In terms of atmospheric gases (such as oxygen and water vapour), fog and rain can add to the free space loss attenuation and their effect are worst at 2.4 and 5 GHz links. Also tree with leaves that have dimensions near the wavelength of 2.4 GHz will cause higher attenuation at 2.4 GHz. In addition, a single Wi-Fi based network can have a network size of only up to 2007 nodes whereas SRD bands based networks can have over 65,000 nodes in one such network. These disadvantages of higher frequency bands are some of the main reasons we chose to work on 868 MHz SRD bands for our masters this project.

Based on these properties, we try to develop a prototype using 868 MHz enabled radio modules to send data from transmitter prototype to receiver prototype over long distance, which will then be received by the receiver and use the data to project the informations that was sent from the source (transmitter) node. Then we tried to implement a two-hop transmission protocol to forward the received data from source node to a further destination hope and measured the performance of the running systems. Finally, a solution for retransmission is implemented where this procedure will occur after all the packets of a large file have been transferred by the sender in the first phrase of transmission and calculating which packets are dropped on receiver side during the second phrase of retransmission. The dropped packets are requested to the sender by the receiver nodes and this entire retransmission process will continue in an adaptive data rate.

Our motivation is to design and implement a working transmission scheme to achieve long distance and low power transmission over the low data rate wireless network. This system design can influence development on the devices used on the real-life scenarios such as emergency rescue services, GPS positioning or even underwater research fields. This low power transmission can establish a network based on IoT and support our existing wireless network infrastructure. Our goal is to design a protocol to foster IoT for future researches. The IoT is the network for physical object-devices, vehicles, buildings and other items embedded with electronics softwares, sensors and network connectivity to enable these objects to collect and exchange data between each other or to the central base stations. Using IoT we can let the objects surrounding us to be controlled and collect useful data from them across existing wireless or wired network infrastructure, that creates the ability for more direct integration of the physical world into computer systems. This will help us to improve the efficiency, accuracy as well as benefit on economic point of view. “Things” in IoT refers to a wide variety of devices such as biochip on animals, electric clams, heart monitoring implants, RFID, motion detectors, home automated sensors, environmental monitoring sensors, even a light bulb with heat recognising sensor are in this “Things” class which are mix of all softwares, hardwares, data and services. The devices in these classes are designed to collect and send data over medium by using very exiting technologies and protocols. The point of IoT is to connect all the devices together in the same network so that anyone can control or overview the information gathered by their devices from anywhere in the world in despite where the device

is located and how they are connected to the network. This opens up the opportunities of vast field of advance data transferring protocol and services in terms of connectivity and data sharing. The goal is to connect every device with every other single devices around the world for better, smarter, more informative, faster and better world. We try to give a thorough insight of our system design which might inspire future researchers to achieve an improved working solution for low data rate transmission research. The proposed transmission schemes can impact on the exiting research field and can help on the development of the WSNs.

1.2 Problem Statement

Data transmission has emerged as a key issue for designing next generation wireless networks. Having to periodically send data over a network, achieving longer distance, avoiding degradation of data signals while transmitting over long range, reducing packet loss ratio are some of the main interesting and focuses in wireless communication research work. On the other hand, most protocol depended on networks variables, such as transferring mediums (radio modules) have their own limitations due to their hardware configurations, range, data rate, transmission power , etc. Packet transmission over low data rate and low power consuming devices is a promising approach that addresses these issues with such networks, in general, wireless networks, by considering various parameters which we categorise in the following parts:

- **How can we design a protocol which can be acclimated in our research to achieve our goal?**
  We need to design a protocol that can accommodate various types of data transmission while maintain the data flow flawlessly between nodes to ensure the handshake between them. This handshakes can be used to maintain the constantability of the connection between nodes and aid the reduction of packet loss.
- **What type of services we need to utilise via our designed wireless network?**
  Among various types of services, we have to choose the types of services that have the most usefulness, low in data rate and at the same time does not need huge amount of data to transfer in the transmission. For our design, we have to consider types of services which can only use a few amount of bytes due to our limitation of the data bytes allocated to transfer via radio module.
- **How to evaluate the performance of this embedded design?**
  We have to evaluate the performance of our system design by putting them to the tests in various test scenarios and measure the parameters extracted from the systems. The experimental results need to be compared and discussed to asses the performance of our protocol design.

1.3 Objectives and Research Approaches

Design for data traffic transmission requires efficient modelling of the system to achieve the best possible data transferred over-the-air are developed by considering the parameters such as longer distance, better transmission power, lower packet loss, energy efficiency, stable connectivity and so on. It is not always easy to maintain the balance between these parameters when designing and implementing an embedded system as they are not always well-balanced for every kind of situation. We briefly discuss some of the main objectives and research approaches done in this project:

- In need of a longer range of connectivity, packet loss can get higher. Stable connectivity can sometimes affect energy efficiency which is also considered to be an obstacle in the research method. Even though when the system can be capable of sending the data over such network, a real-time transmission can get a bit tricky since most of the low powered radio modules can face some signal degradation due to atmosphere changes as we discussed briefly in Sec. 2.1. We are designing our transmission schemes in such a way that it can perform in various condition of scenarios according to the chosen types of services.
- The stable connectivity issue reduce data packet loss on the sender/receiver side can become the main issue while designing a system. Our project goal is to maintain the stable connectivity between the nodes during transmission process to address this issue.
- Types of services are also distinguished focus in our project as different types of services need different design approach to transfer and retrieve over wireless network correctly.
- Our system design goal is to achieve a long distance of data transmission by adapting to multi-hop network concept where a forwarding node will relay the transmitted data from the source node to the destination node. The flawlessness of this design can give us suitable performance of our system which will reflect on our experimental results extracted from the various test scenarios.

We study several literatures related to these topics and assess as we further progress in our project. We came across several theoretical techniques proposed by many researchers and found many mechanisms have been done which contemplate the data transmission over the distance wirelessly. Many researchers proposed their own algorithms and protocols to implement their concept on the scenarios based on real life, and some are based on simulations. We design an embedded system and implement in low energy consuming components to send various type of services of data such as image, GPS co-ordinates , etc., over 868 MHz enabled radio module, which have a better stability and a very low rate of packet loss. Our design has a good on-sight range of connectivity which ensures good quality of data transfer for a distinguishable long distance /range over the network. In addition, the different experimental results from our experimental scenarios are collected and compared the results in-between them.

1.4 Report Outline

Rest of our report is organised as follows:

- In Chapter 2, we discuss the enabling technologies that is implemented in this project. We also discuss the tools we are using in our particular system design and review some of the related work that has been done on 868 MHz enabled radio modules.
- In Chapter 3, we discuss our system design principal, hardware and software integration and show comparisons between the probable radio devices that could be used in this research work.
- In Chapter 4, we discuss our implementation techniques of our transmission schemes and corresponding types of services. We also give an overview of designed frame structures in our system in this chapter.

- In Chapter 5, we show the results extracted from our experimental test scenarios and evaluation of the performance of our proposed design.
- And finally, in Chapter 6, we draw our conclusion and discuss the future work.

Chapter 2

Enabling Technologies and Related Work

A designer of wireless applications has multiple choices of standards and protocols, ranging from the simple to the amazingly complex design. Many wireless devices operate in 868 to 870 MHz band. This line of low power SAW components and hybrid transmitter and receivers are designed to conform to ETSI / CEPT regulations applying to low power SRDs. The components are designed to operate on somewhere within the 868 to 870 MHz are as follows: TX resonators - R02164A, R02156A; LO resonators - RO2165A,RO2166A; receive filters - RF1336B, RF1319B; Transmitters - HX3004B, HX3007B and receivers - RX3004B, RX3007B. This components helps to reduce the interference problem that has been faced on 433.92 MHz band (which is usually caused due to polarised ISM equipments, licensed services and high powered amateurs). These new bands supports various application such as automotive security, home security, SRD radio communications links and mainly data collection. There is SAW resonator components which are developed for discrete transmitters which are for , etc.-10.7 MHz offset local oscillators and front-end filters for receivers. Hybrid transmitters and ASH receivers products are adapted to support the band requirements within three sub-bands. All devices meet the frequency stability and application requirements for each band. In the world of health and safety feature, monitoring an individual's physical condition or to monitor an entire platform, these components which operate on this bands delivers a significant amount of service.

2.1 Data Transmission in ISM Band

The ISM radio bands are portions of the radio spectrum reserved internationally for the use of RF energy for industrial, scientific and medical purposes, other than telecommunications. Some of the best application of these bands are microwave oven, RF process heating, medical diathermy machines and etc. The reason these devices allocated with such specific frequency band (ISM band) as these powerful machines can interfere with the other frequencies by creating electro-magnetic interferences. These certain devices are only to use these limited bands of frequencies. All the equipments used specifically for communications need to have the ability to tolerate the interferences generated by all the ISM applications, where users have no regulatory protection from ISM device operations. ISM bands have multiple allocations, recently it is used for short range, low power communications systems such as cordless phones, bluetooth devices, near field communications devices and wireless computer networks. Although these low power emitters are not considered as ISM. The ISM bands are defined by the ITU Radio Regulations (article 5) in footnotes 5.138, 5.150, and 5.280 of the Radio Regulations. Depending on the national radio reg-

ulations the uses of these bands designated for these sections may differ for individual countries. According to the principal of licensing, unlicensed devices need to be tolerant of the interference from other devices, hence any unlicensed operations are usually permitted to use these bands [3]. Article 5 of the ITU radio regulations (edition 2012) provides the allocation if radio frequencies. The majority of service allocations determined in article were incorporated in national Table of Frequency Allocations and Utilisations. This is mainly to improve the harmonisation in the spectrum utilisation.

ISM uses

The most commonly used ISM device is microwave oven which operate on 2.45 GHz, moreover there are many devices that exist which uses these ISM bands. Some of the other uses are industrial plastic welding processes, shortwave and microwave diathermy machines with ISM tolerance to be used as muscle relaxation for medical purposes, microwave ablation is also an ISM application which being used to treat solid tutor through the use of RF heating. Long distance wireless power systems have been proposed and experimented with which can be used to send power to remote locations using high power transmitters and rectennas.

Non-ISM uses

ISM bands have also been shared with (non-ISM) license-free error-tolerant communications application such as WSNs in the 915 MHz and 2.450 GHz bands. Its also being used in wireless LANs and cordless phones in 915 MHz, 2.450 GHz and 5.800 GHz. Unlicensed low power users are able to work in these bands without creating any interference for other ISM bands users since the unlicensed devices are need to be tolerant to the ISM emissions. All of this equipment does not add a radio receiver in the ISM band, for example, microwave ovens does not have any receiver installed on it. To be noted that 915 MHz cannot be used in those countries outside region 2, except those countries which allows GSM-900 band for cellphones. Wireless LANs and cordless phones requires approval to use in a country which is usually based on each county's regulations. DECT phones are allocated to use spectrum outside ISM bands in Europe and North America as well as ultra-sideband LANs require more spectrum than the ISM bands and it uses IEEE 802.15.4a that designed to use the such spectrums. These low power communication devices are sometimes referred to as ISM bands even though these bands are outside the ISM bands. Several brands of radio control equipment use the 2.4 GHz band range for low power remote control of toys, from gas powered cars to miniature aircraft so as WDCT is a technology that uses the 2.4 GHz radio spectrum [2]. Our thesis will focus specifically on the UHF bands which lies within 300 MHz to 3 GHz of frequencies.

UHF

UHF is known as the decimetre band because the wavelengths range in this band is from one meter to one decimetre. Radio waves propagates by the line-of-sight in UHF band. The waves can be blocked by large hills and buildings even though the transmission though the walls of buildings can be high enough for good indoor reception. Cell-phone, satellite communications, broadcasting of television, GPS connections, personal radio such as Wi-Fi, Bluetooth, cordless phones, walkie-talkies and other such applications are used with the help of UHF band. According to IEEE standard letter designations for RF Bands, UHF radar band as frequencies in the range of 300 MHz to 1 GHz. Some of the characteristics of UHF also the reason behind our motivation

for choosing this band to work on. To explain this we need to know what causes the effect of attenuation degradation of the frequencies. The point to point transmission and reception of TV and radio signals is affected by many variables such as, atmospheric behaviour, solar wind, time of the day, physical obstruction and etc. These variables can affect transmission and reception signal quality, because all of the radio waves can partially absorb by the atmospheric moisture. This absorption can attenuates the strength of the radio signals of UHF over long distances since degradation increases with frequency. The ionosphere (earth's atmosphere layer), is filled with charged particles (usually electrically charged atoms and molecules) that can reflect some of the radio waves. This property of the ionosphere helps ti propagate the lower frequency HF signals around the world as the trapped waves are bouncing around in the upper layers of the ionosphere until they are refracted down at another point of the earth. This another point can be the receiving side of the communication and often referred to skywave transmission. In the case of UHF radio signals are not carried along the ionosphere but can be reflected off of those charged particles down at another point on Earth which reaches farther than the typical line-of-sight transmission distances [4].

Advantage of UHF

The main advantage of UHF transmission is the short wavelength that is generated by HF. Although it has a very limited broadcast range UHF is mostly used in two way radio communication systems and cordless phones. Transmission generated in two way radios do not travel far enough which reduce the interference with the other local transmissions. IEEE 802.11 wireless LANs (Wi-Fi) are widely used within UHF frequencies. So as the widely used GSM and UMTS cellular networks falls into UHF cellular frequencies, Usually a repeater is used to propagate UHF signals to address the shorter line-of-sight issues occur and greater distances than this is required. Most of the bandwidth of the UHF are used for mobile device communication in present days. It is also used for public safety area and military purposes. Many personal radio services using the allocated frequencies in the UHF bands, but the usage of this frequencies differ from country to country.

2.1.1 Brief discussion on 2.4 and 5 GHz

The 2.4 GHz ISM band provides 13 overlapping channels spread equally over the frequencies. In Japan an additional channel is used with the center frequency 2.484 GHz. This leaves available only three non-overlapping channels in the 2.4 GHz band. This channels have to be used very efficiently in order to avoid interferences between these wireless LAN connected devices. Also the efficient installation requires careful frequency planning. The installation costs can become higher than the actual wireless equipment installed due to expensive installation solutions such as leakage cables. On the other hand the 5 GHz band is divided into sub-bands called UNII bands and are usually named as U-NII-1, U-NII-2, U-NII-2e and U-NII-3 (U-NII-3 is not available worldwide). In total there is 23 non-overlapping channels where four of these have location limitations. Most common wireless LAN solutions uses U-NII-1 band in 5 GHz (5.18 to 5.24 GHz) with frequency channels 36 to 48. There are some suppliers of the equipment who have extended the range to include the U-NII-2 or U-NII-2e band (5.26 to 5.70 GHz) with frequency channels 52 to 140. Wireless LAN IEEE 802.11b/g/n is already well-established with its huge installed base and a wide range of products made available. The 2.4GHz band provides the advantage of operating in a worldwide available ISM band. While using the same output power the achieved range is certainly better on 2.4GHz compared to radios using the higher frequency 5GHz band. The basic problem with 5GHz ISM band is not available for use worldwide entirely. Availability of components and products are still somewhat limited compared to the 2.4GHz band. The best

feature of the 5GHz band is the availability of 23 non-overlapping channels; 20 more channels than what is available in the 2.4GHz band. Since there is no other wireless technology that compete for the radio space, the 23 available non-overlapping channels can provide a possibility for easier planning of an interference-free and stable wireless communication. In 5GHz band good number of available channels provides for increased density, which means that more wireless devices can be connected in the same radio environment [5].

2.1.2 Brief discussion on 868 MHz

The 800 MHz frequency band is a portion of the electromagnetic spectrum, or frequency band, that encompasses 790–862 MHz. 868 MHz band was allocated by the ITU to Broadcasting as the primary user in Region 1 and was used for analogue television broadcasting before changing to digital terrestrial television in many countries. As such it is also referred to as **digital dividend** spectrum. In Europe and to some extent elsewhere, the band corresponds to UHF channel 61–69. In most territories the band was also used by Services Ancillary to Broadcasting (SAB) or Services Ancillary to Programme Making (SAP), both often now referred to as PMSE (Programme Making and Special Events) in the form of professional wireless microphones, radio talkback systems and wireless monitor systems. The European Parliament approved May 2010, the change of use of the 800 MHz band making it available for purposes other than broadcasting (television) – e.g., mobile broadband. The frequency band between 862 MHz to 870 MHz, or a subset of this band, e.g., 863 MHz to 865 MHz, is sometimes referred to as 'Channel 70'. 862 MHz to 870 MHz is itself divided into many smaller sub-bands used in many territories by vast numbers of a wide range of different low power radio devices, often on a licence exempt basis. For example 863 MHz to 865 MHz is used in many European countries by cordless audio devices such as some cordless headphones, assistive listening devices and some wireless microphones. Any low power radio devices operating in the range 862 MHz to 870 MHz are most likely to be affected by interference from the portable devices using the 800 MHz LTE system since they will be transmitting in the upper part of the 790 MHz to 862 MHz band, i.e., the part of the band which is closest to the frequencies used by Low Power Radio Devices. System that use the 868 MHz-band (868–870 MHz), for example, thermostats, fire systems, burglar systems, and DIN-transceivers may have difficulty communicating when there is a strong 800 MHz broadband transmitter nearby [6].

We discuss more about 868 MHz in Chapter 3 in details.

2.2 Overview of the Tools

Based on the features enabled by SRD (UHF) bands and inspiration behind idea of IoT, we researched and utilised the following components in our protocol design. Some of the main features of the components or tools, which helped us to choose these products, are as follows:

2.2.1 Connect2Pi

Connect2Pi USB dongle is a product of LPRS easyRadio technology which can operate in 868 MHz in UK and Europe and 915 MHz in US [7]. This radio device operates as a bridge between RPis (Pi2Pi) or can be between a RPi and a PC or any other device that supports USB serial communications. This device provides considerably greater range and less power consumption

than WiFi or Bluetooth dongles which are mainly operating in the overcrowded 2.4 GHz bands. Connect2Pi USB module gives the user ability to change it frequency, bandwidth, output power and data rate according to user's preferences. The product can communicate between each other or any other RF devices (working on the same frequency) with interference. It is configured with the ERA900-TRS radio module which helps it to free of requiring any RF protocol software. This radio module uses the 'Sense and Control' technique as it's working protocol.

Figure 2.1: ERA-Connect2Pi USB Dongle [7]

Some of the features of the device are as follows:

- It supports Bi-directional Link.
- It has very low current consumption which can be powered by RPi.
- The integral SMA antenna connector allows the use of extension for optimal antenna position.
- Transit and receive LEDs are installed to diagnostics purposes.
- Host device can send and receive (half duplex) up to 180 bytes of data per packet that will be seamlessly delivered and presented to other hosts within the range. These 180 bytes are ideal for sense and control applications as this product is made to target such applications. No need for any complicated bit balancing. Easy data in data out technique makes Connect2Pi a suitable product for our project.
- It has a built in temperature sensor which can be utilised by the host program.
- RF parameters are highly configurable (using provided software) that helps to fine tune the optimal performance. For Linux and Windows machines FTDI FT232 USB IC drivers can be used.
- Some of the product specifications are as follows:
  - Supply voltage: 5 V
  - Supply current: 25 mA
  - USB host data rate: 2.4 to 115.2 Kbps (19.2 Kbps default)

  - Frequency: 869.75 MHz (configurable to 868 MHz for Europe) and 915 MHz (configurable for US)
  - RF output power: -1 to +7 dBm (+5 dBm default)
  - RF data rate: 115.2 kbit/s
  - Receive sensitivity: Max -117 dBm (-107 dBm default)
  - Range: Up to 2.2 Km
  - Operating Temperature : Max +85 °C to Min -40 °C
  - Antenna: 1/4 Wave whip antenna
  - Size: 80 x 22 x 10 mm (including connectors, excluding antenna)
  - Width: 22 mm
  - Weight: 11 g (without antenna)
  - USB connector: USB type A plug

- The modes of transmission include an enhanced easyRadio protocol with 16 bit encryption and anti-cross talk software. In addition raw data modes where users can use self-coding system which can be set in both FSK and ASK modulation.
- Digital RSSI can be used to called via a simple command.
- Temporary channel or power level selection allows the user to scan other channels on the fly without storing the settings in internal EEPROM.
- Temperature compensation plus crystal controlled synthesiser for frequency accuracy less than ±1 kHz over full temperature range. Temperature sensor that can be used by the user.

The Connect2Pi USB device comes with ERA900-TRS transceiver that combines high performance very low power transceiver, a micro controller and voltage regulator. The serial data input and output operate on the 19200 baud rate. It has bandwidth down to 12.5 KHz. It has two handshake lines providing optional flow control to and from the host. The transmitted bytes of data are internally buffered before transmitting to in an efficient over-the-air format. Any easyRadio transceiver can hear the transmission while in range and can place the data in received buffer to download it by the host for processing and interpretation. The radio can transmit and receive but cannot do it simultaneously, hence it is half-duplex. However it can be given an appearance of full-duplex scenario by increasing the internal buffers which allow user to upload while downloading is in progress. Connect2Pi radio module has the capability to take radio commands from the user or specified application by the user and change its settings to it accordingly. Such commands need to be entered in a form of such as ER_CMD#T3 (will show the radio's firmware version number), ER_CMD#T7 (will give the current ambient temperature), ER_CMD#B2 (to change the bandwidth settings of the radio module to 5 kHz - 9600 bps), ER_CMD#P0∼9 (will adjust the RF power output of the radio), ER_CMD#T9 (will show the current RSSI) followed by a 'ACK' command in sequence 'A"C"K'. Each channel frequency is calculate relative to the channel number, the channel width and the start frequency of the channel. Three commands control the settings of each of these parameters: ER_CMD#Cn (channel command, where n is the channel number), ER_CMD#Bn (bandwidth command, where n is the channel spacing) and ER_CMD#bn (band plan command, where n is the start frequency of the band plan being used). The centre frequency pf each channel is calculated using the formula:

$$Centre\ Frequency\ (f) = b + cs + \tfrac{s}{2}\ [7]$$

where b : band plan start frequency, c : channel spacing and s : channel spacing. The capability of using the radio commands to alter settings for radio while transmitting makes this radio an ideal candidate for our data transmission project. easyRadio Advanced modules have the added versatility of being used without the proprietary protocols from easyRadio and yet still being used as a multi-channel, multi-bandwidth module. This allows the user to set precise frequencies to replace other raw data devices on exact frequencies. Both FSK (FM) and ASK (AM) modes are supported, and with the integration of a configuration mode, AM/FM modes, power levels, channels and bandwidth settings, can be changed on the fly with a very efficient command structure [7].

2.2.2 Raspberry Pi

The Raspberry Pi is a series of credit card–sized single-board computers developed in **Pencoed**, Wales by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools and developing countries. All Raspberry Pi include processors with an ARMv6-compatible core or newer ARMv7 cores and have included on a chip with VideoCore IV GPU and have at 256 megabytes of RAM, except later models (models B and B+) that have 512 MB. The system has Secure Digital (SD) (models A and B) or MicroSD (models A+ and B+) sockets for boot media and persistent storage. In 2014, the Raspberry Pi Foundation launched the Compute Module, which packages a BCM2835 with 512 MB RAM and an eMMC flash chip into a module for use as a part of embedded systems. Raspberry Pi newer versions (e.g., model B)includes a Broadcom BCM2836 system on chip (SoC), with a quad-core ARM Cortex-A7 CPU and a VideoCore IV dual-core GPU; 1 GB of RAM.

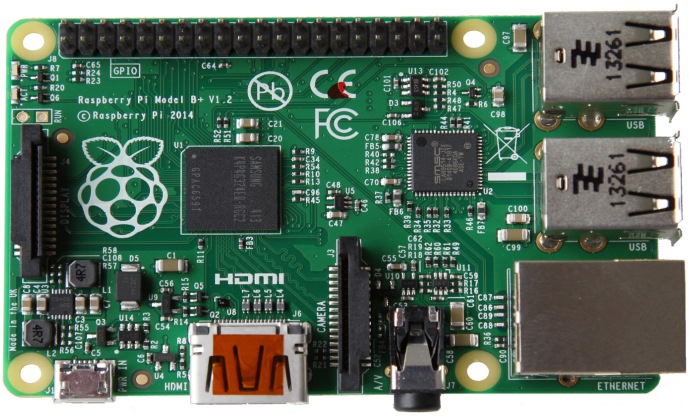


Figure 2.2: Raspberry Pi 2 Model B+ [8]

Some of the main feature we should know about Raspberry Pi are as follows [8] :

- **Hardware**: The newer versions of Raspberry Pi (model B and B+) comes with a Ethernet port and contains up to 4 USB ports.
- **Processor**: The Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes an 700 MHz ARM1176JZF-S processor, VideoCore IV GPU and RAM. It has a Level 1 cache of 16 KB and a Level 2 cache of 128 KB. The Level 2 cache is used primarily by the GPU. B+ model comes with Broadcom VideoCore IV @ 250 MHz, OpenGL

ES 2.0 (24 GFLOPS), MPEG-2 and VC-1 (with license) and 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder.

- **Power Consumption**: By replacing linear regulators with switching the power consumption has been reduced by between 0.5 W and 1 W, about 600 mA (3.0 W). Its power source is 5 V via MicroUSB or GPIO header.
- **Overclocking**: Most Raspberry Pi chips could be overclocked to 800 MHz and some even higher to 1000 MHz. The second generation can be similarly overclocked, in extreme cases, even to 1500 MHz. In some cases the Pi automatically shuts the overclocking down in case the chip reaches 85 °C (185 °F), but it is possible to overrule automatic over voltage and overclocking settings. Newer versions of the firmware contain the option to choose between five overclock ("turbo") presets that when turned on try to get the most performance out of the SoC without impairing the lifetime of the Pi.
- **RAM**: On the older beta model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release model B (and model A), three different splits are possible. The default split was 192 MB (RAM for CPU). For the new model B with 512 MB RAM initially there were new standard memory split files released(arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). The newer models of Raspberry Pi comes with up to 1 GB of RAM.
- **Low level peripherals**: B+ models got 17x GPIO plus the same specific functions, and hardware at the top (HAT) ID bus.
- **Baudrate**: The default baud rate of raspberry pi is 115200.
- **Real-time clock**: The Raspberry Pi does not come with a real-time clock, which means it cannot keep track of the time of day while it is not powered on. As alternatives, a program running on the Pi can get the time from a network time server or user input at boot time.

2.2.3 Python

As the programming tool we use **Python** programming language as our software platform to design our protocol. It is a widely used general-purpose, interpreted, high-level, dynamic programming language. Its design philosophy gives priority to code reliability. The syntax of Python allows programmers to express concept in fewer lines of code than would be possible in languages such as Java or C++. This language helps to get a neat and clear programs in both large and small scales. It supports multiple programming paradigms which includes object-oriented, functional procedural styles. This language can be packaged as standalone executable program most operating systems. This allows this programming language to be running on any operating environments without any help from Python interpreters [9]. Python is a free and open-source software, runs by a non-profit software foundation, which features a dynamic type system and automatic memory management and has come with late and comprehensive standard library. In our project, we use Python v2.7.3 to build our design and code implementations.

2.3 Related Work

A lot of interesting work has been done on wireless radio communication in particularly in 868 MHz frequency band as it is one of most flexible license free band authorised to use in UK and

Europe (for US its 915 MHz). This creates the opportunity in the research field of IoT development as well as data transmission protocol design over low power, low band and low data rate wireless communications between mobile radio modules or systems. Here we like to focus on some of the theoretical and practical work, which are done on wireless sensor network research particularly on 868 MHz SRD band. As mentioned, any device, system, person referred as things can be remotely controlled using application running on smart device (e.g., smart phones, remote controller and etc.). It is not possible to find one common protocol stack to be finalised for IoT things. Each IoT things need to have assigned IP addresses. First of all each things or devices must have a physical layer connection to medium with wire or wirelessly. Secondly, each should have layer to interface with backhaul network as per technology where the devices will be developed for. The basic IP address are the third option which need to be focused on with which the IoT protocol stack is completed on. IoT networks can be designed with the HUB connecting all the things together or to the things directly. In case of controlling the things a controller can be included to the network. The following are the brief description of the research work we find most interesting in our early research:

2.3.1 Research work based on WuR system

The researchers on the paper [10] have bought our attention to the duty-cycled Medium Access Control protocols, which always improve the energy efficiency of wireless networks. But most of the these protocols suffer from large amount of overhearing and idle listening. Idle listening occurs when a node listen to the wireless medium during no-transmission periods in order for next incoming payload to the node itself. Also the overhearing occurs when a node listens to communications meant for another node. In an energy-constrained wireless networks those are crucial factors in aspect of energy usage of the sensor nodes. To address this problem many researches have proposed WuR systems which drastically reduce the energy consumptions. According to the many solutions it is addressed by completely switching off the micro controller unit and main radio transceiver until a secondary, extremely low-power receiver is triggered by a particular wireless transmission, hence the name WuR. But most of these proposed solutions are based on theoretical platforms and simulators. The authors of paper [10] proposed a real-life simulation based on OMNET++ simulator which is a promising WuR hardware platform. In WuR, a simple receiver is installed in addition which is called WuR receiver which is attached via the micro controller's GPIO pins, configured at it lowest power mode by default. This hardware is designed to listen only to wake-up calls meant for that particular node sent by another wake-up transmitter of the remote nodes. When the WuR receiver confirms that the remote node is trying to communicate with its attached micro-controller, then the controller unit activates the next level of data transmission. This makes a big change on the energy consumption ratio of he sensor nodes by reducing the idle-listening, overhearing and latency issues and allows for considerable energy savings, higher PDR and less complicated software implementations. This factor of reduction is by a factor of 1000, which is a crucial factor for wireless nodes with small battery capacities. However, such low current consumption comes at the cost of worse receiver sensitivity, which in turn implies that the wake-up transmitter has to transmit at a higher power. The authors first intend to provide a lean and modular OMNET++ simulation model for WuR systems, which represents a real hardware platform and proposed several realistic application scenarios and MAC protocols. They simulated the model in real life test bench and gives the performance of their WuR flavours that gives the future developers to adapt to them if needed. This paper [10] was one of first to provide such realistic and performance evaluation with comparative analysis of with other existing MAC protocols to the best of researchers' knowledge. The paper demonstrates how the use of our WuR platform presents numerous benefits in several areas, from energy efficiency and latency to packet delivery ratio and applicability. This research work helps us understanding the principle of MAC protocol design and we acclimated to this idea behind this WuR technique

to design our system's protocol.

2.3.2 Theoretical performance review based related work

In paper [11], the researchers evaluated the performance of radios operating on the 868 MHz frequency band. They discussed the lossy links characteristics of the WSNs. 2.4 GHz band has a disadvantage over the issue that it is crowded spectrum and expose to a great amount of external interferences such WiFi and, Bluetooth , etc., 868 MHz is noise free so it makes a perfect candidate to the researchers' indoor testbed on a large scale (about 108 nodes). They performed an experiments on the characteristics of low power lossy lines of a 868 MHz radio and analyse the testbed experiments to identify fundamental link properties, compare these to properties of 802.15.4 links and discusses about it in their research paper [11]. They observed that the 868 MHz is noise free but multi path and human activity result in time varying lossy and asymmetric links. Some of their research observations helped us gain good knowledge on 868 MHz characteristics as our protocol design is based on the this frequency.

Another theoretical work [12] we review where the authors of this paper used the 868 MHz band for geolocation service. They used a statistical frequencial repartition technique to choose one frequency that have the emitted signal, more precisely inside each channel of 200 kHz which is centred on f_c, emitted signal will have a frequency that is far by f_i from f_c. The implied that the bandwidth of each emitted signal is around 100 Hz which represents the number of symbol bits emitted. With their algorithm the authors used a cross-correlation method for estimating TOA and transform to temporal domain the correlated signal. They found the maximum integer of correlated signal to find the integer part of the delay. To find the fractional delay by using parabolic interpolation between integers and over-sampling between the integers. They concluded with their performance evaluation of this frequency band that the narrow band signals gives good performances of geolocation with much better SNR in these conditions with a very low cost. Their application can be implemented on long range tracking of the frail, fall detection, access control and remote alarm.

In paper [13], the authors presented their proposed plan for the deployment of 868 MHz RF modules in WSN scenarios. Their approach covers the site-general ITU propagation model calculation compared with the real field measurement made with open-source hardwares platform based on Arduino UNO and SeedStudio RFBee communication modules. Their design goal is to predict the best positions for wireless sensor stations in WSN and also to avoid collision the 2.4 GHz frequency range. They used two type modelling: site-specific, which requires detailed information the building layout, furniture and transceiver location and the site-general modelling where the position of the sensors and furniture is unknown. Site-general models gives a gross statistical prediction of path loss for link design and node deployment. They utilised the popular mITU path loss model and log-distance model with the usage of adjustment. There are two important calculations are needed for the ITU path loss modelling:

$$P_{rx} = P_{tx} + G_{tx} - L_{tx} - Lpl - L_m + G_{rx} - L_{rx}$$

where, P_{rx} : received power (dBm), P_{tx} : transmitter output power (dBm), G_{tx} : transmitter antenna gain (dBi), L_{tx} : transmitter losses (dB), L_{pl} : path loss (dB), L_m : miscellaneous losses (dB), G_{rx} : receiver antenna fain (dB) and L_{rx} : receiver losses (dB). The formula can be reduced to as follows due to very low $G_{tx}, G_{rx}, L_{tx}, L_{rx}$ and L_m:

$$L_{pl} = 20log_{10}(f) + Nlog_{10}(d) + L_f(n) - 28dB$$

where, N : the distance power loss coefficient, f L: frequency in MHz, d : distance in meters (d >1 m), $L_f(n)$: floor penetration loss factor and n: number of floors between transmitter and receiver. The measurements are made in institutional building on only one floor, so the $L_f(n)$ is omitted from calculations. The proposed model in paper [13] shows to be accurate for the exponent N value of 42. Although the model was not suitable for distances less than 8 m and for the locations where the obstacles with big impact on reducing the signal strength are present between transmitter and receiver. This model design give us insight of system design model on indoor environment particularly for WSNs.

2.3.3 ZigBee based data transmission

Among all the previous research work, we find very interesting is the research done on GPS-based positioning and identification system for ECO-boats, which was a student bachelor project done at 2013 in Department of Information and Communication Technology in University of Agder. The team proposed a prototype consists of a GPS-Module, a Radio-module and a Micro-controller. They made a system consists of three prototypes. Each prototype was mounted on three ECO-boats. Each prototype were designed to send their GPS information to each other and based on that they integrated a nautical mapping service where all the ECO-boats will be visible. The boats get its own GPS positions from its GPS module and broadcasted to other boats with the help of the radio module. Radio-Module operates on a licence-free 868 MHz band with which one can get the range of 1 km. Upon receiving the GPS information from other boats over its own radio and then it shows graphically on a map. Based on this every boat sees each other positions on the map can avoid possible future collisions. The researchers of that project tried to show that the boats can get its own GPS positions even though they do not have any line of sight due to some obstacles(e.g. mountains). The team used the ZigBee modules as their radio module which achieved a range of 1 km successful transfer of data while there was 2.9 km of stable connectivity between each radio [14]. They proposed their technique to avoid the collision of the boats by using PID of each other and calculates which way the boat will be reached in certain time according to its own individual speed. Although they used the radio module (XBee 868LP) to send the GPS coordination that it achieved from GPS-module and just sent that small amount of bytes (59 bytes in 200 kHz to be exact) according to ZigBee specification. Zigbee devices have MAC protocol overhead of 25 bytes and PHY protocol overhead of 6 bytes. In total Zigbee has 31 bytes of protocol overhead which is similar to Bluetooth devices according to IEEE 802.15.4 standard. Their protocol was mainly based on simple GPS information sending and receiving and using the information to store in map logging software. They didn't implement any active protocol in their project design. This project caught our attention which helped us to understand the basic technique of sending small data over wireless networks.

2.3.4 RISC micro-controller based WSN

Another interesting project that was done, where the researchers designed a system using RISC micro-controller checks the performance of both 868 MHz and 2.4 GHz frequency bands and compared their performance to the results from another system operating in the 433 MHz frequency band using the same scenarios. They designed and developed their WSN with the goal of network flexibility a, energy efficiency and versatility. In paper [15], the developers addressed the issue of the energy efficiency of commercial, industrial and residential buildings which creates the

demand for measuring the temperature and humidity in various places. Significant number of wireless sensors are used to implement such network. This sensors are designed to operate on mostly adaptable frequency bands such as 868 MHz and 2.4 GHz. The problem of all the network co-existing at the same time creates the issue of spread-spectrum transmission which provides narrowband frequency immunity. Another problem is propagation inside the buildings, which depends on many factors. The authors built a WSN with real-life applications. The network has tree architecture with a fixed routing table where the root node is connected to a host system (PC) via USB connection and the network structure allows the forwarding of packets sent from distant leaves via one or multi hops using routers.They used 8 bit RISC micro-controller which is a part of ZigBit module. The module consists of an RF transceiver which can be programmed to either 868 MHz or 2.4 GHz, since it got a dual chip antenna. The module has a balanced RF output (for both 868 and 2.4) designed in such a way that both 868 MHz and 2.4 GHz versions share the same PCB footprint. It has temperature and humidity sensor. The link budget of a two-point connection is measured as the following equation:

$$\frac{P_r}{P_t}[dB] = P_r[dBm] - P_t[dBm]$$

where P_r : RSSI and P_t : Transmitted Power. The authors chose RSSI over LQI (LQI calculation based on signal strength and symbol correlation in each frame). Although LQI gives more accurate and detailed information as it estimates packet error rate but RSSI has pure physical meaning and ideal for testing purposes. Their setup was measured using hundred bursts, each containing 100 packets. Their test was done in outdoor to have practical performance with expected theoretical values and they chose a sight with transmission range of 885 m at 2.4 GHz and 1500 m at 868 MHz (even though according to manufactures it was 4 Km and 6 Km respectively). They also tested their system indoor by testing the transmission via different types of walls such as 5 cm wide dry wall, 10 cm wide brick wall and 15 cm wide reinforced concrete wall. Their result in indoor test shows that at 868 MHz the wall causes the same RF power loss (about 3.8)regardless the wall types. For the 2.4 GHz test highest loss was measured truth the dry wall (about 6.1). The researchers conclude with that the 868 MHz has proven to be more reliable than 2.4 GHz band because of its better propagation and lower possibilities of interference. The results presented in this paper [15] are valuable for the modelling of our system and transmission scheme design and implement on the real-life based test scenarios to evaluate and compare the system performance.

While researching on the related work done in this particular field (WSN), we observe that most work is done by concentrating or preferring either long distance in range of area for data transmission or quality of the services that the designs are meant for. Low data rate packet adaptation are noticeable in many interesting work and many of them tried to conclude with the fact that a long distance of transmission can be hazardous due to path loss and packets drop ratio over distance. Other researchers proposed their own protocol which can achieve long range by compromising the loss of data. Although a few have also managed to successfully achieve a long distance without compromising the data, their system design was considerably expensive to implement in the real-life scenarios. We aim to find a working scheme with a radio module that can cover an impressive amount of range so that can acclimate to both low data rate packet transmission and long distance in our system design.

Chapter 3

System Design and Integration

In our system design, we implement different types of services in short interval to send via the Connect2Pi radio module which operates on the 868 MHz band to the destination which is placed in long distance from the source node wirelessly. Then we develop our design to send same types of services via a relay node from a source node towards the destination node by using the two-hop transmission concept in wireless technology. Finally we develop our transmission schemes to adapt to lower data rate while retransmitting (in case of packet drops on receiver side) by calculating the number packets that have been dropped during first phrase (transmit phrase) and requesting to retransmit in the second phrase. In order to achieve such, we design our hardware and software integration and implement them in real-life scenarios.

Our system design consists of three nodes referred as sender, relay and the destination node. We will also refer these nodes as the transmitter, forwarding and receiving systems respectively in our following sections. Fig. 3.1 illustrates the design principal of our implemented design on the nodes. The basic working principal is to gather the data collected according to the selected types of services. This data will then be processed using the ARM CPU of the RPi micro-controller and send towards the relay node via the serial port of the system. The radio module attached to the serial port transmit this frames of packets towards the destination. The relay node has the same function as the destination node including the functionality to relay the data towards the destination node. Relay node process the collected data (via radio module) and checks if the sender selected the relay node as the original destination. If it has another destination address other than relay node's source address the packets will be transferred towards that selected destination. The packets will be then relay or forwarded to the destination address. Destinations address collect the data coming via radio module and send ACK or NACK accordingly. In the following sections we discuss our hardware and software integration of our system design.

3.1 System Overview

We integrate our embedded design using a radio module (ERA-Connect2Pi) which can be configurable to operate on 868 MHz band. This radio module is a low power RF device. Low power RF devices are useful when building a wireless links for remote control, metering and sense and report applications. In most case unlicensed (license free) wireless products are used to build such links. Unlicensed does not mean they are not unregulated as it simply means that the user do not need an individual license from the telecommunication regulatory authorities to build an application in these frequency bands. The product that will be used for such project need to be strictly regulated and be certified by the regulatory authorities.

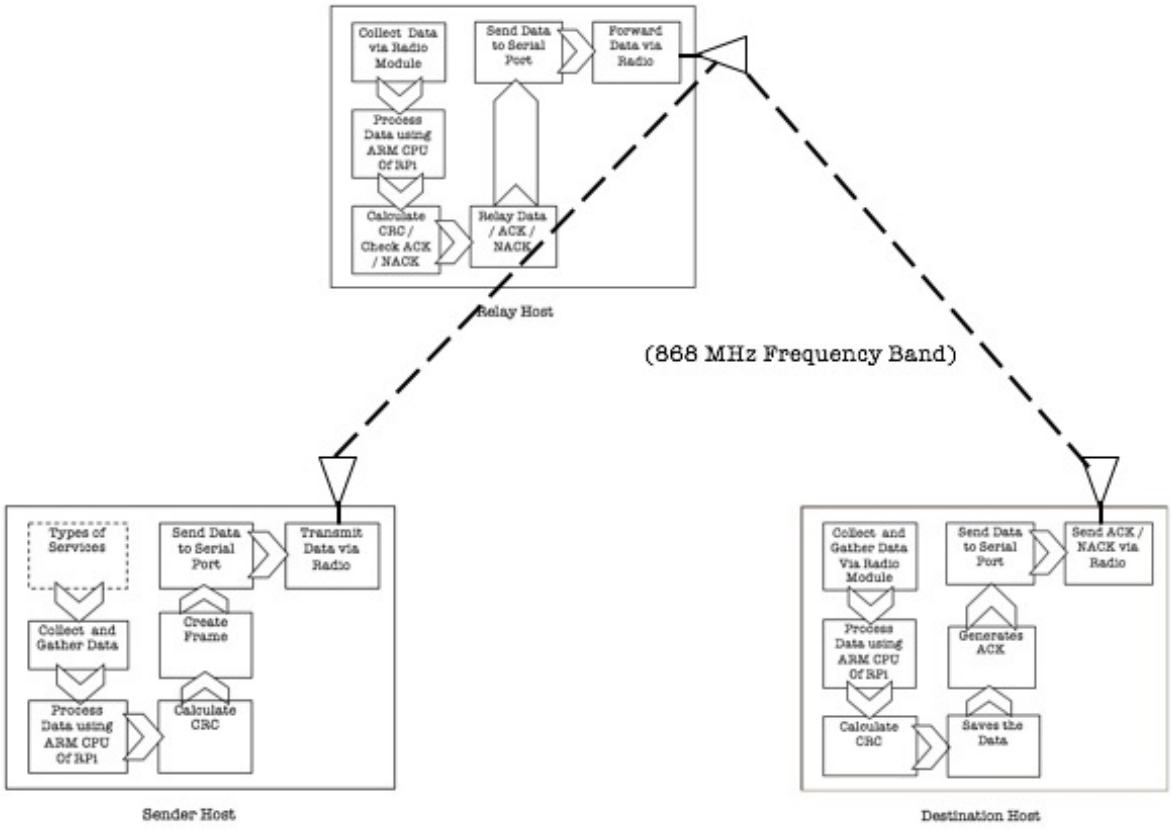


Figure 3.1: Illustration of Working Principal of the System Design

3.2 Overview of the Radio Module

Our radio module (ERA-Connect2Pi) is designed to operate on 868 MHz frequency band. It is a highly configurable radio and we choose this module to be integrated in our embedded system design. We give a brief description of 868 MHz in Sec. 2.1.2 and we discuss further the usability of this ISM band in this section.

3.2.1 868 MHz frequency band

Like we discussed earlier in Chapter 1, the SRD products are used for European Union for unlicensed applications. EU regulatory agencies are inch are of placing limitations on the operating frequencies, output power, modulation methods and transmit duty cycles, among other things to these products. These regulations governing low power wireless devices are defined by two separate bodies. One group defines the allocation of frequency bands and their use and the other group defines the test methodologies and general transceiver specifications. ECC (part of CEPT) governs the SRD products limitation allocations which is followed by most countries in Europe, occasionally different between member countries. Thirteen different type of SRD applications are defined by ECC.

The idea behind this application list is to reduce the interference frequency bands and to have a dedicated purposes for them. Although any application can be defined in any non-specific SRD but the advantage of this list to confirm that no other application will be allowed in its undes-

Table 3.1: Frequency Allocation for SRD860 [16]

Frequency	**Duty Cycle**	**ERP**
863 - 865 MHz	100% (wireless audio)	10mW
863.0 – 865.6 MHz	0.1% or LBT+AFA	25 mW
865.0 – 868.0 MHz	1% or LBT+AFA	25 mW
868.0 – 868.6 MHz	1% or LBT+AFA	25 mW
868.7 – 869.2 MHz	0.1% or LBT+AFA	25 mW
869.4 – 869.65 MHz	10% or LBT+AFA, 25 kHz channel spacing	500 mW
	10% or LBT+AFA	25 mW
869.7 – 870.0 MHz	100% (voice communication)	5 mW
	1% or LBT+AFA	25 mW

ignated area. But we have to keep that in mind that any application can be done on any other application field for research purposes, considering the high chances of interferences. The ECC recommendation 70-03 defines the maximum transmit power and limits to the bandwidth and the duty cycle for each predefined frequency band. As we discussed some part in Chapter 1frequency band specified for license free specific application in Europe with their corresponding channel bandwidth are represented with Table 3.2.

Table 3.2: Frequency Band for License Free Specific Application in Europe [17]

Frequency Band	**ERP**	**Duty Cycle**	**Channel Bandwidth**
402 to 405 MHz	–16 dBm	No limits	25 kHz[1]
868.6 to 868.7 MHz	+10 dBm	$< 0.1\%$	25 kHz[1]
869.2 to 869.25 MHz	+10 dBm	$< 0.1\%$	25 kHz
869.25 to 869.3 MHz	+10 dBm	$< 0.1\%$	25 kHz
869.65 to 869.7 MHz	+14 dBm	$< 10\%$	25 kHz
863 to 865 MHz	+10 dBm	No limits	200 kHz
863 to 865 MHz	+10 dBm	No limits	300 kHz
1785 to 1800 MHz	+7.85 dBm	No limits	200 kHz
2400 to 2483.5 MHz	+17.85 dBm	No limits	No limits[2]
2446 to 2454 MHz	+24.85 dBm	No limits	No limits
2400 to 2483.5 MHz	+11.85 dBm	No limits	No limits
2446 to 2454 MHz	+24.85 dBm	No limits	No limits
2446 to 2454 MHz	+33.85 dBm	$< 15\%$	No limits

Standard for low power wireless devices are defined by the ETSI. As with ECC recommendations, these standards have to be translated and enforced by all the EU countries. Non EU countries may use these standards for regulatory purposes. The most important harmonised standard for low power wireless devices in the frequency range between 25 MHz and 1 GHz in EN 300 220, and EN 300 440 applies to the frequencies between 1 GHz to 40 GHz. In cases of standards can no be applicable, a special certified test laboratory has to asses the compliance declaration. These standards are as separated into two parts, transmitter tests and receiver tests. In transceiver tests, which is distinguish between narrow and wide band systems. A radiator is a wide band transmitter and works in a continuous frequencies covering more than 25 kHz. Otherwise it is called a narrow band transmitter.

3.2.2 868 MHz enabled radio modules

Among many radio modules which are 868 MHz frequency enabled, we consider some of them as our potential candidate for our system design. Not all the products meet our desired goal considering their technical capabilities such as modulation schemes, data rate and sensitivity. We prefer also the transceivers as we want our's to be capable of both transmit and receive (not necessarily at the same time) while communicating. Table 3.3 shows the most commonly used 868 MHz band enabled radio components which we find most capable for our system design.

Table 3.3: Types of 868 MHz Enabled Radio Modules

Module	**Antenna**	**Frequency**	**Output Power**	**Data Rate** (up to)	**Range** (up to)	**Channels**	**Power Consumption**
GAMMA-868 (RF Telemetry Module)	No	868 MHz CE /915 MHz FCC compliant	+13 dBm	9.6 Kbps	16 Km	4 Acknowledged Channels	1.8 to 3.6 V
AMB8465 (Wireless 868 MHZ USB Adapter)	SMA Connector	863.0 MHz to 868.6 MHz	+2 dBm	500 Kbps	2000 m	Configurable RF	TX: 45mA, RX: 20mA
RF-LoRa-868-SO (LORA Transceiver Module)	Antenna Pin Connector	Preset to 868 MHz	+20 dBm at 100 mW constant RF	300 Kbps	16 Km	Configurable	2.2 to 3.7 V
KAPPA-M868-SO (Radio Modem Module SMT Package)	No	868 MHz	+13 dBm	56 Kbps	500 m	5 User Selectable	2.4 to 3.6 V
SMART ALPHA-868 (Smartalpha Wireless Tranceiver Radio Modem)	Pin Available for Helical or Whip Antenna	433 Mhz/868 MHz/915 MHz	-21 dBm	19200 bps	300 m	Selectable Narrowband RF	2.3 to 3.6 V

[1]The whole frequency band may be used as one channel for high speed data transmission
[2]Maximum power density $\leq$ 7.85 dBm/1 MHz for DSSS systems and $\leq$ 17.85 dBm/100 kHz for FHSS systems

BRAVO-T868 (BRAVO-T Tranceiver Telemetry)	Pin Available	868 MHz	+13 dBm	9.6 Kbps	1000 m	8 Channel Transceiver	1.8 to 3.6 V
ZULU-T868-SO (Zulu Smart Radio Telemetry SMT Transceiver)	Antenna Input/Output Pin	868 MHz	+20 dBm	9.6 Kbps	2000 m	10 Channel Transceiver	2.2 to 3.6 V
XBee-Pro 868	Wired Whip /U.FL connector /RPSMA Connector	868 MHz	+25 dBm	24 Kbps	Indoor: 550 m, Outdoor: 40 Km	Single Channel	3.0 to 3.6 V
eRA900-TRS (FCC ID:SLW-ERA9TRS)	50 Ω RF input/output	804 MHz to 940 MHz (Programmable)	5 mW	2.4 Kbps to 115.2 Kbps	2.2 Km	Up to 132 Channels	2.5 to 5.5 V

Table 3.3 shows the variety of radio modules with their corresponding attributes. In our research process we gather these product informations to compare their properties and choose the most compatible module for our protocol design implementation by comparing the properties of these radio modules by their transmission range, sensitivity and data rate. In addition, we try to find the radio that is compatible with our microprocessor (RPi). Note that, although some radio modules have a greater range in indoor and outdoor area but in real life scenario the range can be much lower than the stated range value given by the manufacturers. Keeping that in mind we implement our system design with the eRA900-TRS radio module which comes with the USB serial connector for Raspberry Pi and PC.

3.3 System Design Requirements

In our research, we compare many radio modules which has the capability to operate on 868 MHz. We come across some of the basic principals to consider on a wireless radio module before we design and implement in our system.

- The first principal of choosing a radio module is to consider that, as frequency raises available bandwidth typically rises too. This means the distance and ability to overcome obstacle while transmitting can reduce too. For example, 915 MHz installation will have less a path loss (around 8.5 dB) than of 2.4 GHz installation. The more sensitive the radio, the lower the power signal it can successfully receive, which means it is more exposed to the noise

level.

- In low power transmission range, the manufacturers rated receive sensitivity is a key factor in wireless system and range estimates. Although the receive sensitivity can be improved by reducing the transmission baud rate as receive sensitivity is a function of the transmission baud rate. Many radios have the option to reduce the baud rate according to user's preferences. The sensitivity can also be improved by using lower frequencies such as 868 MHz.

- The RF background noises generates from various sources such as solar activity or high frequency radio communications components. Typically the noise floor (where the background noises are established) are lower than the specified receive sensitivity of the radio, so it will not be a factor in our system design.

- We also need to consider while choosing a radio module that how many dB a received signal may reduced by without causing system performance to fall below an acceptable value. This scaled the fade margin. Usually minimum 10 dB of the fade margin level can guarantee the high performance of the system which will operate significantly in a variety of weather, solar and RF interference conditions. Some radios have a programmable output power which can be dialled to the right measurement in case of performance degradation.

- The range of a given radio signal in wireless communications also need to be considered. In a clear path via air, radio signals attenuate with the square of distance. Doubling the range requires a four fold increase in power. So halving and Doubling the distance respectively decreases path loss by 6 dB and increases by 6 dB as well. When indoor, we should use more aggressive rules in such to change the both values to 9 dB as paths in indoor tends to be more complex. Radio manufacturers advertise line-of-sight range figures which means in general the antenna of a transmitter can see the antenna installed on receiver. As we discussed before, every obstacles in the path of line-of-sight can degrade this. In this case, the type, size, number of the obstacle play a role in the path loss. A single obstacle will not much affect on the path loss but the obstacles located near the antennas cause a dramatic path loss. Most effective way to reduce the path loss is to elevate the antennas at approximately 2 m, as the Earth's curvature is about 5 km, so anything taller than the ground level will help to reduce it.

- Weather conditions can also play a role in the path loss as we discussed briefly on the Sec. 2.1. Higher the moisture in the air, higher the path loss. Some obstacles are mobile, such as vehicles and metal housings, which need to reconsidered while designing the wireless system. This rules apply to the obstacles such as trees as well. Levitating the system over the treetop in a woodland area can usually solve the problem. We can consider the basic path loss rules of thumb to avoid such problem [18].

- Considering the impact of wireless communication in our system design is important as well. Acceptable bit error rates are many order of magnitude higher than wired communications. Most radios can handle error detection and can retry automatically to transmit on behalf of users, at the expense of variable latencies and throughput. Our application software must be well designed and transmission schemes must be tolerant of variable latencies. Not every protocol can tolerate the radio interference and latencies. The protocols which are sensitive to inner-byte delays may require special attentions or specific support from the radios.

3.3.1 Our system design principle

In our system design, we first consider choosing the right hardware to achieve our goals. The problem, as we discussed in Chapter 2, with the hardware and modules with low configurations required to achieve such transmission scheme design which need to be self-contained and highly efficient on data transmission. We briefly explain our system design principle as follows:

- Low power, low data rate enabled hardware need to be re-modified or reconfigured in such a way so that it can operate on the licence free 868 MHz band. We select ERA-Connect2Pi radio module which is programmable with user's preferences. We have to configure our radio module in such a way that it can communicate with the other radios and send and receive data in same data rate while avoiding the interference and noises created by other devices operating on different frequency bands.
- The radios need to be re-configured to be able to transfer data efficiently in the wireless network. By re-configured we meant the term of software manipulations specially according to the channel configuration.
- The designed embedded system need to able to retransmit in case of data loss or packet loss while transmitting cause by miscellaneous reasons and this recovering from such situation can be done after receiving response or acknowledgement from the receiver.
- In the two-hop scenario, our system can automatically identify the real destination node and should response accordingly. By real destination node we meant the node for the which the transmission is meant for by the source node.
- Our system should be configured to handle various types of services preferred by user. We mainly focus on services such as random data, image transmission, GPS co-ordinates and ambient temperature collected from the source node. We will discuss more in details in Sec. 3.5.
- Our systems need to automatically adjust their radio settings in order to adapt to a lower data rate to ensure faster transactions (for the transmission scheme III)

3.3.2 Transmitter system integration

The transmitter system we design to integrate in our system is configured to send data bytes to the receiver via the 868 MHz radio module. The data bytes are in different types of service which we discuss in details in Chapter 4. According to the services the transmitter configure itself to the necessary settings and response correspondingly. Transmitted system has the ability to let user choose the destination nodes to connect to it and send the chosen types of service to that destination. Transmitter or sender system also responsible to have stable connectivity of the ongoing transaction and it can response to the acknowledgment sent form the destination systems. To start with our transmitter system can send random string of bytes to the chosen destination. We also configure the transmitter system to divide an image in multiple bytes and send it wirelessly to the destination node in blocks of bytes. It also stores the payloads in a buffer in case the destination system request a retransmission until the entire transaction is finished. A GPS module is connected to this system to gather GPS information and to send to destination. Transmitter system can also gather temperature around its environment and can send it. In some part of the project, we are also referring this transmitter system as sender node.

3.3.3 Forwarding system integration

Our forwarding system can receive the incoming packets as retrieve the information from it. We design our forwarding system and integrate in such a way that it can determine which real-destination node the sender requesting to connect to. If it confirms that the transmitter system requesting to connect and send data to forwarding system itself then it select itself as the destination address. Then the forwarder we act as the destination node and establish a connection based on requested type of service. The forwarder system is designed to response to every incoming packets from sender with an acknowledgement and store the payload extracted from those packets. We also refer this system as the relay or forwarding node in many future discussion of this project.

If the forwarding node determines (from the incoming destination addresses from sender) it will then activates the relaying process and select the requested destination node as the destination. Upon selecting the forwarding node will then request the connection invitation for the transmitter node and continue to forward the incoming packets to the destination node (receiver) and again forward the acknowledgement incoming from receiver system (upon successful transaction) to the transmitter node. As a receiver system the relay node will exactly response as the receiving system and as a forwarder this system will work as relay node. Forwarding system will response as receiver system or forwarding system based on which transmission scheme has been implemented.

3.3.4 Receiving system integration

The receiver system or as we sometimes refer it as destination node, is almost similar to the forwarding system except it is designed only to gather information incoming from the transmitting system (transmission scheme I and III) or via a forwarding system (transmission scheme II). Based on the schemes, the receiver choose either the sender or the relay node as its destination to send the acknowledgements for successful transaction or the retransmission requests.

The Fig. 3.2 gives an overview of our system design implementation in real-life scenario.

3.4 Hardware Integration

To start with the hardwares integration, we choose to prepare our RPis by installing operating systems on the SDHC cards. We choose Raspbian Wheezy as OS. While setting up the Pis, we choose three Pis for scheme implementation as we will implement our transmission scheme on test with three nodes. In our first version of the transmission scheme one of the RPi will be set up as the source (sender) and the other one is to set up as the sink (receiver). For the version two of our scheme we will convert our second RPi as a forwarding / relay node to transfer the data packets to the destination node (RPi). The third RPi will be configured as same as the second RPi in first scheme. While setting up OSs, we decide to maximise the CPU usage of the hardware by choosing the overclock speed of the Pi's to High (950 MHz). As the manufacturer of the Raspberry Pi declared that Pi 1 model B+ can be overclocked to maximum presets of 950 MHz ARM, 250 MHz core, 450 MHz SDRAM and 6 over-volt without needing of any heat-sink to be attached to the Pi's CPU. The baudrate need to be configured according to our radio module Connect2Pi's baud rate which is 19200 configured by our scheme design. This will ensure our faster CPU implementation along with other modification in our process. The overall hardware setup idea is as follows:

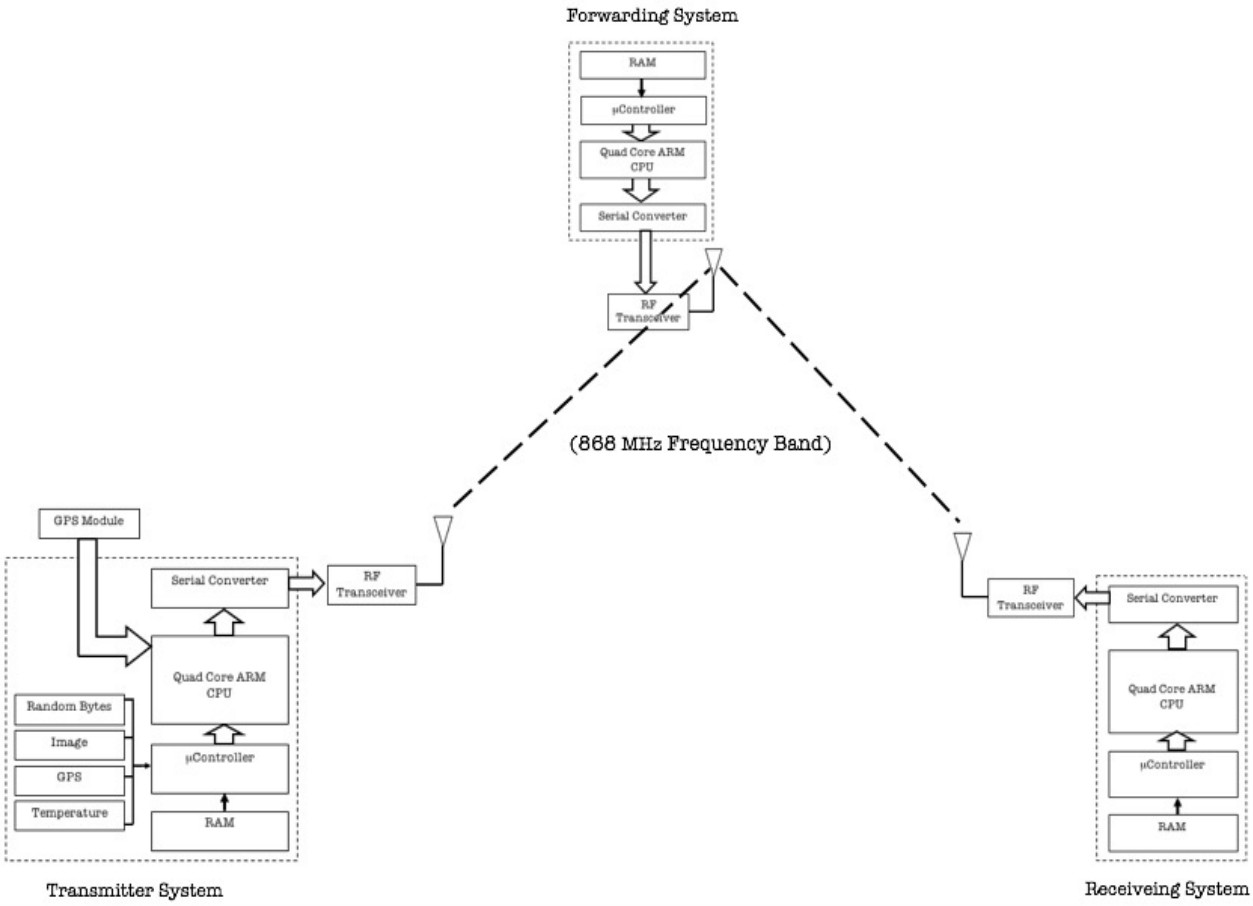


Figure 3.2: Block Diagram of Integrated Systems

- Our first-end of the system design consists with a Adafruit ultimate GPS breakout and a Connect2Pi radio module attached via USB port of RPi to communicate with other nodes placed on the other end of the transmission. The first end node in the system is considered as the source node. For simplicity, from now on we are going to address this system as **A**. In addition we will store a copy of our test picture in order to use the image transferring service which will discuss in details in Chapter 5.

- The sender node will capture GPS co-ordinates using the GPS module and send it via the radio module to destination node. The node on the other end on the scheme design is considered as destination node in transmission scheme I and relay node for transmission scheme II. We are going to address this node as **B** for simplicity.

- Node B will be also configured with a Connect2Pi radio module which is reconfigured to the exact same settings as the radio module attached to node A.

- Since we are designing our own data transmission schemes we used "**0A**" and "**0B**" as the source and destination address of the nodes respectively.

- In transmission scheme I node A will collect the data according to the service type requested by user via its simple user-friendly interface and will be sent to node B.

- Node B will then gather the data and will run the procedure and use the data according to the service type.

- According to transmission scheme II, we configure the node B as the relay node. We

include another RPi with the same hardware and software configuration and attach another Connect2Pi USB dongle to it. It is denoted as node **C** and addressed as "**0C**" in our scheme.

- In second scheme, node C will be considered as destination node and node A will request its transmission to the node B to forward its collected data to node C.
- Node B will then collect and forward the data at the same time towards C.
- Node C will receive the data and act as the destination node and use the data received according to the type of service chosen by the user in node A.
- The third transmission scheme is similar to scheme I as it is also implemented on node A and B where A is the sender and B as the destination node. The difference between scheme I and III is that in scheme I the receiver request the for retransmission while the transaction in ongoing with sender node. But on transmission scheme III, we design the receiver wait for the sender to finish the entire transaction until it has no more payload to send for that period (in case of sending a large file) and then activate the retransmission process by calculating if there any packets have been dropped in the first phrase and requesting for retransmitting to sender node accordingly. We discuss these schemes design in details in Secs. 4.3 - 4.5.

3.4.1 Block diagram of embedded design

The block diagram in Figs. 3.3(a) and 3.3(b) illustrates the embedded design we employ to implement our designed transmission schemes.

Fig. 3.3(a) shows the block diagram of the transmitter system. Our transmitter system is configured with a ERA-Connect2Pi radio module to transfer data to the other nodes in the same network, an Adafruit ultimate GPS breakout to collect GPS information and send it to the transmitter when the GPS transmission is ongoing as service type and a micro-controller to implement the transmission schemes on the system and to control the radio and GPS module.

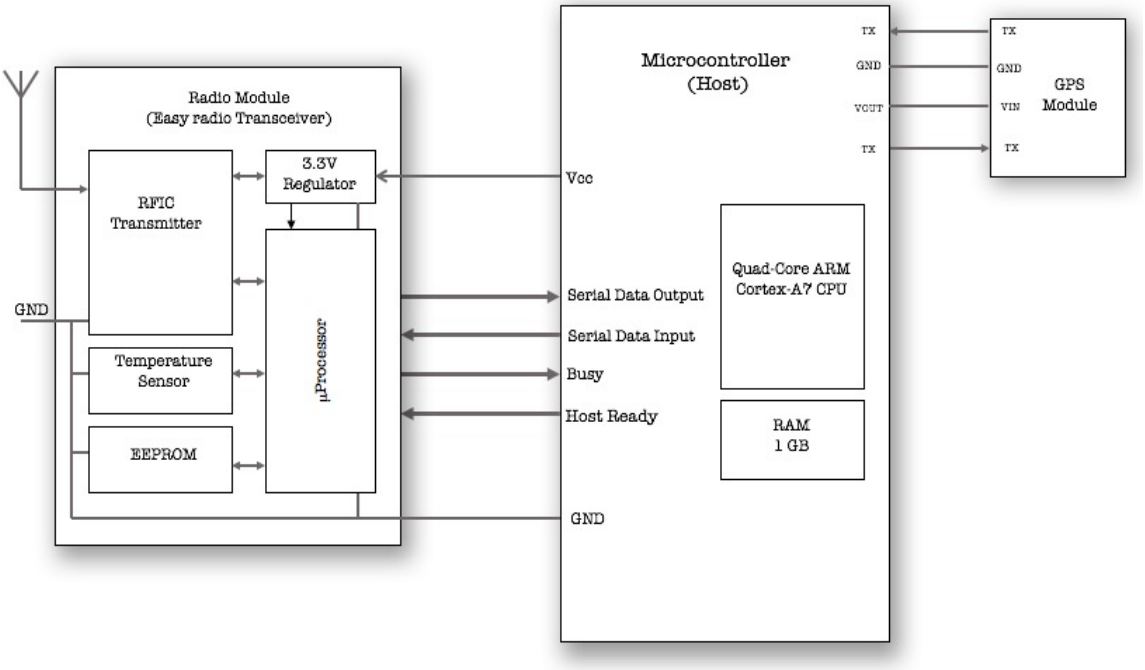
Fig. 3.3(b) refer to the block diagram of our designed forwarding and receiving system. Both of these systems have a radio and a micro-controller each for the same functionality as the transmitter system except these two systems do not have any GPS module attached to them. Our relay node will only gather data packets incoming from the sender node and relay them to destination node if requested. As a receiving system the node will only collect the data from the payload sent from transmitter system. It will not gather any other information such as GPS or temperature by itself. In the forwarding and receiving system data will be collected only from the packets sent from transmitter system.

3.4.2 Peripheral components on transmitter system

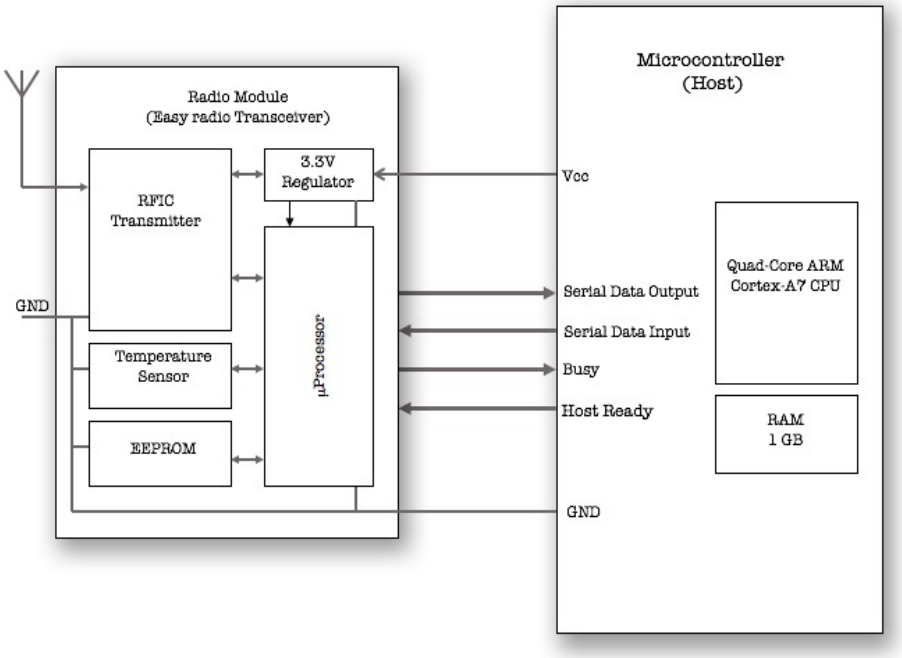
Some other peripheral components we use in our transmitter system setup which is used to implement our services and system integration. The brief description of those components are as follows:

Adafruit Ultimate GPS Breakout v3

In our masters thesis project we use the Adafruit Ultimate GPS device to collect GPS information and send it via our designed transmitting system. This breakout is built around the MTK3339 chipset, a high quality GPS module that can track up to 22 satellites on 66 channels. It has an



(a) System Design of the Transmitter system



(b) System Design of the Forwarding and Receiving system

Figure 3.3: Illustration of Embedded System Design

excellent high-sensitivity receiver (-165 dB tracking), a built in ceramic antenna. It can do up to 10 location updates a second for high speed logging or tracking. Power usage is incredibly low (only 25 mA while tracking and 20 mA during navigation). To have a bigger antenna we can attach any 3 V active GPS antenna via the uFL connector. The module automatically detects the active antenna and switch over. The ultimate GPS breakout also comes with an optional CR1220 coin cell to keep RTC running and allow warm starts. It has a led indicator to show

satellite searching (blinks at about 1 Hz) and a satellite fix (blinks once every 15 sec to conserve power). It has a built in data logging system which can used by users optionally. It also has the jammer detection and reduction with multi-path detection and compensation [19]. Some of the main features of this radio module are as follows:

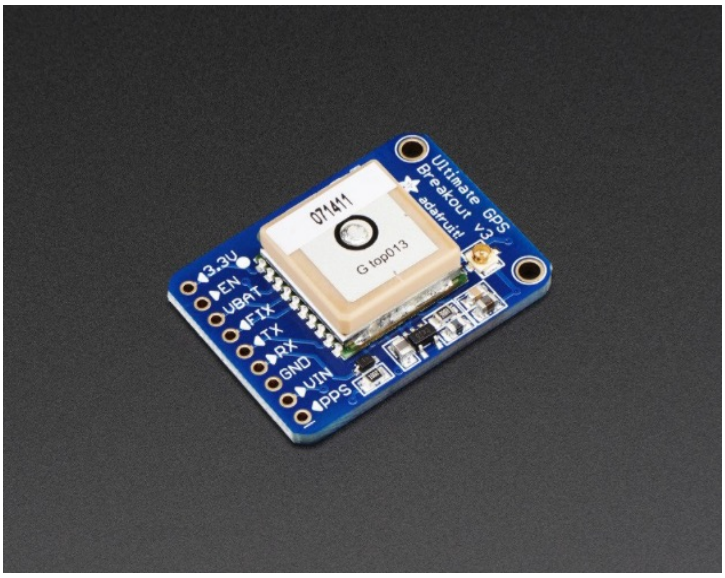


Figure 3.4: Adafruit Ultimate GPS Breakout v3 [19]

- Patch Antenna Size: 15mm x 15mm x 4mm
- Update Rate: 1 to 10 Hz
- Position Accuracy: <3 m
- Velocity Accuracy: 0.1 m/s
- Warm/cold Start: 34 sec
- Acquisition Sensitivity: -145 dBm
- Maximum Velocity: 515 m/s
- Vin Range: 3.0 to 5.5 VDC
- Output: NMEA 0183, 9600 baud default
- DGPS/WAAS/EGNOS supported
- PRN Channels: Up to 210
- Weight: 8.5 g (without coin cell)
- Dimensions: 25.5mm x 35mm x 6.5mm / 1.0” x 1.35” x 0.25” (without coin cell)

Memory Card (SanDisk 8GB MicroSDHC)

It features:

- Seamless speed and performance with microSDHC compatible devices
- Class 4 Speed performance rating (based on SD 2.0 Specification)
- Item Weight 5 g
- Product Dimensions 1.5 x 1.1 x 0.1 cm
- 8 GB in total size

In addition, we use USB 3.0 extension cable to power up the Raspberry Pi with 5 V and a regular Ethernet cable to SSH to Pi via VNC (a client to SSH in to system) and Telnet (for Windows and Unix platforms). We also use a MAC to burn the images to SD card and SSH the systems while running it. ERA-Connect2Pi USB dongles are connected to one of 4 USB ports of Pi. All sending, forwarding and receiving system configuration need to attach a USB radio dongle and a microSDHC card to each RPi.

3.5 Software Integration

We start designing our system by first configuring all the nodes with the same settings in such a way that it can capture the data bytes from the sender node and send it over to the destination node/s. This data bytes will differ for various types of services selected by the user's input. In this software integration section we discuss the platform development and adjustment of setting of RPi. This is necessary because our nodes will be based on this RPi micro-controllers and these nodes need to compatible with our radio module devices and the GPS modules. Software setup of a node is equally important as hardware integration. Every machine needs to talk to each other in their own language and for this communication between every hardware components setting, the compatible software integration is also necessary in an embedded system design. To achieve such we configure our node / systems with the necessary software configuration.

3.5.1 Overview

As we mentioned earlier, our radio module Connect2Pi can only read or write data bytes from the nodes once at a time. It can be either data in or data out at a time. We ensure that our schemes can control this data flow in such a controlled manner so that it doesn't overflow the sender node neither nor the receiver node/s. Only maximum 180 bytes can be read / write by this chosen radio module. Our application need to be focused on designing a smooth flow of data to maintain and avoid buffer overflow. We setup our nodes with operating systems and install drivers for our radio module. We also need to setup our GPS module in our transmission system which we discuss in the following sections. We install our programming tool (Python) on our RPi hosts to operate, implement and run our designed schemes on them. We also set our baud rate configuration of peripheral devices in this part. Sec. 3.5.2 gives detailed description to the reader about the setup procedure for our software integration.

3.5.2 Setup procedure

We start with burning the Raspbian Wheezy (any version of Raspbian, Noob , etc., will work as well) image in SD card by using a PC. We insert the microSDHC card into an SD card reader slot and format it. SDFormatter, which is a free tool, is used to format the SD cards, but a user can choose any other software they are comfortable with. After formatting, we use a disk imaging tool (e.g., Win32 Disk Imager) to select our Raspbian Wheezy version and let it burn the image to SD card. Now we insert the card into Pi's microSD card slot and power it up by using the USB 2.0 extension cord and a 5 v USB power brick. While boot up we see the **raspi-config** menu where we have to make sure we enable the **expand_rootfs** with the feature **Expand root partition to fill SD card**. This need to be done because when we flash our SD card, an exact copy of the operating system and its disk formatting is copied to the card. As a result, the SD card can look like it has less capacity than it really does, which means we can quickly run out of space, even on a higher capacity card size. One of the first things you should do is use this option in raspi-config to ensure your Raspberry Pi can use all the space available on your SD card. When we press Enter with this option highlighted, it runs straight away. At our next start up of Raspberry Pi, the Pi resizes the file system, which can take a few minutes, during which the screen will not update. The new capacity then becomes available to us. Additional option we can enable is **SSH** from the advanced options of raspi-config. SSH is a way of setting up a secure connection between computers, usually, we can control one computer from another computer (optional). We setup the overclocking presets to high or turbo mode depending on the model versions of Raspberry Pi and power supply a user has. Pi has five different presets to choose from. The speed of the CPU is measured in MHz, and the highest overclocking setting increases the speed from 700 MHz to 1000 MHz computer. Rest of the options setting such as overscan, locale, keyboard configurations, boot behaviour, password etc are optional. Then we restart the system and Pi install the OS in the system. This technique is same for both sender and receiver system in our design.

Now we start with configuring the sender system by updating and upgrading OS of Raspberry Pi by using the following commands. It is always a good idea to run these commands to make the system up to date before installing any packages in the OS. Since the Raspbian Wheezy is a Linux-based operating system the commands in the terminal are as follows:

```
1  $ sudo apt-get update
2  $ sudo apt-get upgrade
3  $ sudo reboot
```

After rebooting, we open the terminal again and start with the installing the packages required to enable the radio work with serial port of the raspberry pi. In addition, we configure the ultimate GPS breakout to work in sender node. We attached the configuring process of our system components to the hosts (RPi). First we edit our **cmdline.txt** on the boot folder on RPi to enable the connected radio module (via USB port) as our only listening console for the RPi. We find our port for attached radio on RPi by running this command on RPi terminal:

```
1  $ sudo ls /dev/ttyUSB*
```

This returns the port that attached to our radio module. In our case it is **/dev/ttyUSB0**. Now we enable it by editing the cmdline.txt according to this:

```
dwc_otg.lpm_enable=0 console=ttyUSB0,19200 console=tty1 kgdboc=ttyUSB0,9600 root=
```

```
/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait usbhid.mousepoll=0
```

Here the ttyUSB0s are configured to be only port chosen for serial module to send / receive data over while we run our applications. Since we are using Raspbian Wheezy as our RPi we have to disable the respawn of ttyAM0 by opening **inittab** on the **etc** folder on RPi. Then find the following line disable it by using comment (#):

```
# T0:23:respawn:/sbin/ghetty -L ttyAMA0 115200 vt100
```

Now we add our **dialout** to the **pi** group:

```
1  $ sudo usermod -a -G dialout pi
```

By running the command **id** on terminal we can check if dialout is in the pi group which will be represented by this.

```
uid=1000(pi) gid=1000(pi) groups=1000(pi),4(adm),20(dialout),24(cdrom),27(sudo),
29(audio),44(video),46(plugdev),60(games),100(users),106(netdev),996(gpio),
997(i2c),998(spi),999(input)
```

We need to change the **stty** to work on 19200 to display or change the characteristics of the terminal:

```
1  $ stty -F /dev/ttyUSB0 19200
```

Our radio module (ERA-Connect2Pi) uses FTDI driver. We need to install the necessary drive to the RPi to run the USB dongle with our system nodes. We check if our FTDI device is connected properly by running the command **dmesg — grep FTDI** on the terminal which will response with something like this:

```
[ 6440.280777] usb 1-1.4: Manufacturer: FTDI
[ 6440.383321] usbserial: USB Serial support registered for FTDI USB Serial Device
[ 6440.383939] ftdi_sio 1-1.4:1.0: FTDI USB Serial Device converter detected
[ 6440.388100] usb 1-1.4: FTDI USB Serial Device converter now attached to ttyUSB0
```

We now install the **pyserial** on RPi which is not included on the RPi distro. The second command will allow a user to run a python script which imports the serial module. We also install **python-pip** and upgrade its contents via **pip**.

```
1  $ sudo apt-get install python-setuptools
2  $ sudo easy_install -U pyserial
3  $ sudo apt-get install python-pip
4  $ sudo pip install --upgrade pyserial
```

In Linux, the VCP driver and D2XX driver are incompatible with each other. When a FTDI device is plugged in, the VCP driver must be unloaded before a D2XX application can be run.

Use the remove module (rmmod) command to do this. When the FTDI device is power cycled or reset the VCP driver will be reloaded. The rmmod process must be repeated each time this occurs. It is possible to write a simple script that unloads the VCP driver before running the D2XX application.

```
$ sudo rmmod ftdi_sio
$ sudo rmmod usbserial
```

We download the right software from the FTDI D2XX driver web page [20]. In our case for software integration it is **1.3.6 ARMv6 hard-float** for RPi. This archive contains the D2XX driver and directory of sample code. Using following commands on terminal we setup the FTDI driver on RPi (assuming the downloaded .tar.gz file placed on **/home/pi** folder).

```
$ gunzip libftd2xx-arm-v6-hf-1.3.6.tgz
$ tar -xvf libftd2xx-arm-v6-hf-1.3.6.tar
$ sudo cp /home/pi/release/build/libftd2xx.* /usr/local/lib/
$ sudo chmod 0755 /usr/local/lib/libftd2xx.so.1.3.6
$ sudo ln -sf /usr/local/lib/libftd2xx.so.1.3.6 /usr/local/lib/libftd2xx.so
$ cd examples/
$ make -B
$ cd EEPROM/read/
```

Now plug in our radio module to the RPi and run this command **sudo ./read** which runs an application to list the configuration descriptors of the attached FTDI devices similar to this:

```
opening port 0
ftHandle0 = 0x8e89220
Signature1 = 0
Signature2 = -1
Version = 2
VendorId = 0x0403
ProductId = 0x6001
Manufacturer = FTDI
ManufacturerId = FT
Description = USB-Serial Converter
SerialNumber = FTG5FL9U
```

We upgrade RPi's distro and reboot:

```
$ sudo apt-get dist-upgrade
$ sudo reboot
```

We find the attributes of radio such as **idVendor**, **idProduct** and create the following file on **/etc/udev/rules.d** folder:

```
$ udevadm info --name=/dev/ttyUSB0 --attribute-walk
$ sudo nano /etc/udev/rules.d/99-usb-serial.rules
```

We edit and save the file with the following information we collected from the previous commands:

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SYMLINK+=
"ttyUSB0"
```

This ensures that our host RPis will look for the above attribute informations every time it is turning on. In order to turn off hardware flow control of the radio module we run the following setup.

```
$ sudo apt-get install minicom
$ minicom -b 19200 -o -D /dev/ttyUSB0
```

And we open the options by pressing ctrl+A and then press "O" and "F" to disable the hardware flow control. We need to save the configuration as dfl and exit the settings options. After rebooting the host RPi it will be ready to run our application code and can send and receive data via our radio module.

We have to configure our radio modules with exact same settings to communicate to each other and operate on the 868 MHz band. We set our settings to exact same radio configuration on our Connect2Pi USB dongle by running the software **easyRadio Companion** provided by the manufacturer. To start with, we configure all the radios in the following settings except for scheme II of our design where we change the data rate to different settings for test scenarios. We will discuss more about the following in Sec. 4.4.

- UART: U4(19200)
- Channel: C0
- Power Level: P9
- Bandwidth: B3 (100 kHz, 19200 bps)
- Band Plan: b0

Additionally we set the frequency of eRA900 radio module of the USB dongle to set on 868 MHz by opening the Advanced eRA Frequency Settings tab on the software and choosing 868 MHz for eRA900 and program it to its default settings. Now our radios are ready and set to implement in our system design.

Above settings and setup are same for node A, B and C. Additionally we attach our ultimate GPS breakout v3 to one of the available USB port on sender node (A) and setup the module according to the following method:

- We attach a USB to TTL serial cable to GPS module to connect with RPi. We connect **VIn** pin of the red connector, **GND** pin to black connector, **RX** to blue connector and **TX** pin to the white connector of the serial cable. Now we attach the USB to the RPi USB port.
- We check which port its attached to by running the following command on RPi terminal:

```
$ ls /dev/ttyUSB*
$ sudo lsusb
```

- The second command will return the information of the GPS module on the terminal.
- We need to install the GPS daemon (gpsd) on the RPi.

```
1  $ sudo apt-get install gpsd gpsd-clients python-gps
```

- Then we run the commands one after another overtime we restart our device so it will run properly. We included the following commands on our system design so that the user do not need to run these codes every time they attach / detach the GPS module to the RPi.

```
1  $ sudo killall gpsd && sudo gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock
```

- Our ultimate GPS breakout is now configured on node A. Running **gpsmon /dev/ttyUSB0** will show the running GPS informations collected by the GPS module on the terminal.

Our nodes are now configured to be implement our designed scheme. We describe our transmission scheme design in Chapter 4 in details for readers to understand the working principal of our system.

Chapter 4

Transmission Schemes and Services: Design and Implementation

We design three scheme of transmission which can transmit and receive low data rate packets over long distance. We aim to gain long distance by using our schemes in our system design and develop this system to advance level to include a relay node to forward the transmitted packets from sender to the destination nodes. We categorise our schemes in three versions. Transmission scheme I consists of two nodes which are node A and node B as we discussed briefly on Sec 3.3.1. In transmission scheme II, we include the third node C which act as a destination and node B turns into relay node to forward incoming packets from node A towards node C. Finally in transmission scheme III, a technique is implemented with which the nodes will enable retransmission phrase at the second phrase after the sender finished sending the first phrase of transmission of fresh packets. This retransmission phrase will then determine the missing packets on receiver and the sender will send those specific packets again according to receiver's requests. Our data packets are used in form of different types of services. Based on services and the schemes the packets are constructed in different manner and our transmission scheme implementation depends greatly on this frame structure of the packets. We describe the frame structure of our low data rate packets in the following section. In our scheme design, all the transmission schemes are using the same frame structures and same types of services except slight alternation on scheme III (see at the end of Sec. 4.2). These frame structures and types of services are discussed in Secs. 4.1 and 4.2.

4.1 Frame Structures

In this section, we give a thorough description of our data frame structure by using the diagrams. As we explained before, our radio module ERA-Connect2Pi USB dongles have very limited amount of data to send and receive every transmission. This module is only capable to send / receive 180 bytes in half-duplex mode. This means the radios can either read maximum of 180 bytes and write maximum of 180 bytes at a time. We must focus on the bit balancing between the data frame and construct our frame in such a way that it does not exceed the limit of maximum allocated data bytes (180) for each transmission. We also have to make sure that the data receiving on the receiver end of the transmission can also use the maximum amount of reading bytes by using the most out of the allocated limitations. For start we construct our frame structure for first types of service which is designed to transmit randomly generated bytes to the destination. Fig. 4.1 to 4.6 represents the data frame created on nodes transmit towards each other in order to maintain the harmony of data flow between them. In this simple design of frame structure, first 10 bytes are added as header part of the frame structure where the basic information of the source,

destination and types of services will be configured. The low data rate packet to send random bytes are construct as follows:

Packet Type

The first 4 bytes of the packet consists of packet type. Depending on the types of services this packet types may vary for each frames. According to our design the packet type field consists of 8 different types of packet types. They are as follows:

- Packet type '0000' [**INVITE**] is used to include on the data frame on an invitation packets from the sender node at the beginning of the transmission to request connection with the receiver or the relay node.
- Packet type '0001' [**READY**] is used in order to response to the connection with the sender node to the destination. The relay node will also send packets with this ready packet type to confirm that it is ready to receive the packets from the sender node.
- Packet type '0010' [**ACK**] is included to define the acknowledgement sending from the one node to another node to confirm that a packet has successfully received or the transmitter is starting to transmit now.
- Packet type '0011' [**NACK**] represents the packet type when receiver did not receive a packet in a period of time. NACK represents as Negative-ACK. This packet type will also be added to as a header when there is a CRC error in the receiving process.
- Packet type '0100' [**RAND-BYTES**] represents the data packets which contains random data bytes as payload.
- Packet type '0101' [**IMAGE**] included in the data frames as part of the header part to represent the incoming image payloads from sender.
- Packet type '0110' [**GPS**] is the packet type for packets with GPS payloads.
- Packet type '0111' [**TEMPERATURE**] is the packet type representing temperature payloads.

Source Address

Bytes 5-6 stores the source address of the node that while sending data packets to each other. This source address defines which node the packet is generating from and contract with 2 bytes. Each nodes have their own distinguished source address in form of 0A, 0B and 0C.

Destination Address

Bytes 7-8 holds the destination address and consists of two bytes. Destination address is used to control to which node the transmitted packets need to be reached or via which relay node it need to be forwarded.

Real-Destination Address

Real-Destination address one of most important part of header in every packet frame structure in our system design. This two bytes data (9-10) stores the information of the destination address the low data rate packet must reach. Based on our transmission scheme design our packets are construct with real-destination address of the final end of the transmission. Our design uses this field of frame structure to define which scheme about to run in the ongoing transmission. The real-destination address depending on transmission scheme version may vary. It can be either **0B** or **0C** for the schemes I and II respectively.

These 10 bytes represents the packet header and it is design in a way to help the nodes in our wireless network. They can identify which scheme is requested and implementing. The rest of the packet attributes are different from each other considering the scheme designs and working principle.

Sequence Number

Sequence number consists of 4 bytes and this bytes represents the sequence numbers generated on the node (in our case node A). At each transaction a chronologically implemented number will be attached to the data frame in prior to the payload to mark them as the unique packet. This marking helps the receiver to confirm the flow control of the incoming data. In case of data loss or connection exertion the destination can be unable to listen a packet being sent. The destination can request the packet retransmission to the sender node in such case. In another case, for image transmission sequence numbers are used to maintain the order of data and to avoid reception of duplicate payload, as it crucial for image transmission technique to misplace any payload while building an image from received payload bytes. Sequence numbers starts with in form of '0000' and continue to increment by one as '0001','0002','0003',..... etc.

Payload

Payload field of the data frame structure is the most sensitive and important part. The data generated in sender node are inserted to this part to send over wirelessly to the destination (in case of scheme II to the relay node to forward them) node. Upon receiving, the node reconstructs the data bytes from this payload part and response to it accordingly. Based on different types of services our payloads can contain randomly generated bytes, hexadecimal bytes for image, GPS informations and bytes representing ambient temperature of sender node.

CRC

CRC field in packet frame structures is used to calculate if the packet is received on the receiver nodes. Using CRC checking a node can confirm if the packets sent from the sender are received in the same exact condition on the other node. In the WSN loosing a small packet or a bit from a packet is crucial for information reconstruction. This missing bit/byte/packet can occur due to the conditions such as channel propagation, processing delay and etc. To ensure the received low data rate packet is in pristine condition while reconstructing, requires some sort of data bit error check function to initiate. CRC field is represented with 7 bytes. In our system design we use Python to implement CRC checking procedure in our application code for transmission schemes design. The process of CRC checking is as follows:

- The sender calculates its CRC by using the following code on Python:

```
1     import random
2     import binascii
3     .......
4     .......
5     .......
6     def CRC_calculate(payload):
7         k = binascii.crc32(payload)
8         h = abs(k) % (1<<32)
9         s = bin(abs(h))
10        crc="".join(value for index, value in enumerate(s[:26]) if index % 2
11        == index % 4 ==1)
12        return crc
```

- Here the payload is injected to the function CRC_calculate in byte format. CRC32 function of Python compute CRC-32, the 32-bit checksum of data, starting with an initial CRC. This is consistent with the ZIP file checksum. Since the algorithm is designed for use as a checksum algorithm, it is not suitable for use as a general hash algorithm. Note that, to generate the same numeric value across all Python versions and platforms can use CRC32 (data) 0xffffffff. If we are only using the checksum in packed binary format this is not necessary as the return value is the correct 32 bit binary representation regardless of sign. Our randomly generated payload bytes are signed which can give both positive and negative integer after calculating CRC32 value [9].

- We choose a specific pattern from the last list of the function before returning the CRC value. This pattern can be chosen as users preferences. The pattern we chose is to be a 7 byte long string starts with a 'b'. These 7 individual bytes are selected as randomly from the calculated binary conversion of our CRC32 value. CRC can be returned as e.g. 'b101001', 'b110110', 'b001100', etc.

- This 7 bytes is then included at the end of the data frame and sent via the 868 MHz enabled radio module (Connect2Pi) to another node (receiving node).

- The receiving node is then extract the payload from the frame and calculate the CRC on its own using the exact same function.

- If the CRC pattern calculated on receiving node matches the CRC extracted from the received packet then the receiving node will accept it as an uncorrupted packet. And it will send an ACK to sender.

- If the CRCs does not match with each other the receiver will discard the corrupted packet and send an NACK with the received sequence number by requesting a retransmission to the sender.

ACK

ACK is the shortening for acknowledgement. ACK consists of 3 bytes and usually define as '111'. ACK is sent to the sender to acknowledge the successful reception of the data frame from the receiver and also for the connection confirmation period of the schemes. Fig. 4.1 represents the frame structure of the ACK packets for responding to a successful transaction. We attach the last known sequence number received by the receiver in order to guarantee the transmission is in sequential manner. An ACK packet is 17 bytes in size in total.

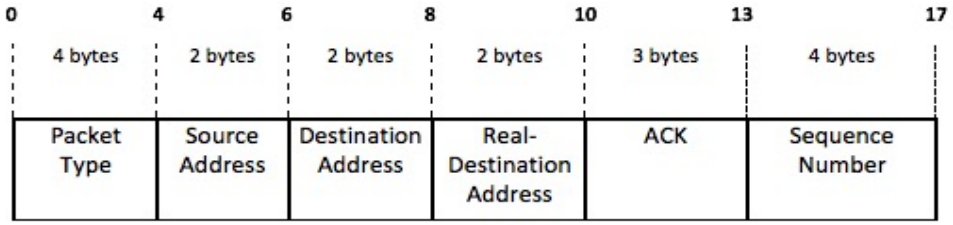


Figure 4.1: Frame Structure: ACK

NACK

NACK is also represented as 3 bytes and written as '000' in our Python code for transmission scheme design. NACK is sent when there is a mishaps happened such as CRC error or packet loss. NACK also included on the outgoing low data rate packets when receiver cannot hear or receive the packets from sender in specified period of time. In Fig. 4.2 we show the NACK frame structure with the sequence number included to it. The sequence number is in here to ask for the sender for retransmission in case the packet is corrupted or connection cut off. NACK packets are similar in size as ACK data packets (17 bytes).

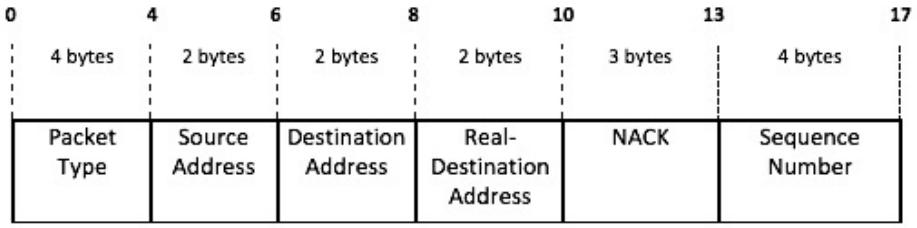


Figure 4.2: Frame Structure: NACK

The retransmission process in transmission scheme III is different from scheme I and II since the NACK request are not sent in same manner in scheme III. In the first two schemes the retransmission request are sent while the transaction is ongoing. The sender retransmits the requested packets (marked by the sequence number) in the same transaction period of fresh packet transmission. But in scheme III, we develop the process such a way that the entire transmission of fresh packets will occur without any interruption from the destination until the sender is done with the phrase one. When the receiver sense that sender node is done with the first phrase it activates phrase two and checks if there is any packet was missing during phrase one. If there is any, receiver then request for retransmit that specific packet with a NACK frame and upon receiving the NACK frame the sender will start sending the packets until there is no more packets are requested to retransmit. The frame structures of all the NACK packets are same in structure in every schemes.

4.2 Types of Services

As we mentioned earlier, data bytes in our system design can be used to transfer specific information based on user's preferences. Developers can implement various types of services in their design and these services can influence the scheme design, because a transmission scheme is built around the factor that what will this types of service will be used for. We select four types of services to implement in our scheme designs. The following are the brief description of our implemented types of services.

4.2.1 Random bytes

In terms of random bytes services we start with by simply sending 159 bytes of randomly generated bytes. This bytes are constructed by using the following code in python:

```
1  import random
2  ........
3  payload = bytes((random.randint(1111,9999))) + format(0,'0155d')
4  ........
```

Here the **payload** variable stores a string of 159 bytes. We calculate the CRC and add it with the payload in addition to the packet's quality as well as we include a sequence number incremented by one in this random bytes frame. Ain our thesis, we implement this types of service as we researched on how the data packets are transmitted in between two nodes. These 159 bytes are different then each other for every iteration and unique for every single packets. Fig. 4.3 representing the frame structure of a single packet with random bytes inside. These packets are sent in every 1 second of interval when the node (in our case node A) receives an ACK packet. On the receiver side, the payload is extracted upon receiving the packet and print it out on the display after passing the CRC checking. The total size of the data packet frame for carrying random bytes is 180 bytes.

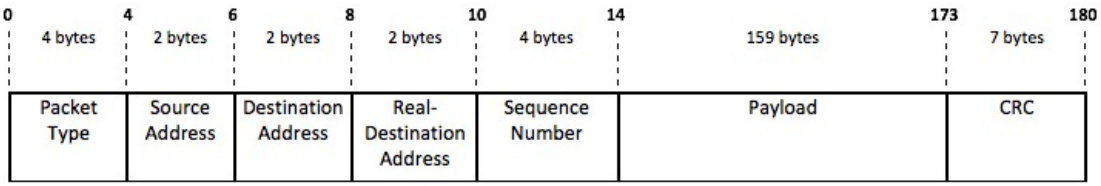
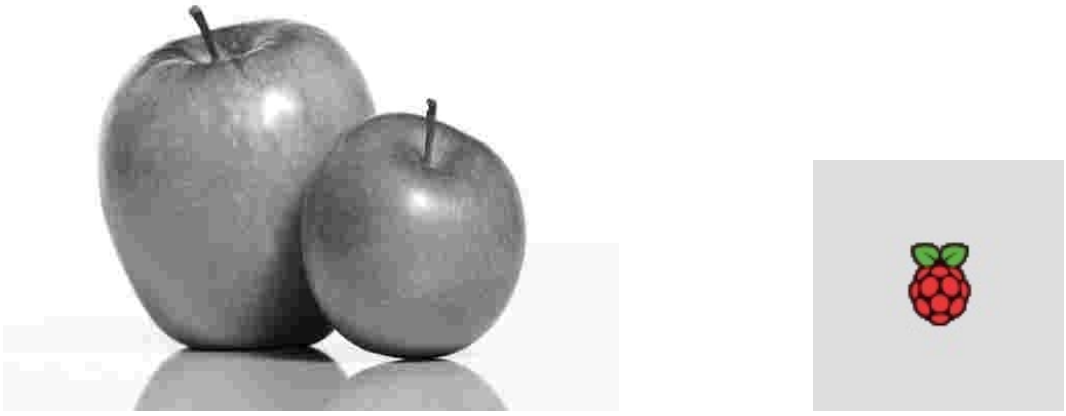


Figure 4.3: Frame Structure: Random Bytes and Image

4.2.2 Image

We implement this type of service in order to utilise the proper implementation of sequence number order in ongoing transmissions. We want to send an image using low data rate packets from sender node to the destination node via radio module operating on 868 MHz frequency band over long distance. The frame structure of the data packets in this type of service is similar to the



(a) Apple (JPEG) (b) Raspberry Pi Logo (PNG)

Figure 4.4: Images Transmitted as Type of Service

frame structure in random packet types of service. In this case, we select a default picture of an apple (JPEG, 7.8 KB) and alternatively a RPi logo (PNG, 1.5 KB) in our test scenarios. Fig. 4.4(a) and Fig. 4.4(b) shows the apple and RPi log respectively that have been transmitted over our 868 MHz band network. In order to send the images in form of packet we need to convert the JPEG file of Apple in (same process for png file of RPi logo) small byte size. We first read the file and convert it into string of readable format. This string of bytes comes in similar to this form:

```
‘‘\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x00H\x00H\x00\x00\xff\xe1\r!http://
ns.adobe.com/xap/1.0/\x00<?xpacket begin=\’\xef\xbb\xbf\’ id=\’W5M0MpCehiHzreSzNT
czkc9d\’?>\r\n<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="Image::ExifTool 8.50">
\r\n\t<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">\r\n\t\t<r
df:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/">\r\n\t\t<
/rdf:Description>\r\n\t\t<rdf:Description rdf:about="" xmlns:xmpRights="http://ns.
adobe.com/xap/1.0/rights/">\r\n\t\t\t<xmpRights:Marked>True</xmpRights:Marked>\r
\n\t\t\t<xmpRights:WebStatement>true</xmpRights:WebStatement>\r\n\t\t</rdf:Descri
ption>\r\n\t</rdf:RDF>\r\n</x:xmpmeta>\r\n.........’’
```

This string of data bytes can be sent over wireless medium but upon receiving on the receiver end the radio module reading the data start to take the ’\x’s, ’\n’s and ’\r’s as escape commands and initiate them accordingly. For Python those string values are not valid UTF-8. To solve this issue with change our above values to printable hexadecimals. We achieve it by importing the **binascii** library in Python and converting every byte of data into the corresponding 2 digit hex representation. The following code will change the unprintable ascii code to printable hexadecimals:

```
1  import binascii
2  ........
3  ........
4  f = open('location of the apple.jpg','rb')
5  byte_read_image = f.read()
6  ........
7  payload = binascii.hexlify(byte_read_image)
8  .........
9  f.close()
10 .........
```

```
11  .........
```

After the conversion of the string it transforms into similar to this:

```
''ffd8ffe000104a4649460001010100480048000ffe10d21687474703a2f2f6e732e61646f6265
2e636f6d2f7861702f312e302f003c3f787061636b657420626567696e3d27efbbbf272069643d27
57354d304d7043656869487a7265537a4e54637a6b633964273f3e0d0a3c783a786d706d65746120
786d6c6e733a783d2261646f62653a6e733a6d6574612f2220783a786d70746b3d22496d6167653a
3a45786966546f6f6c20382e3530223e0d0a093c7264663a52444620786d6c6e733a7264663d2268
7474703a2f2f7777772e77332e6f72672f313939392f30322f32322d7264662d73796e7461782d6e
7323223e0d0a09093c7264663a4465736372697074696f6e207264663a61626f75743d222220786d
6c6e733a64633d22687474703a2f2f7075726c2e6f72672f64632f656c656d656e74732f312e312f
223e0d0a09093c2f7264663a4465736372697074696f6e3e0d0a09093c7264663a44657363726970
74696f6e207264663a61626f75743d222220786d6c6e733a786d705269676874733d22687474703a
2f2f6e732e61646f62652e636f6d2f7861702f312e302f7269676874732f223e0d0a0909093c786d
705269676874733a4d6172........''
```

Since while converting the hexadecimal converts each bytes to 2 digits of bytes to represent the specific byte, the amount of the string becomes double the length of size. For example after reading the apple.jpg file we collected a string of 7841 bytes, but after conversion of that string our final string of bytes became 15682 bytes. The bytes are then inserted in the packets in 159 bytes per packets and added a sequence number to each packets. On the destination node the payload is extracted and converted back to the original form of string for the image. The following code helps us to achieve such conversion on destination node:

```
1  import binascii
2  ........
3  ........
4  hexadecimal = binascii.a2b_hex(payload_recv)
5  nf = open('apple_hexa.jpg','wb')
6  nf.write(hexadecimal)
7  nf.close()
8  ........
9  ........
```

We can then write the string of bytes into a file and create a file by combining them together on the receiver end. The sequence number integrated in the packet frames are used to put the strings in order and avoid duplicate packets as it is crucial for any file to have order of bytes to reduce corrupted file transfer. Fig. 4.3 also represents the frame structure of the packets that carries the payloads of the images. The total packet size for this types of service is also 180 bytes.

On the receiver side the packet is then gathered together and extract the bytes for image using the following codes. The picture is generated after joining the strings of data together.

```
1  import struct,binascii
2  ........
3  f = open('temp_image.txt','wb')
4  ........
5       ........
6       if seq_num_recv not in seq_num_recv_array:
```

```
            ........
            seq_num_recv_array.insert(n,seq_num_recv)
            seq_num_recv_array = seq_num_recv_array[-n:]
            payload_recv_array.insert(n,payload_recv)
            payload_recv_array = payload_recv_array[-n:]
            f.write(payload_recv)
            pass
        ........
    ........
        ........
        f.close()
        rf = open('temp_image.txt','rb')
        byte = rf.read()
        try:
            ........
            hexadecimal = binascii.a2b_hex(byte)
            nf = open('apple_hexa.jpg','wb')
            nf.write(hexadecimal)
            nf.close()
            rf.close()
            ........
        except TypeError:
            ........
```

For transmission scheme III, we also use this array sequence number to check if any packet has been dropped during first phrase of transaction. The code to calculate the number of packet dropped during fresh packet transmission are calculated on receiver node using the following function of **missing_elements** and this fragment of codes:

```
from random import randrange
........
def missing_elements(L,start,end,missing_num):
  complete_list = range(start,end+1)
  count = 0
  input_index = 0
  for item in complete_list:
      if item != L[input_index]:
          v = int(item)
          count += 1
          return v
      else:
          input_index += 1
      if count > missing_num:
          break
........
NACK_seq_num_recv_array = []
payload_recv_array = []
........
    seq_num_recv = msg[10:14]
    payload_recv = msg[14:-7]
    ........
    if seq_num_recv not in seq_num_recv_array:
        ........
        NACK_seq_num_recv_array.insert(n, int(seq_num_recv))
        ........
        NACK_seq_num_recv_array.sort()
        seq_num_recv_array.insert(n,seq_num_recv)
        payload_recv_array.insert(n,payload_recv)
        new_list = zip(seq_num_recv_array,payload_recv_array)
        ........
........
```

```
33    ........
34    L = NACK_seq_num_recv_array
35    start = L[0]
36    end = L[-1]
37    try:
38        ........
39        new_seq_int = missing_elements(L,start,end,0)
40        t.sleep(3)
41        missing_seq_num = format(new_seq_int, '04d')
42        input = str(packet_type[3] + src_addr + dst_addr + real_dst_addr + NACK +
              missing_seq_num)
43        ser.write(input)
44        pass
45    except ValueError, TypeError:
46        ........
47        new_list.sort()
48        t.sleep(0.005)
49        payload_sorted = [payload_recv_array for seq_num_recv_array,
              payload_recv_array in new_list]
50        while j < len(payload_sorted):
51            f.write(payload_sorted[j])
52            j = j + 1
53            pass
54            if j == len(payload_sorted):
55                f.close()
56                rf = open('temp_image.txt','rb')
57                byte = rf.read()
58                ........
59                try:
60                    ........
61                    hexadecimal = binascii.a2b_hex(byte)
62                    nf = open('apple_hexa.jpg','wb')
63                    nf.write(hexadecimal)
64                    nf.close()
65                    rf.close()
66                    ........
67                    pass
68                ........
69                pass
```

The above code runs the missing packets checking procedure and return the packets have been dropped during first phrase and request for retransmission to sender until all the packets are received correctly in the second phrase. Then it creates the picture using those bytes which is similar technique in every schemes.

4.2.3 GPS co-ordinates

In this types of service, we send GPS information that in form of data bytes to the receiver. This GPS information we collect from the Adafruit ultimate GPS breakout v.3 module that is attached to our node A. To achieve this we collect GPS information such as time, latitude, longitude, altitude and speed from this module. We use the following piece of code to collect these data:

```
1    import gps
2    ........
3    ........
4    session = gps.gps("localhost", "2947")
5    session.stream(gps.WATCH_ENABLE | gps.WATCH_NEWSTYLE)
6    ........
7    ........
```

```
8   report = session.next()
9   if report['class'] == 'TPV':
10      if hasattr(report, 'lat'):
11          time = str(report.time)
12          latitude = str(report.lat)
13          longitude = str(report.lon)
14          altitude = str(report.alt)
15          speed =  str(report.speed)
16          ........
17          payload = time + latitude[:6] + longitude[:6] + altitude + speed[:4]
18          ........
19      pass
20   ........
21   session = None
```

The above code retrieve data from GPS module and import the requested information into the string of payload. The module listens on port 2947 (gpsd) of localhost on RPi and gather GPS data. This payload is 44 bytes in total and sent over our ERA-Connect2Pi radio module to the receiver end by attaching the additional informations of the frame (packet type, source , destination and real-destination addresses, sequence number and CRC value). The receiver receives the packet with GPS information and extract the data in reverse order by defining which part of the payload defines what attributes of GPS informations. The size of a single packet carrying GPS information is 65 bytes in total which is illustrated in Fig. 4.5.

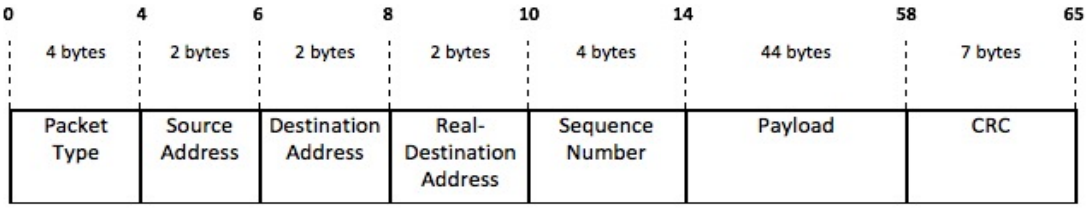


Figure 4.5: Frame Structure: GPS

4.2.4 Temperature

Temperature service in our project is bit tricky to achieve as we are not using any specific temperature sensor in our system design. From our brief description of Connect2Pi (Sec. 2.2.1) we know that is has a built in temperature sensor which can measure the ambient temperature around the radio module's environment and can give the values in hex as response to ER commands. To achieve that we have to send the command "ER_CMD#T7" followed by "ACK" command to the Connect2Pi module which returns the ambient temperature in 4 bytes. We define the functions using following code in our application:

```
1   ........
2   def listen(name,timeout):
3     listen_txt = ''
4     exitflag = 0
5     while True:
6         while ser.inWaiting()>0:
7             t.sleep(0.005)
```

```
8               listen_txt += ser.read(1)
9           if listen_txt !='':
10              temp_txt = listen_txt[-7:-3]
11              return temp_txt
12              listen_txt = ''
13          if (exitflag == 1):
14              exitflag = 0
15              thread.exit()
16  ........
17  listen_txt = ''
18  input1 = 'ER_CMD#T7'
19  ser.write(input1)
20  t.sleep(0.005)
21  input2 = 'ACK'
22  ser.write(input2)
23  temperature = listen("Listen",0.05)
24  payload = temperature
25  ........
```

After sending the commands the radio echoes the inserted commands and listen at the same time. This return value of temperature is then captured by the radio's antenna. The timing for the listening function is very important as it is possible that the radio cannot only here its own echo but also the other radio's at the same time. As a result the radio can return a value with error to the user. We measured the time intervals of the radio listening function and defined the exact time interval for radio to listen. After achieving the correct temperature from the radio we add this 4 bytes of data in the temperature data frame and send it to the receiver wirelessly. This packet is in total 25 bytes. Receiver receives the packet and accept it as valid packet if the CRC does not give any error. The temperature is now printed on destination node and an acknowledgement will be sent to the sender nodes. Fig. 4.6 shows the frame structure of the data packets containing temperature in form of payload.

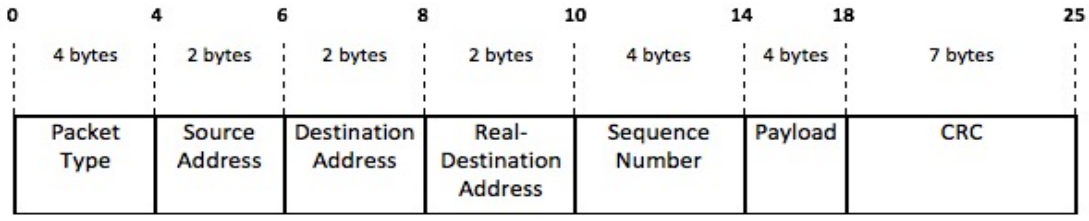


Figure 4.6: Frame Structure: Temperature

Note: Alternation only on Scheme II

All the frame structures are similar in structure for every schemes implemented except there is a slight difference ink scheme II. In this scheme, we add up some number of bytes to make every packet size to 180 bytes. This is done because we want make the same size of packet size for the entire network. It is the similar concept of bit padding or bit stuffing in the networking. When we send some small data to a network it is important that the radio on the receiver (in our case both relay and destination node) need to have a standard amount of size so all data sent and received can be transfer via network. This bit padding is done to make sure the network equipments configured to interpret or forward information that matches a particular length handles the packet correctly

and to keep the appearance of a continuous stream of data flowing since some telecommunications equipment require a constant flow of data. We also put the padding on the packets so that we do not need to reconfigure our radio settings to read the amount of byte at each interval which is different for different packet size. Although it is ß a problem for single hop case in first version of the scheme, this becomes an issue when we implement the same packet frame structures for scheme II. As in this scheme we there is three different nodes communicate with each other and we can only configure one radio setting in each node for each transmission. To recover from this issue we add necessary extra bytes to the frame structures which are smaller than 180 bytes in size, i.e., every type of frame except the payload with random bytes and image services. This extra bytes are added to the existing frames with some mock bytes (basically 0s). For example we add 163 extra bytes to the ACK and NACK frames where with the GPS frames we add 115 packets. This makes every packets to 180 bytes in total. Now this helps all the nodes in our entire network to be exact same configuration for radio to read on same amount of bytes per transactions. The data receiving in every nodes are easily being extracted by discarding those mock payloads first and then work on the bytes according the working procedure (similar to the scheme I and II). Rest of the frame structure are same as it is in other schemes. Note that, this padding is only implementing in scheme II scenarios and this do not affect the functionality of the protocol in any given situation.

4.3 Design and Implementation: Transmission Scheme I

In our first transmission scheme, we aim to connect two nodes and transfer data between each other. According to this scheme we will have sender system to send data bytes wirelessly to the receiver node. Depending on the types of service the packets' frame structure will be constructed differently than each other as we discussed in Sec. 4.2. In the following sections, we introduce the readers about the main features and procedure of our transmission scheme I.

4.3.1 Scheme I overview

Our transmission scheme I consists of two nodes. We consider node A as the sending node and node B as the destination node in this design. According to the design principal both of the nodes contains its own source addresses in form of **0A** and **0B** respectively. The destination addresses are **0B** for node A and **0A** for node B. Our application written on Python language defines this scheme implementation for every scenarios. In this scheme, we configure our channel to read and write to the RPi's serial port of the sender node A and B according to the Table 4.1:

Table 4.1: Channel Settings of the Nodes in Scheme I

Settings	**Transmitter**	**Receiver**
Node's Address	0A	0B
Listening Serial Port (Radio)	/dev/ttyUSB0	/dev/ttyUSB0
Baudrate	19200	19200
Parity	None	None
Stop Bits	1 bit	1 bit
Byte Size	8 bits	8 bits
Write Timeout	2 sec	0 sec
Read Timeout	6 sec	0.5 sec

4.3.2 Scheme I features

In this scheme, we concentrate on the stability of the connection between two node while transmission in progress. This scheme features also the reduction of packet drop in receiver end as it is important factor of any wireless network to achieve a fully functional scheme of transmission to achieve that. Our first scheme sends and retrieve the packet successfully by carefully balancing the read and write timeout of the channel configuration with a decent amount of distance between the two nodes.

4.3.3 Scheme I handshake procedure

Our design starts by taking the input choice of real-destination node in sender node A which is 0B in this version of scheme. This real-destination defines which scheme will be activated for our following transmission. 0B defines that transmission scheme I has been chosen and it will automatically select the node B as the destination node for node A. Node A then will create the **Connection Invite** packet and send it via serial port defined by for the RPi to request connection with node B. This activates the serial port for the radio and send a packet consists of 10 bytes constructed with the packet type ('0000'), source address of the A, destination address and real-destination address. The invitation packet then travel wirelessly and reach to the receiving node B.

We configure our receiving radio of node B to read and write in faster manner than of node A since it is important for the node B to quickly response to the received packet sent from A. Node B stays in listening mode until it reads an invitation packet from the sender. Receiver then checks what type of the packet it is and if that packet has the destination address which matches its own source address. B also checks which scheme version is requested from the real-destination address. As it is the first scheme of the embedded system design the B will select itself as the destination address and send the **Receiver Ready** packets to the node A (if B is not busy). The data frame consists of packet type ('0001'), source address of the B, destination address of the A (received as the requested source address from the invitation packet) and the real-destination address which defines the node B's acknowledgement of the scheme version selection. This ready packet is also 10 bytes in total size. This packet then transmits via the serial port of B and sent towards node A.

While B is listening and configuring the ready packets node A will sleep for pre-defined time interval. When A receives the ready packet from the receiver, it checks the packet type of the incoming packet. If the packet type is receiver ready packet it continues to check if this following packet from the same source it's selected destination for connection invitation packet. If it is true that the receiver ready packet is from the node B it consider the receiver B is ready to receive the data generated for selected types of services.

4.3.4 Scheme I data exchange procedure

In node A, the types of services now need to be selected. After selection and confirmation of the types of service node A activates the corresponding procedure to send the data regarding the type of service has been chosen. A **Connection On** packet sent towards node B (destination) to acknowledge that the sender node (A) is now ready to transfer the data packets. This connection on packet (10 bytes in size) consists of the packet type (according to the types of services), source address, destination address and the real-destination address to define the ongoing scheme version.

Node B receive this packet and checks what type packet is received. Based on the selected service the packet type will vary. Upon checking node B now know what type of service is selected for transmitting from the sender A. B will check if the received packet is from node A and if this packet has the destination address of the node B. If all positive then the receiver B now know that A is ready to send data depending on types of service chosen and also initiate the procedure to receive the same type of service and generates a packet to send to A over wireless network. The packet is constructed with packet type ('0010') (packet type for ACK), source address (B), destination address (A) and an ACK ('111'). The packet is then sent towards sender A to acknowledge that receiver know what type of service has been chosen and is now ready to receive them from sender. This ACK packet is 17 bytes in size as we discussed about the frame structure in Sec. 4.1.

Now the packet will be received by the sender node and it (sender) checks if the packet is from the destination node and if it confirms the types of service along with this scheme have been chosen by destination. If this confirms, the sender checks if the packet contains an ACK from the destination node. If there is an ACK it starts to send the packets which is different in construction and size for different types of services. We discussed about frame structures of the different services in Sec. 4.2. After sending the data frame with payload included the sender waits for the next ACK packet from the receiver. Sender also saves the following payload in its buffer with the corresponding packet sequence number. This is because in case of there is a packet drop or CRC error on receiver side and a request for retransmission has been made. If there is no packet containing ACK during the defined interval time for listening on radio the sender sends the same packet again and continue to do that until the an ACK packet is received from the receiver node (B). If any other packets receiving from different sources while listening to the serial port it will be then discarded and set back to listening mode for the next ACK or NACK from designated destination.

When the packet with payload received at destination the node checks if this packet type matches the selected type of service. It also checks and confirms that this packet is from the source address that the destination has established the connection earlier. This ensures the stability of the connection with the sender node. If the source address does not match with the appointed source address then node B continues to send the ACK packets towards the designated sender's source. If the source address is from the same source node then the receiver receives the packet as valid and extract information from the packet according to the types of service. Destination also calculates the CRC value generated by using the received payload and match with the received CRC from the sender. CRC values are unique for each payload and it defines the quality of the received packets. Packets with the CRC error are considered to be a corrupted packet and a NACK will be sent to the sender by requesting the retransmission of the same packet. Destination node then request the retransmission by including the sequence number received with that packet. This NACK packet is constructed as similar to the ACK packet except instead of ACK in the ACK field a NACK ('000') will be included. Also the packet type will be replaced by the packet type of NACK ('0011'). Upon sending the NACK packet the receiver then waits for the retransmission from the sender (node A). If the received CRC matches accurately with the calculated CRC (on node B) the receiver now prints out the information and sends an ACK packet to the sender in order to acknowledge that the transmitted packet has been received successfully and this transaction is successful. Receiver then send an ACK towards sender (node A) including the sequence number received and source, destination and real-destination address. The ACK field will contain the ACK bytes in this data frame.

If the transaction was not successful the sender will receive a NACK data packet and upon receiving it will then check its buffer (of all the recent sent packets) and finds the corresponding payload

from the list initiated with that particular sequence number. Sender will then calculate a new CRC value (which is same in our case as CRC value is quite unique) and retransmit the packet again towards the receiver. In case the transmission is successful and when sender receives and acknowledgement packet from receiver it construct its next packet with payload to send. This new packet will consists of a new sequence number (incremented by 1 number), related next batch of payload and a calculated new CRC value for this new packet. It will then send the packet via 868 MHz enabled radio module to the designated destination. The transaction continues to process until a connection get lost or system gets exhausted.

On the receiver side it continues to listen until a new packet arrives from sender with a new batch of payload. Between its pre-defined time interval for serial port to listens for any incoming packets all the other packets received from different sources other than connected node (node A) will be discarded. If there is no new packets from source for certain period of time the destination node sends a NACK packet requesting new packet and start to listen via the radio module (Connect2Pi) on the serial port again. It continues to send a NACK packets to the sender after each interval until new packets arrive. If no new packets are arrived after 4 or 7 times of NACK packets request the receiver will consider the connection has been disconnected or the sender node has been exhausted. Then the receiver terminates the ongoing transmission process and initiates the radio in preliminary state. This means the radio resets to first state of the system where it is listening to the new connection only. We define the NACK request counter as ether 4 or 7 times. Each retransmission request iteration is chosen randomly either 4 times or 7 times. If a new batch of payload is received during this time then the retransmission counter will be reset to null again until next counter for NACK request is initiated.

Sender node also initiate the retransmission counter randomly either 4 or 7 times. This means the sender will send the packets and retransmit them for 4 times or 7 times during the pre-defined serial port listening time interval until a ACK or NACK request is received from the destination node. When the counter for retransmission hits the limit it will then activate its reset state of scheme where the sender needs to re-establish a new connection with receiver again.

The process then starts from the beginning of the state of sender and receiver nodes. Both nodes need to re-establish the new connection as the same manner as above method.

4.3.5 Scheme I flow diagram

Fig. 4.7 illustrates our transmission scheme I. The transmission continues until the transmission teardown occurs due to user request or the radio disconnection. In case of image transmission service the procedure will be terminated after all the bytes for an image file have been sent from sender node.

4.4 Design and Implementation : Transmission Scheme II

Our second transmission scheme comes with a new feature which is consider as a two-hop transmission. We use and connect three nodes in this scheme where one represents as sender node and other as the receiver node. We introduce the third node as a forwarding node to relay our incoming data from sender to the receiver node. The forwarding node will act as the one hop away from both sender and receiver. Sender only sends to forwarding node while transmitting and receiver only accept packets coming from the forwarder. Packets received other than these

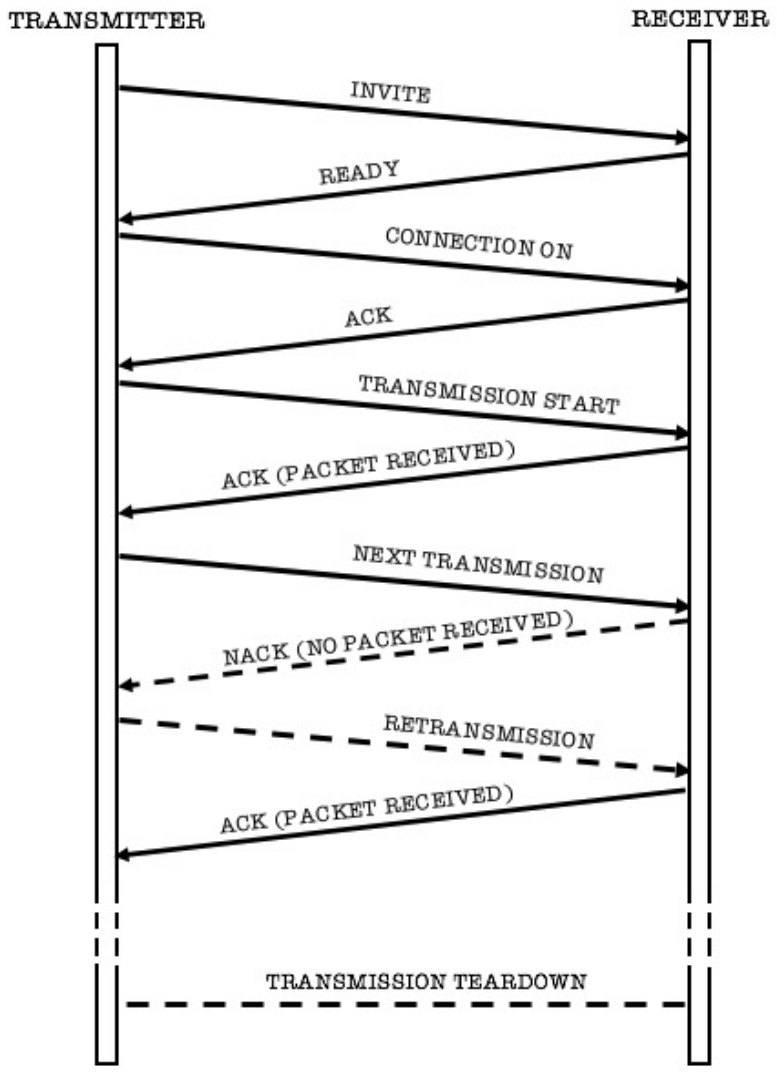


Figure 4.7: Flow Diagram: Transmission Scheme I

following sources will be discarded in each system.

4.4.1 Scheme II overview

In our scheme II design, node A is also the sending node and node C is considered as the receiving node. The node B will be considered as the forwarding / relaying node in this design. Similar to the first scheme, all three nodes (sender, relay and receiver) have their own source address which is **0A**, **0B** and **0C** respectively. We implement our design using the Python language which helps also to collect the required test results from the real-life test scenarios. The channels are configured similar way in this version of scheme except the timeout value for serial port reading and writing is different than of each other since we have to balance the radio listening timing interval for all three radios. The channel configuration of the RPi's serial port of the node A,B and C are as follows:

- Listening Serial Port (Radio): /dev/ttyUSB0

- Baudrate: 19200
- Parity: None
- Stop Bits: 1 bits
- Byte Size: 8 bits

We removed the write timeouts for all the channels. The read timeouts (also can be referred as message service interval in our case) of the channels are configured using the following Table 4.2:

Table 4.2: Message Service Intervals in Scheme II

Nodes	Node Address	Random Bytes	Image	GPS	Temperature
Transmitter	0A	1 sec	4 sec	2 sec	4 sec
Forwarder	0B	1 sec	1.5 sec	1 sec	1.5 sec
Receiver	0C	1 sec	2 sec	2 sec	3 sec

It is noticeable that we configuring our relaying node to listen to its serial port is slightly faster than of node A or C. The reason behind it that we have to confirm that our forwarding node is always ready to relay the incoming packets from sender and forward them to receiver as it will forward the data frames sent from receiver towards transmitter node, in case of successful transaction of data packets or if there is request for retransmission.

4.4.2 Scheme II features

This protocol or scheme helps us achieve a longer distance between the sender and receiver node by including a new node in the middle as a forwarding node. This forwarding node relay the packets from receiver to the receiver end using the two-hop transmission method to cover a reasonable amount of distance. Our goal is to implement a working solution and a transmission scheme which can achieve a good range of distance between the nodes by controlling the flow of transmission of the packets. Transferring low data rate packets over long distance has a complex mechanism and we try to accomplish this technique by designing and implementing this transmission scheme.

4.4.3 Scheme II handshake procedure

In this scheme the handshake part is quite different from the transmission scheme I. The process starts by taking the input choice of real-destination address in sender node A. This real-destination address is 0C in this scheme II design as this ensures the two-hop transmission method. The destination address for node A is still 0B but as for this scheme the real-destination is changed to 0C. Similarly the receiving node C also has 0B as its destination address. For relaying node B there is a two alternative of destination address depending on the packets it is receiving and the steps it has to take in response to that received packet. So for the relay node, 0A and 0C are both the destination addresses table which alternates during the transaction period. After selection of scheme II, node A creates the **Connection Invite** packet and send it via serial port configured on the RPi. This packet is configured similar to the invite packet we discussed on Sec. 4.1 except this time the real-destination address will be selected as 0C. This invite packet is constructed with 10 bytes where first two bytes is the packet type for invitation ('0000'), bytes 5-6 are the source address of A (0A), bytes 7-8 will be the destination address of the node B (0B) and finally the last two bytes will contain the real-destination address which is C's source address in this case (0C).

This packet is then being sent towards node B. Node B stays in listening mode until it receives an invitation packet from node A. Similar to first scheme, relay node B checks if this a packet type indication to the invitation packet. Node B also confirms that this packet is only receiving from A by checking its address table. We have designed the transmission in such a way that B will only response to node A's invitation packet as it has no intention to take any request of invitation from node C. This confirms one way flow of packets towards the real-destination (0C). Relay node also need to confirm that the destination address of the invitation packet is addressed as 0B which confirms invitation process is requested to node B only. Upon confirmation the relay node checks the real-destination address included in this invitation packet. As of scheme II of transmission this real-destination address is selected as 0C from sender node. By examining this address field relay node confirms that scheme II is in progress and select itself as the forwarding node. After transforming, as a forwarding node B then sends an invitation packet towards receiver node C as the transmitter node wish to send data packets to node C in two-hop transmission manner using node B as the forwarding node. The invitation process starts with creating a new invitation packet to establish a connection with B on behalf of node A. This packet is almost similar to the invitation packet sent from node A towards B except the node B replace the source address of this packet with its own source address (0B) and destination address of the packet as receiver C's address (0C). This invitation packet sent from the relay node is now then sent towards final destination (node C) wirelessly.

As the real-destination node receives this packet it checks if it is an invitation packet. If it is then it checks packet's source address from the source address field and confirms it is sent from forwarding node B. Node C also confirms that this packet is sent towards its own radio by checking the destination address. Upon receiving and confirming that this is an invitation for connection establishment from B, node C configure itself a **Receiver Ready** packet to send it towards the forwarding node. Note that node C as a receiver of the future packets does not consider the real sender of the packets in this scenario. It only considers the packets incoming from node B and consider that node as its destination address. The ready packet is constructed with packet type for ready packets ('0001') followed by source address of the node C (0C), destination address (forwarder 0B) and the real-destination (0C) to define node C's confirmation of understanding that the scheme II is implemented now in this transaction. This data frame will be then send towards the radio of forwarding node.

The radio in relay node B is configured to listen in faster manner than the other two nodes. The ready packet sent from receiver is received to relay node. It checks that if this is a ready packet type and if the source address of the packet is the receiver. This confirms that receiver is ready to connect with the relay node and can now receive the forwarded packet from source node. Relay node now also creates a **Receiver Ready** packet and replace the source address to relay node's source address and destination address for this packet to sender's source address. The real-destination address stays the same as default to mark that transmission scheme II is ongoing. The new altered ready packet then travel wirelessly to the sender node. Sender receives the packet and checks if the relay is ready by checking the packet's source address and if this address is sent for transmitter (by checking destination address of the packet). This ready packet has the real-destination address as receiver's address which is different from the ready packet received in transmitter node in scheme I. By checking the real-destination address the sender node now know that the scheme II is ongoing and the relaying and receiving node also aware of the scheme version of the ongoing transaction. This concludes the handshake process for scheme II of our system design.

4.4.4 Scheme II data exchange procedure

At this point all our implemented system are running and ready for the transaction to proceed. The procedure for sending packets are almost same as the scheme I, but here we use the relay node to forward data from sender to receiver. Our goal is to achieve a longer area of distance by using the two-hop transmission technique in our transmission scheme II. The type of service need to be selected at this point in transmitter. After selection of the service type sender node executes the procedure that will proceed with the relevant steps to start sending the service. Similarly (to the first scheme) a **Connection On** packet will be constructed on transmitter A and sent to relay B to confirm that a type of service has been selected and it is ready to send the data frames with payload. The packet size is also same as before 10 bytes in total size and includes the packet type (packet type for service type), source address of A, destination address and the real-destination address (0C is this case).

Forwarding / relay node B receives this packet and checks the packet type (based on types of services chosen) and source address it came from. If the source address in the connection on acknowledge packet matches the source address of A then it continues to check for the destination address is for node B. If all is positive then forwarder choose the procedure for forwarding packets for ongoing service type by confirming it with the packet type included in connection on data frame. Node B now know what type of service is selected and send a similar packet (Connection On) towards receiver node C. The packet has the same packet type which indicates the types of service and source address of the forwarder. Destination address will be changed to node C's source address and real-destination address will stay as default (0C). Relay B now waits and listen for the ACK from receiver C. Receiving system C also receives a packet (Connection On) from relay node (B) and decode it by checking the source and destination address included in this packet. Node C now select type of service receiving procedure by checking the packet type received. It then transmits an ACK packet towards node B to acknowledge its confirmation of choosing the right type of service. Node B receives the packet from node C and checks if this is from the receiver node C by checking the address fields and all it checks if this packet is for its own address. When the relay B finds an ACK in this packet it will then also send an ACK packet to transmitter A by constructing it using packet type for ACK, source address of the node B, destination address of the node A, the real-destination address (0C) and finally an ACK.

Sender A now wait and listen to the serial port via radio module for incoming packets from node B with an ACK included. Upon receiving transmitter A now checks the incoming ACK packet received from node B. Node A checks and confirms that forwarding node is ready to relay the packet towards receiving node C for the sender node A. This packet is 17 bytes in size and A decodes the packet exactly the same manner as it does in scheme I. Now every node in the entire network knows which transmission scheme will going to be executed on the systems and what type of services are going to be sent from A to C via B node. Inside this frame transmitter A checks if there is any ACK bytes ('111') in there and if there is, it starts to construct the frames with payloads in it. The receiver confirms the packet is from the forwarding node (from source address) and it is sent towards itself (from destination address). Following this technique the transmitter can confirm that the relay node is now ready to receive the packets with payloads (depends on types of service). Node A construct the packet containing the packet type for service type was chosen, source and destination address as 0A and 0B respectively and real-destination as 0C as this indicates the relaying node needs to relay this packet to receiver node C. The packet also include a sequence number indicating the order of packets, the payloads (size based on types of service) and finally the calculated CRC value of that payload. The frame will be saved on a list in buffer for future retransmission purposes. Any other packet other than ACK packets from

B will be discarded in transmitter node at this point of transmission scheme procedure.

This fresh packet with the payload is now sent towards the node B. When B receives this packet it checks the packet type is the same as the selected types of services. If it does not match it will drop the packet and start listening to port again. If it matches the service type the packet continues in the forwarding procedure. Node B need to check the incoming packets are only coming from transmitter and it is sent for node B to forward towards real-destination C. After positive confirmation node B calculates the CRC value from payload received and matches it with the extracted CRC value received from sender A. If it is a negative match the calculation will give a CRC error and a request for retransmission will be sent to transmitter using the NACK frame and the sequence number it received. If the CRC value matches then this packet will be considered as the successful packet transmission with no corrupted packet. Relaying node then replaces the source address with its own (0B) and replace the destination address of the packet to the receivers (0C). Before sending the frame to the receiver via the radio the relay also saves the packet in its buffer, particularly the payload and its corresponding sequence number in lists. Node B forwards this packet and wait for next incoming packets from either A or C.

Receiving node C waits and listen via radio until a new packet is arrived from node B. Similarly it also checks if the packet has the same service type and if this packet is forwarded from node B as well as sent towards the receiver's address. If the values from those particular fields matches it continue with the receiving procedure. This procedure maintains the stable connectivity between the nodes. The CRC value will also be checked by the node C in case the freshly packet gets corrupted in its way from B to C. When the CRC value checks out as positive the payload is then prints out on receiver node and an ACK will be constructed and sent towards relay node. This ACK is similar to the previous ACK in construction which includes the latest sequence number received on node C.

When node B receives an ACK from receiver node towards itself (by confirming its packet type, source address, destination address and an ACK) it confirms that the relay of this packet with payload is successfully received by the node C. Node B then construct and send a ACK towards sender node A (by replacing the source and destination node accordingly) to confirm this transaction and to request a fresh packet. Similarly the sender node A receives this ACK packet from relay and confirms that last packet has been sent and reached to receiver successfully. Node A then proceed with the construction of new fresh packet with a new sequence number (incremented by 1) and a fresh payload with its corresponding CRC value.

The transaction process continues in same manner until a connection teardown occurs or the system in exhausted. The packets received to all the nodes with other destination addresses included except their own source addresses will be discarded.

If there is CRC error or receiver's radio cannot receive any packets from forwarder during its listening time interval, a NACK packet will be then sent towards relay node with a request to retransmit. Relay checks the list in its buffer by searching the payload sent with the help of its related sequence number. Every sequence number and CRC value is unique to a single payload, i.e., two packets cannot have the same sequence number and two payloads will never have same CRC value. When the relay finds the payload with the sequence number it construct and retransmit the requested packet towards receiver C until it receives an ACK from C. Receiver sends an ACK if the transaction is successful at this time. Node A awaits until the forwarding node is done with the transaction and sends an ACK towards it.

All the nodes has a setting where it will only try to initiate retransmission process for 4 or 7 times, which is exactly same process as we described in transmission scheme I. After the counter hits the retransmission limit the system will then activate the reset procedure and re-state into the original state of the system and wait for the new transmission scheme selection.

4.4.5 Scheme II flow diagram

The diagram (Fig. 4.8) shows us the transmission procedure of our designed protocol or transmission scheme II. The process continues until the connection need to be re-establish or the system gets exhausted.

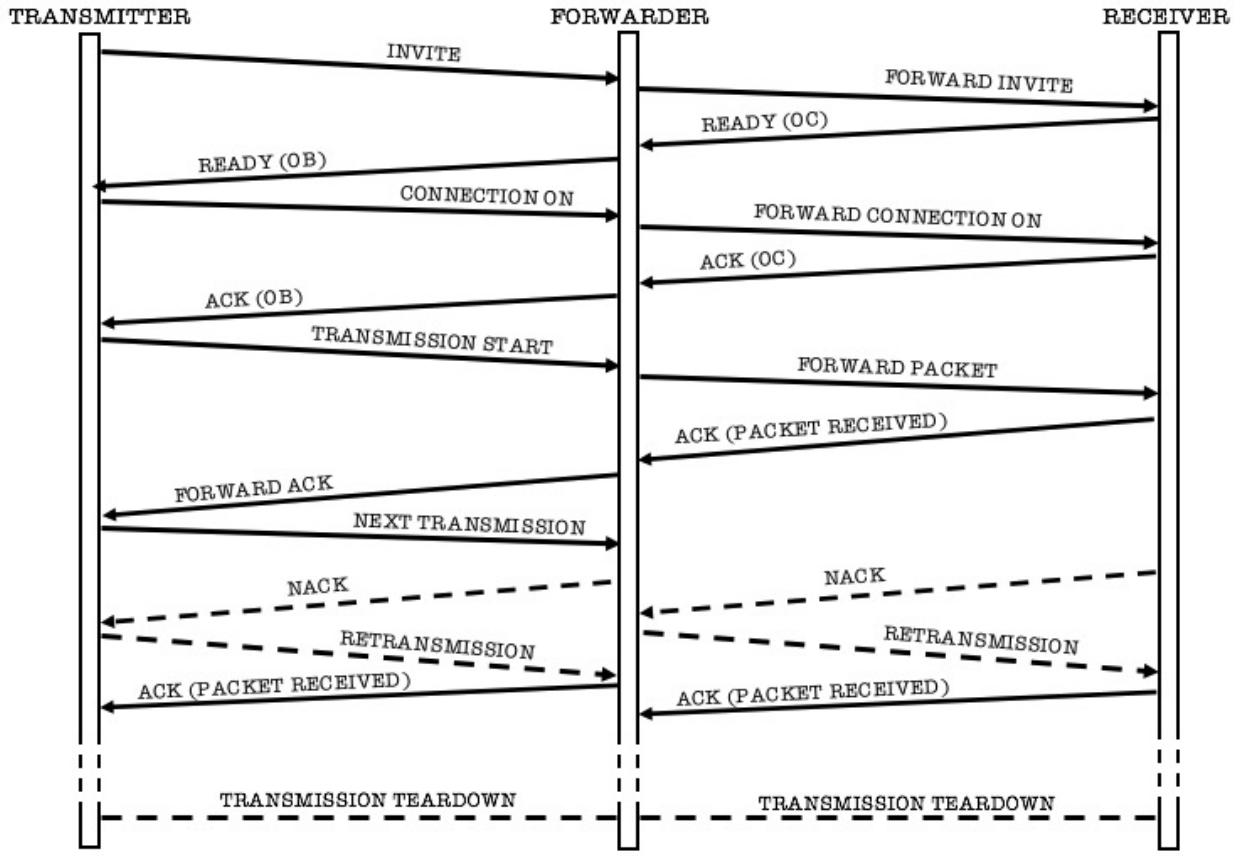


Figure 4.8: Flow Diagram: Transmission Scheme II

4.5 Design and Implementation : Transmission Scheme III

Our designed transmission scheme III has exactly the same procedures as the transmission scheme I. The only difference between these two schemes is while we are implementing the image transfer service type. As we mentioned earlier, we develop our transaction process of image transfer service between a sender node and a receiver node and implemented in such a way that we can request all the retransmission after the first phrase of transmission process is done. In the second transmission phrase only the retransmission will occur until all the packets generated on first phrase are received by the receiver. We remove the ACK sent from receiving system upon receiving the data packets

with actual payloads in the first phrase. This made our system performance much faster in general as the sender is not depended on an acknowledgement while continuing with the transmission. We completely remove the channel read and write timeout interval in transmitter system to make it free of any limitations of sending file. The sending file procedure on the sender system completely depending on the senders processing time and queuing delay.

4.5.1 Scheme III overview

We tend to get a better solution for image file (or any large amount of file) with this scheme. Our aim is to implement a scheme which can improve the quality of our transmitted data in the transmission process. Like the first scheme, this scheme III is also implemented on two nodes where one node is acting as the sending node and the other node will be the destination node. Same way packets sent from sender will be received by destination. The sender node in this scheme is the node A and destination / receiver node is node B. The channel configuration of the node RPis are the exactly same as it is in scheme I. Here we only apply this scheme to one of our service type which is image service. The reason behind is that other services in our system design does not have an EOF and it is more important for any file transfer protocol to have a procedure that is able to completely transfer every single bytes while transferring from one node to another.

4.5.2 Scheme III features

It is crucial for any file if any of its data is missing on the way to the receiver while transmitting. This packet drop or data loss can happen for various reasons such as environmental or hardware issue. Not only the it is important to assure that every single bit of data is sent properly to the chosen destination but also the connectivity between the transmitter and receiver system need to be stable. If any of this data bits are missing on the receiver the file will not be constructed as in whole and it will be considered as corrupted file. The entire transaction will then be considered as the fail transaction for this reason. We modified our transmission scheme I and divide the transmission into two phrases. The first phrase is used when the sender start sending the file (JPEG file) to the destination byte by byte and continue to send until the whole file is transmitted (EOF). In the second phrase the receiver calculates and request for retransmission if there is any packet dropped during the first phrase. Although we are improving our transmission scheme III for image service only, other types services can easily be implemented to this scheme. Maintaining the quality of the transferred file is the main concern in this scheme design and implementation.

4.5.3 Scheme III handshake

As we mentioned earlier, the handshake procedure of this scheme proceeds exactly the same manner as of transmission scheme I. The sender request connection with an invitation packet with the destination and the destination sends a ready packet (if it is not busy) to the sender in response. Then the sending node (A) select the type of service and sends a connection on packet to the receiver. The destination node (B) reply back with an acknowledgement which starts the transmission process of the services.

4.5.4 Scheme III data exchange procedure

In this section we will only discuss our implementation of scheme III on image service as of we did not apply this scheme to other three services. As mentioned before, we divide the entire

transaction process in two phrases. The first phrase is similar to the transmission procedure of the first scheme, except we are removing the retransmission of the packets. We also remove the ACK response upon successfully receiving the packet on the destination node. The entire first phrase implemented as the transmitter keep sending every bytes related to the chosen image (Apple JPEG) in form of payload with the packet type, source and destination addresses, real-destination address (which is 0B in this single hop case), sequence number and the calculated CRC value. When the destination collects this packet it decodes the packet and checks for necessary attribute fields to confirm this packet is for the destination. Note that, in this case the destination node does not send any ACK packet to sender to confirm the successful data transfer. Rather it saves the payload and sequence number in its buffer and start waiting for the next packet with payloads. Sender node construct the next payload along with the new sequence number and CRC after each radio time interval. Sender always keeps a copy the sent packets in a list of array in its buffer and keep storing all the newer fresh packets it is sending. It is also necessary to note that the sender in this situation does not know if the packets are successfully received on the destination node or not. Destination continues to collect the data frames coming through its serial port while listening to it and the payloads received in each transaction are getting stored in its buffer with their corresponding sequence numbers. In case if there is no packet received on receiver the receiver start counting a timer for 4 to 7 times in 0.5 sec interval until it can hear the next packet on the radio. Like we mentioned, this time there is no NACK sent as we implemented in scheme II and III. Instead the receiver will wait in a limited time interval. The sender node continues to send the packets until it stops sending the packets with payload as it is done with sending the whole file or it reaches the EOF. When done, the sender then enter into listening mode and start listening on the serial port of the RPi continuously for the NACK packet (if there is any). This activates the second phrase of our transmission scheme III.

At the same time the sender node will change its radio modules data rate 19.2 Kbps to lower data rate 9.6 Kbps by using **ER_CMD#B2** command followed by **ACK** command. This techniques are discussed in details in Sec. 2.2.1. The reason we change the data rate to give the nodes to communicate and transfer data between each other reliably. We wanted to have this option included in our third scheme design as it is important for the radio to have a stable connectivity and faster transaction after the first phrase of communication is finished. This gives us the chance to implement the automatic radio data rate adaptation on the channel which in terms also is an improvement towards the coverage area for the radio module we used in our thesis.

The destination node on the other side uses the same technique as the sender change its radio data rate setting to 9.6 Kbps as well. This is necessary because the radios attached to the node systems cannot communicate to each other if they are not in the same data rate setting. Before transformation of the radio the destination then continues to wait for the given random time interval for the new fresh packet for several counts. This counter has a time limit which is 4 or 7 time, i.e., the node will try to listen through the radio module for next fresh packet for 4 or 7 times. After the counter is over the destination node then change its data rate settings of the radio to 9.6 Kbps as well. After the data rate has been changed, the destination checks on its buffer of received payloads where it can count and determine how many packets are received from sender to destination. As all the payloads came with their unique sequence number and the sequence numbers generated and attached in sender node cannot be repeated twice so the destination should receive the packets in sequence as well. The sequence numbers are chronological and it increments by 1 in each transmission as we discussed in 4.1, hence if there any sequence number is missing in the buffer list of receiver then it is confirmed that that particular packet has never received by the destination node B in the first phrase of transmission scheme III. This node then select the first packet missing in the list and sends a NACK packet to sender by including

the sequence number of that missing packet. The sender receives this NACK packet and dissect it to check if there is a NACK included in this packet. If there is a NACK it checks the last 4 bytes of this 17 bytes NACK frame as this particular field indicates the sequence number that missing packet destination has requested. After decoding this sequence number sender node checks on its buffer and find the corresponding payload to the sequence number given. Then the response to retransmission request will be constructed and sent towards the destination node. Upon receiving the packet the receiver node checks for its quality of the packet with its CRC value calculation and store this new payload as fresh payload (if it was not received before). And ACK response will be sent to sender but the sender will now only response to the NACK packets in second phrase. The receiver keep searching for the missing packets until there is no other missing packets found. This means that all the packets meant to send the entire image file is now correctly and wholly transferred to destination B. Node B sorts the sequence number of the packets in order as it is very important the payload received are joined together in right order. The payloads corresponding to the sequence numbers are also get sorted and joint together to create the image file on the receiver. After creation and printing the image the destination node change its radio settings to its default data rate (19.2 Kbps) and it send a confirmation to the sender node about the successful completion of file transfer in this phrase. The sender receives this confirmation and changes its radio setting as well. Both node then re-enter the initial state of the transmission scheme. By using this transmission scheme III, we completely change the way the system reacts (on other two schemes) and becomes free of its transmission limitations. The sender and destination send and receives the packets freely from each other and yet at the end of the transaction they re-collect their dropped packets which makes the system much faster on transaction period. In our test situations the systems working on scheme III implementation operate much faster yet efficiently than the nodes in other scheme implementations.

Additionally, both nodes have a limit of retransmission waiting time (on sender) and retransmission request time (on receiver). The retransmission waiting time for sender is set to 300 sec with 1 sec interval which is equivalent to 5 mins. On the receiver side, the retransmission request limitation is set to 10 counts instead of a clock timer. These values are reconfigurable and can be set to any other values by the developers. The reason to add these values in the application so that the sender and receiver do not have to wait forever in a loop of waiting to get response from the other node in the network. For example, sender node can keep listening for 300 sec on the radio for the NACK request from the destination node and after the timer is up sender will consider the receiver node has been exhausted or the connection is disconnected. Then it can automatically set back its radio setting to default data rate (19.2 Kbps) and restate to system's preliminary state. Similarly, the receiver node can continue to send the transmission request for up to 10 times and when the timer is up it will consider the sender node is disconnected from the network. Then receiver will also set the radio setting back to default and reset its state to preliminary system start position.

4.5.5 Scheme III flow diagram

Fig. 4.9 gives us perception of the transmission procedure occurring in first and second phrase of the scheme III. Here the first phrase are indicated as solid lines and second phrase as dotted lines.

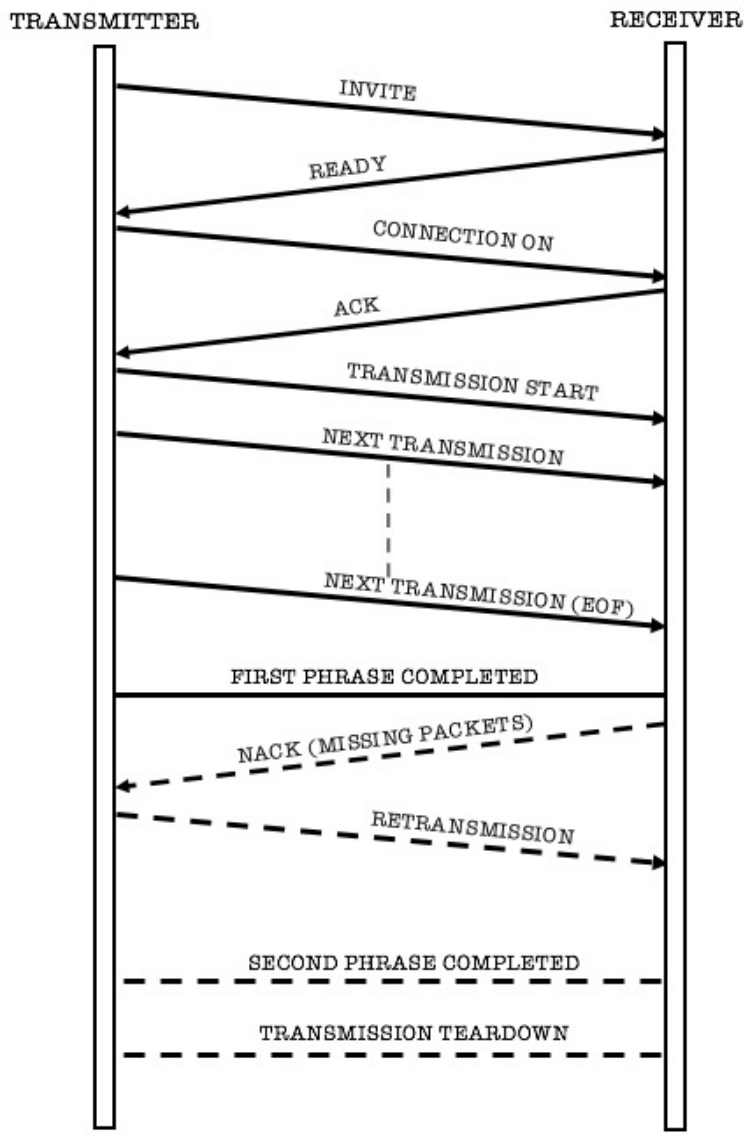


Figure 4.9: Flow Diagram: Transmission Scheme III

4.6 Summary

We implement our transmission schemes in the various test scenarios and collected the experimental data achieved with these tests. We compare our transmission schemes performance with different parameters and investigate the behaviour of the 868 MHz enabled radio module. We give the detailed description of our test scenarios and results in the next chapter.

Chapter 5

Test Scenarios and Experimental Results

In this chapter, we discuss the test scenarios and evaluation process of our system's performances. The chapter is divided into several sections to categorise the results based on their test scenarios and nodes transmission criteria.

5.1 Test Scenarios and Parameters

We implement our transmission schemes in different test scenarios. The tests are based on real-life scenarios and classified in three different categories. The experiments will be tested on indoor, semi-open and outdoor scenarios. Experimental data are gathered by collecting corresponding log files in our sending and receiving nodes. The log files gives us the information about the data packets generated on the sender node in order to sent them via ERA-Connect2Pi radio module and same packet received in destination node via the similar radio module attached with the receiver system. Then we the compare these log files between each other for every possible scenarios and various applied conditions. We investigate the performance of our system design by evaluating different parameter scales of this embedded design by implementing our transmission scheme I, II and III on them. The Table. 5.1 gives us a brief overview to our experimental test scenarios.

Table 5.1: Test Scenarios and Attributes

Scenarios	**Name**	**Approximate** Distance	**Number of** Hops	**Schemes Applied**
1	Indoor	A to B = 8 m B to C = 8 m	1, 2	I, II and III
2	Semi-Open	A to B = 50 m B to C = 80 m	1, 2	I and II
3	Outdoor (Residential Area)	A to B = 300 m B to C = 300 m	1, 2	I and II
4	Outdoor (Open Field)	A to B = 750 m and 900 m	1	I

We discuss the test scenarios in these following sections.

5.1.1 Test scenarios

In this section, we describe the scenarios for testing the system to the readers for a better understanding of our test scenarios. We categorise our test scenarios in to three parts which have different settings and placements of the nodes to gather experimental data. The different test scenarios are based on real-life environment in which most of the wireless sensor devices are designed to implemented on.

Indoor

Our first test scenario is performed in an indoor environment. The placement of the nodes are relatively close to each other considering the other test scenarios. Our goal is to test the capability of our embedded system design in an environment where the devices are in a space which is inside a closed door situation and has obstacles such as (dry wall, other electronic equipments, households, etc.). We select a space which is about 50 m^2 and has a dry wall partition in between them. Fig. 5.1 represents our implemented scenario for the indoor test. The red circles are representing the placement of the node A, B and C respectively. The dry walls between these three nodes have standard 12 cm thickness. Distance between A and B is about 8 m and with C its almost 7.5 m. We aim to evaluate the performance of the entire system by implementing our transmission schemes I, II and III on the transmitter / sender, forwarder / relay and destination / relay nodes A, B and C respectively. Our result shows the performance of this test in Secs. 5.2 - 5.4

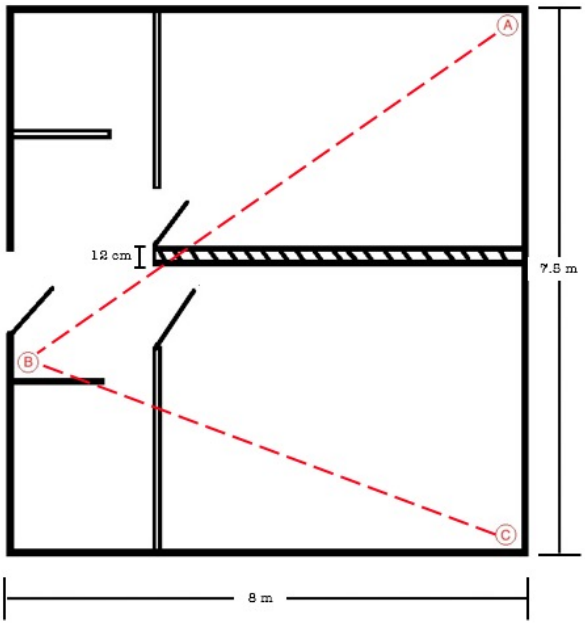


Figure 5.1: Illustration of Indoor Test Scenario

Semi-Open

In this test scenario, we implement our system design in a semi-open environment. We chose the placement of the nodes in various points of this scenario which is represented with the Fig. 5.2. The location of this test is the cafeteria area of University of Agder, Campus Grimstad. We use the floor plan of the area to let user understand the placement of the nodes in this location.

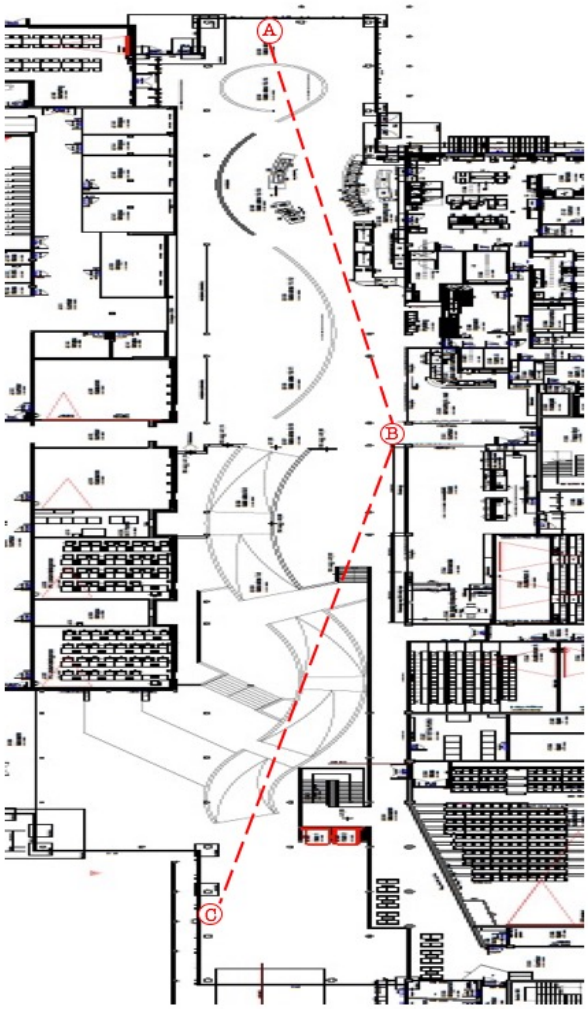


Figure 5.2: Illustration of Semi-Open Test Scenario

The distance between node A and node B is the about 50 m is distance. This distance is more than double than of our indoor scenario. The placing of nodes B and C is also almost 80 m apart. This semi-open scenario is chosen as test to evaluate the performance of our system in a crowded wifi (or other type of network) zone. Our Connect2Pi radio module operates on 868 MHz ISM band and this particular radio is designed by manufacturers to avoid all the interference generated by the devices operating on 2.4 and 5 GHz frequency band. We tend to implement our embedded system in this crowded scenario of campus area to investigate the performance of our designed transmission schemes. We perform our test on a busy day in campus to evaluate how well our system work with the many obstacles (people, other electronics devices, etc.). The red circles in Fig. 5.2 are representing the nodes A, B and C in this semi-open test scenario. The floor plan is the courtesy of University of Agder, Norway.

Outdoor

For our outdoor test scenario we continue our experiment outside in open area. We chose a clear day to run our tests and gather the log files for different conditional scenarios. Two different locations have been chosen for our outdoor scenarios. One of the location is around the Kristiansand Stadium (Snorres Gate, Kristiansand, Norway) and other is in front of the JBU building

in Grimstad (Terje Løvås Vei, Grimstad, Norway). We consider first location as the residential area and the second location as the open field area. Fig. 5.3 shows the experimental placement of the node A, B and C in the residential area. Figs. 5.4(a) and 5.4(b) shows the placements of only the nodes A and B in open field area near Grimstad. All the maps are the courtesy of Google Maps.

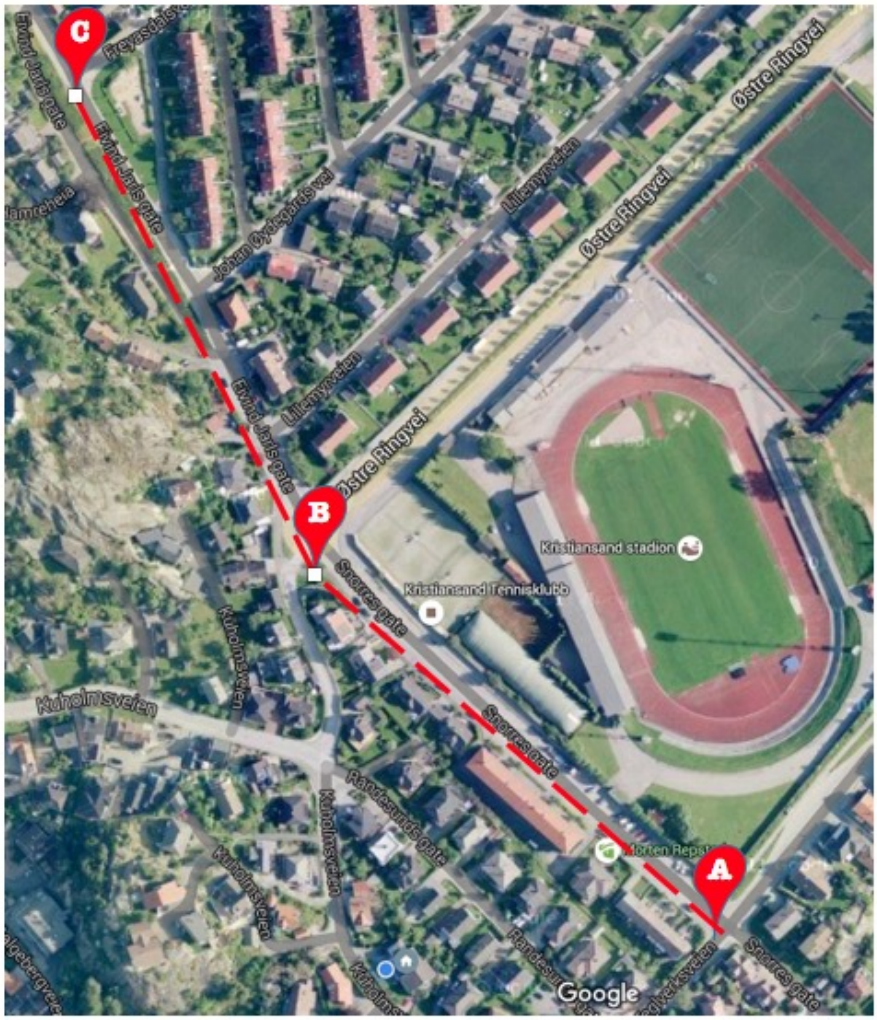
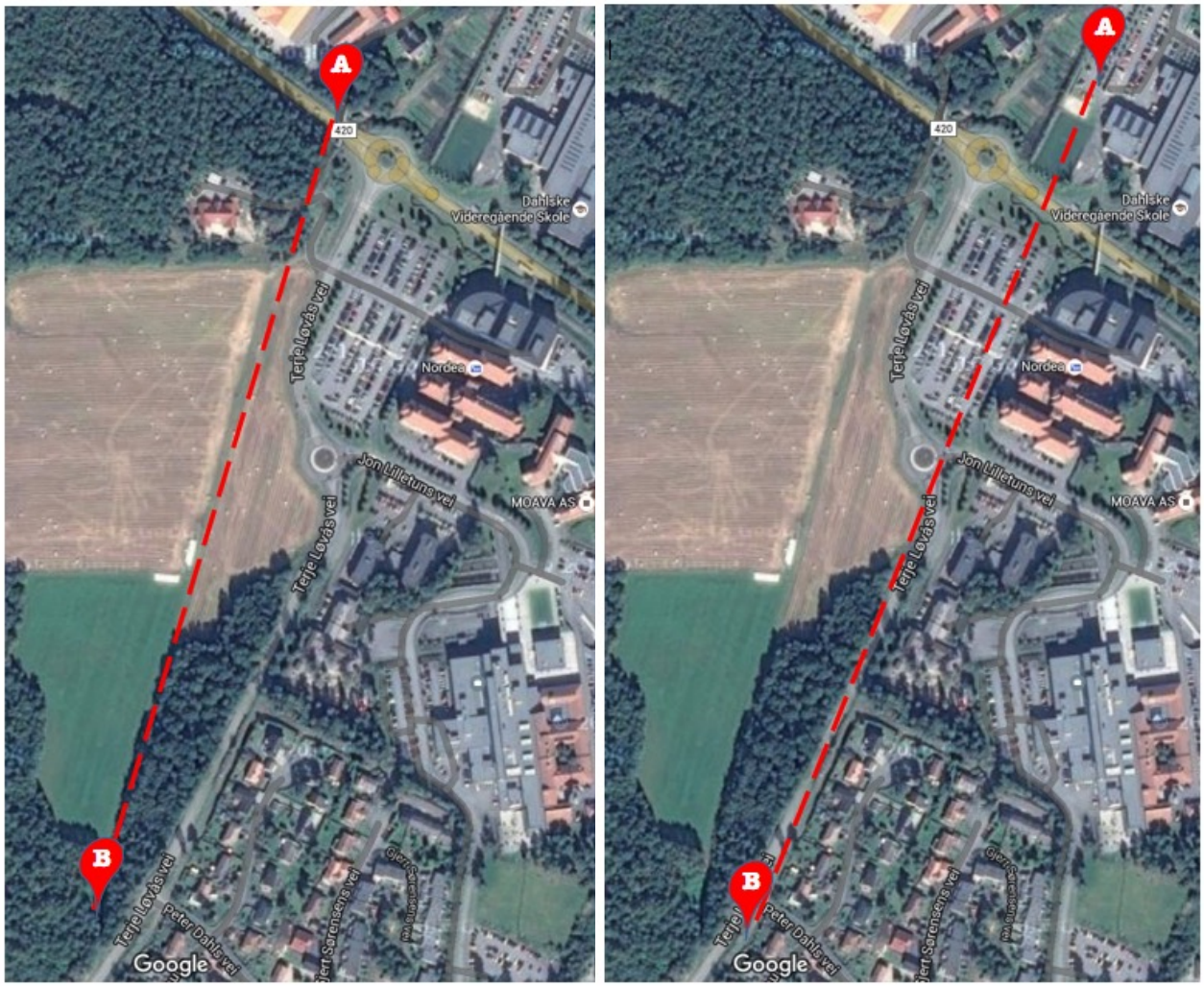


Figure 5.3: Illustration of Outdoor (Residential Area) Test Scenario

In this test scenario, the distance between node A and B is almost 300 m. After collecting the experimental results by implementing transmission scheme I between these two nodes we further placed the node C also 300 m away from node B. Transmission scheme II in implemented for this network .

The first test on Fig. 5.4(a) indicates the test where the two nodes are 750 m away from each other. We run our transmission scheme I in this scenario and gather the data generate at both end. Then we change the location for few meters and place the sender and destination almost 900 m apart from each other. We collect the data (log files) in this scenario as well to compare performance between the two open field test scenarios. Fig. 5.4(b) illustrates the placing of the sender and receiver nodes on the map.



(a) (b)

Figure 5.4: Illustration of Outdoor (Open Field) Test Scenario

5.1.2 Test parameters

We conduct various tests on different test scenarios to gather our test results and compare the performance of our designed system. These results give us a good vision of how the system perform in the real-life based test scenarios. These tests are combinations of different schemes on multiple nodes in different test scenarios. All our results are collected in form of log files in transmitter, forwarder and receiver system and from those corresponding log files we measure our performance evaluations is this section. The corresponding log files usually printed according to the frame structures that has been sent and received in sender and receiver nodes respectively. A researcher can easily recognise the pattern the log files have printed and measure data from these readable versions of raw data. We attach a sample of the corresponding log files in Appendices A and B. Although these log files are easy to read for any reader but it is difficult to understand the real performance evaluation from this text based files. We need to have some basic form of graphical representation for our readers to understand the experimental results and the comparison between them. We compare the data printed on the logs and investigate the performances of the system, then we represent these comparison between the scenarios and put them in graphs using **Microsoft Office Excel 2016**. In the graph, the X axis is represented as the time for sending and receiving packets and Y axis represented as the number of packets. We calculate

the first 100 (50 in some test cases) packets of the both systems in general as the rest of the transaction results has the same continuity. Before we continue with the experiments we must note that none of our nodes are actually synchronised in terms of hardware clock time. The time is different in every nodes because the micro-controller (RPi) the nodes are basically based on (and our designed application is running on) does not have access to the network connection. Since the micro-controllers are AC/DC powered devices and does not have an onboard battery on motherboard it is obvious that the nodes will reset its timer every time it goes through a power cycle. Hence our log files showing the time in different nodes are not synchronised to each other. We consider comparing our experimental data in respect to several parameters. These parameters on which our comparison are based on are as follows:

- **PLR**: We calculate the PLR of the transaction between the two nodes for an ongoing process. The PLR is calculated by using the simple equation: $\frac{\textbf{Packet Received}}{\textbf{Packet Sent}}$. We generally calculate the PLR for the first 100 (50 for some cases) packets on the entire transaction we are evaluating. Future researchers can calculate for the entire transaction for an unlimited period of time if necessary.
- **Transmission Delay**: Transmission delay will also be calculated to find the amount of time is required to push all the packet's bits into the line (in our case it is the radio module). This delay is calculated by the formula: $D_T = N/R\ sec$, where D_T refers to the transmission delay in seconds, N is the number of bits and R is the rate of transmission (bps). It is the delay caused by data rate of the link we assigned in our system design.
- **Network Delay**: We are giving an estimated delay of our running tests which represents the amount of time for a packet of data to travel from one link end to another. We are referring this delay of the entire network. We include this delay as the term of network delay because it is noticeable from the curves in the graphs that the packets sending and receiving time differences increase over the number of packets are processed. We want to point out that this delay is the processing and queuing delay of the receiver node with incrementation of packet's number. However, we cannot give accurate network delay value since we cannot differ this delay from the other delays such as propagation delay, queuing delay, etc. We are referring it as the network delay in this thesis work for reader's understanding of the performance of the nodes.

Rest of the performance evaluations are categorised is different parts which we discuss in the following sections.

5.2 Experimental Results for Scheme I

In this experiment, we implement the first scheme on every possible test scenario and compare the performance results in between them. Here the tests are obtained in different environments and also with the combinations of various data rate (reconfigurable) settings of the radio modules. The following sections represents the combinations of experiments with scheme I implementation.

5.2.1 Scheme I: Transmission vs receiving time

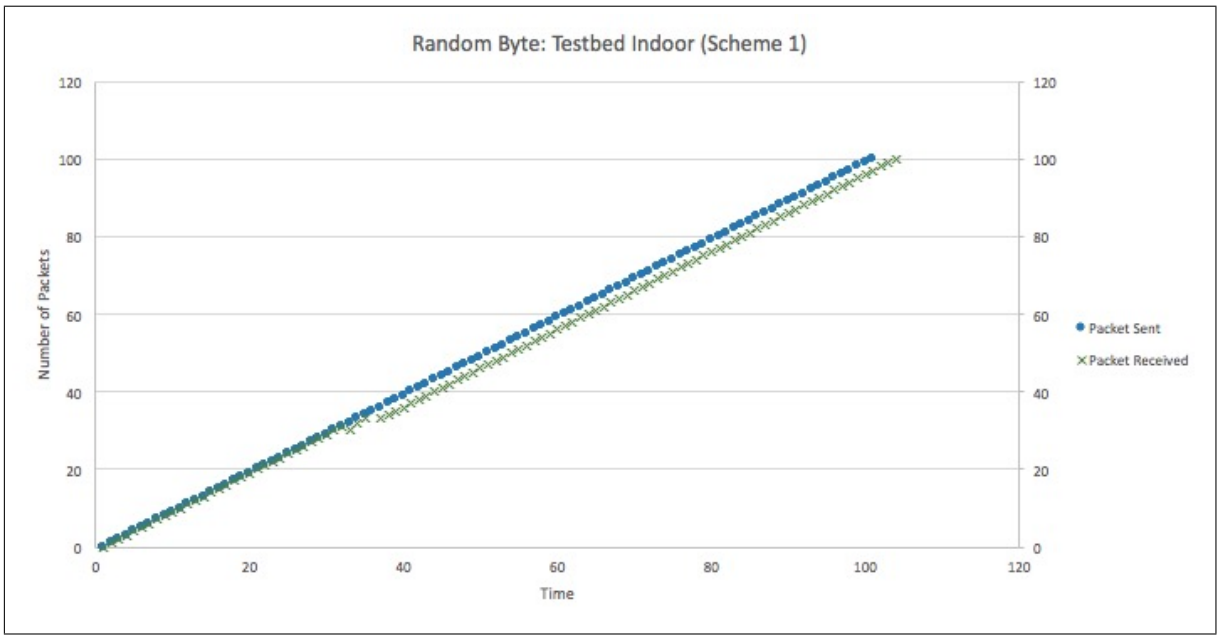
Our first experimental data are gathered on two nodes in different test scenarios. We implement our design scheme I on our sender node and the destination node as we discussed before in Chapters 3 and 4. We conduct our experimental tests by sending the our services i.e., random bytes,

image, GPS information and temperature from sender node (A) to destination node (B). The results shows the performance of system by calculating the exact time the sender node sends a packet and the exact time when the destination received the same packet. We determine how long it takes for a single data frame travel wirelessly from the sender and received in destination. Each packets are sent individually by the sender when it receives an ACK from the destination node. It also retransmits those packet that has been corrupted or disrupted during transaction on the way to destination. This retransmission occurs when the sender receives a NACK from our destination. Keeping this mind, we first test our simplest service type (random bytes) to generate the data frames and send towards the node B. We compare the time when the sender receives an acknowledgement (either ACK or NACK) and the time when the destination received the corresponding packet. Since the packet sent time is very continuous and it generates the time interval in very linear fashion. Thus we decided to use the acknowledgments incoming from the receiver to be the corresponding sending time for transmitter. We like to note that, the actual packet sending time for sender is exactly the same time it received the acknowledgement as the sender node will not continue to transmit unless there is an ACK for fresh packet transmission or a NACK for retransmission. We discussed this mechanism already in Chapter 4. The graphs in Figs. 5.5(a)-(b) and 5.6(a)-(b) represents the performance evaluation of the sender and destination nodes by implementing transmission scheme I on test scenarios indoor, semi-open, outdoor (residential area and open field) respectively.

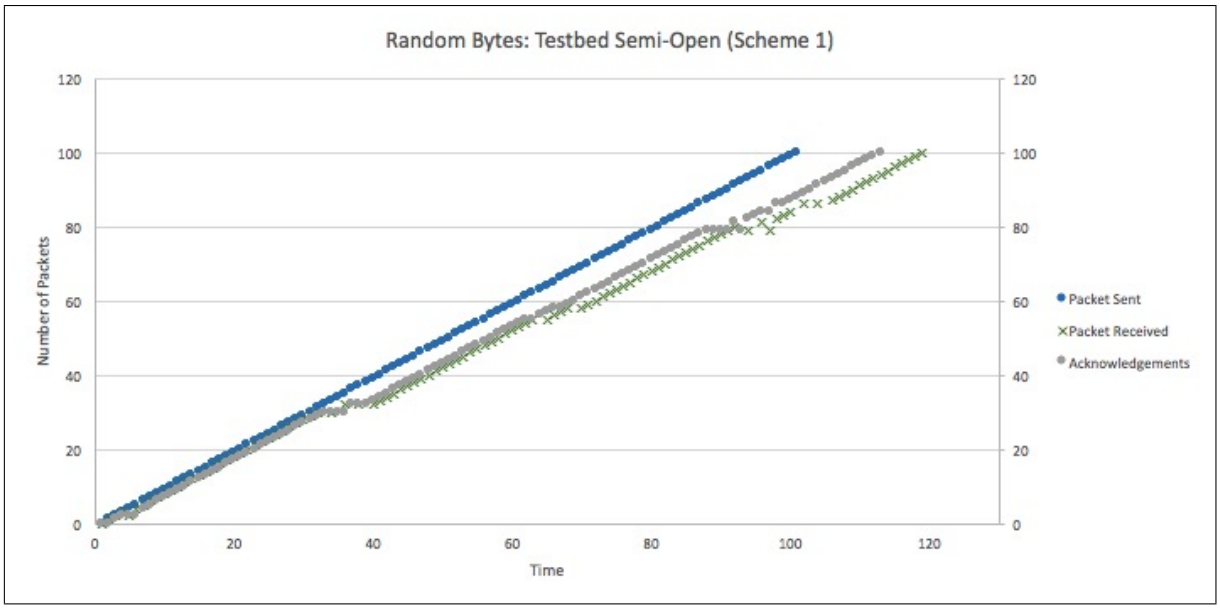
The graph (Fig. 5.5(a)), shows the performance of the two nodes in the indoor scenario. For the first figure we want to give the users some idea of how it will look like if we compare a packet's received time on sender according to the sent time for that packet. As it is noticeable that the sent time curve (blue dots) on sender is quite linear than of receiving time curve (green crosses) and it does not give the actual perception of the performance evaluation since the both system clocks are not synchronised. According to Fig. 5.5(a), it is noticeable that the receiving time for all the packets are quite stable throughout the transaction. The PLR of this entire transaction process is 0% although some retransmission of the packets are noticeable at 35th ms of the transmission for the first 100 packets. This retransmission can occur due to the other radio interference around the test location. With many repetition of this particular test, the nodes give us same results every time.

In Fig. 5.5(b), we include the acknowledgements received on sender on its corresponding time to show the real-time differences between the packet sent time and packet received time on sender and destination nodes respectively. As we discussed before, the actual sent time interval on sender is quite linear to its own timeline and it cannot really give us the opportunity to observe the entire transaction progress with the graphs. We add this acknowledgement curve (grey dots) sent from receiver (ACK / NACK) and received by the sender in order to sent the next packets towards destination. In this figure, we observe the inter-relationship between the two running nodes. We can notice that the acknowledgment and packet received curve is almost in the similar time interval for first 50 packets. Then we noticed some packet drops (about 1% for first 100 packets) and many retransmission procedure took place in later. The network delay is noticeable around 2-6 ms. Some interferences are noticed during this testing which we assume due to possibilities of other radio modules on similar frequency band around the test locations.

Similarly for the outdoor test scenarios we gather the experimental results and try to calculate the PLR and transmission delay between sender and destination. Like we discussed earlier at the beginning of this chapter, we consider two different test scenarios for outdoor experiments. One based on residential area and the other one on an open field. Fig. 5.6(a) showing the graph which representing the packet sent and received time for first 100 packets in this outdoor test scenario (residential area). We can observe the delay between sender node sending and destination node



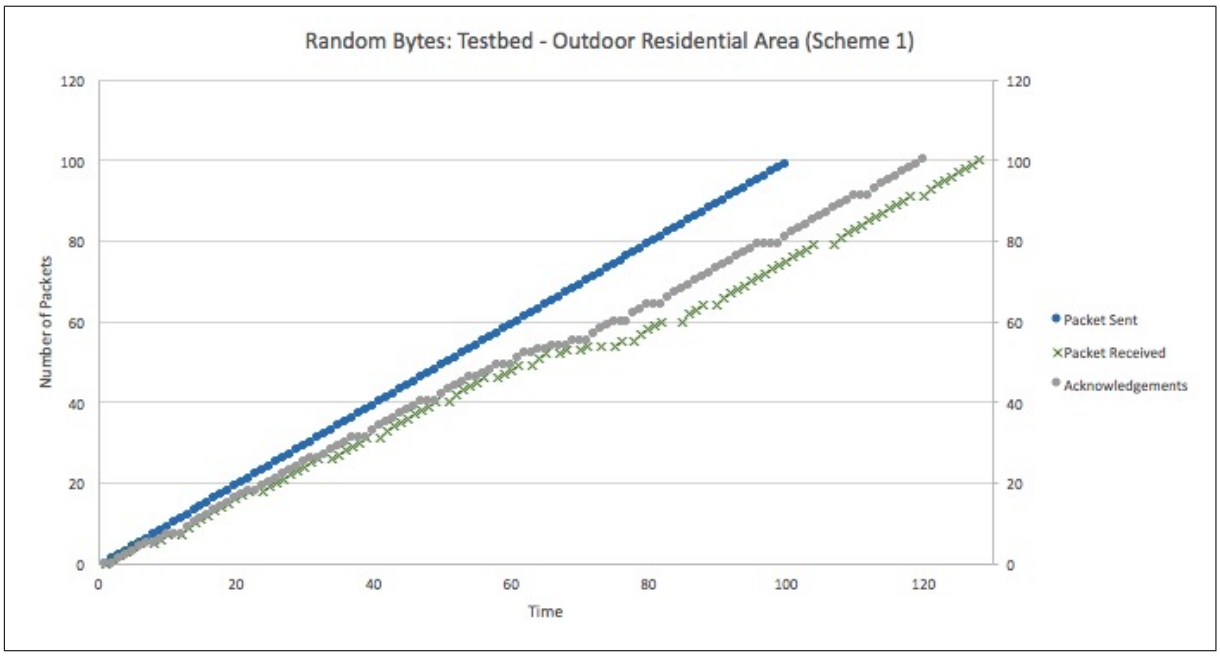
(a)



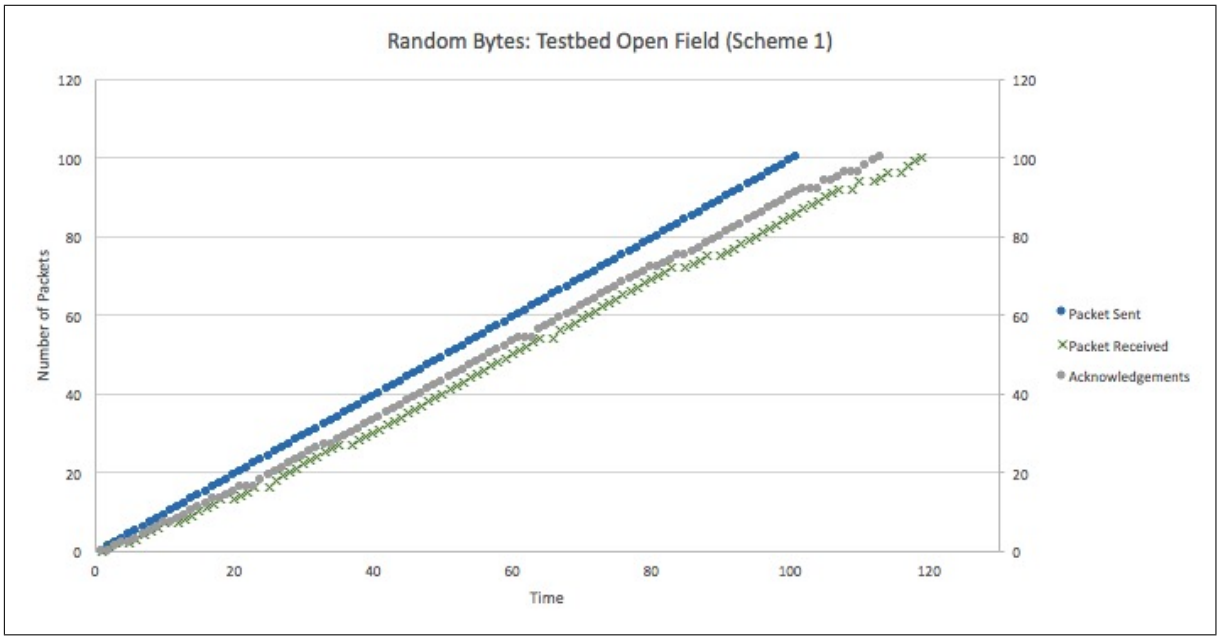
(b)

Figure 5.5: Scheme I (Random Bytes): Implementation on Indoor and Semi-Open Test Scenarios

receiving is much higher than the indoor and semi-open test scenarios. It is also observable, in residential area the number of packet retransmission is more than the previous two cases. The PLR is almost 9% for total of 100 packets and it is much higher than the previous tests. The network delay is quite decent for 300 m of range which is about 5-8 ms in this test scenario. Interestingly, in our outdoor (open field) test scenario the network delay (considering all the other delay in the network combining) shows much lower than the residential area test even though the open field test has more than 750 m of distance in-between the sender and destination nodes. We calculate the PLR from the results collected in this test which is almost 4% on total first 100 packets transaction. The PLR is better than semi-open tests. Fig. 5.6(b) represents the graph for the outdoor open field test. The curves representing packet sent, acknowledgements and packet received time is quite linear in this test which represents the stable connectivity throughout the



(a)



(b)

Figure 5.6: Scheme I (Random Bytes): Implementation on Outdoor Test Scenario

total transaction period. The time it takes a packet sent and received is noticeable as on average 2-4 ms for these 100 packets.

We observe from the test results that our system design works best in the indoor scenario although the performance of system in the open field is considerably much better than the semi-open and close range residential area test considering the distance was achieved in the open field test scenario (about 750 m to 900 m). We assume with a lower data rate or the smaller packet size the outdoor test performance of the nodes will show much better performance considering the transmission and network delay will get much lower than this. The random bytes service consists of 180 bytes

of each packet size and for all the above test we used the same default data rate of our radio module which is 19.2 Kbps. The transmission delay for the entire transmission scheme 1 test is $\frac{180X8bits}{19200bps} = 75$ ms. This value is same throughout the entire tests we performed between two nodes with single hop transmission.

5.2.2 Scheme I: Comparison between types of services in outdoor (open field) test scenario

In this test, we put all of our integrated different types of services to run on the outdoor (open field) test scenario. As explained earlier, this open field test scenario has a good range of from 750 m to 900 m. Our radio module has the ability communicate with the other similar module for up to 2.2 km. We could not go any further 900 m in this test due to the localisation limitations. We perform the test where we run all four types of services (random bytes, image, GPS co-ordinates and temperature) by implementing scheme I with 19.2 Kbps data rate. This means we only used two nodes in this scenario. After running this tests we collect the results from the communicating nodes and create the graph (Fig. 5.7) from which we can observe visually the data gathered (first 100 packets) on the destination node and performance comparison between them.

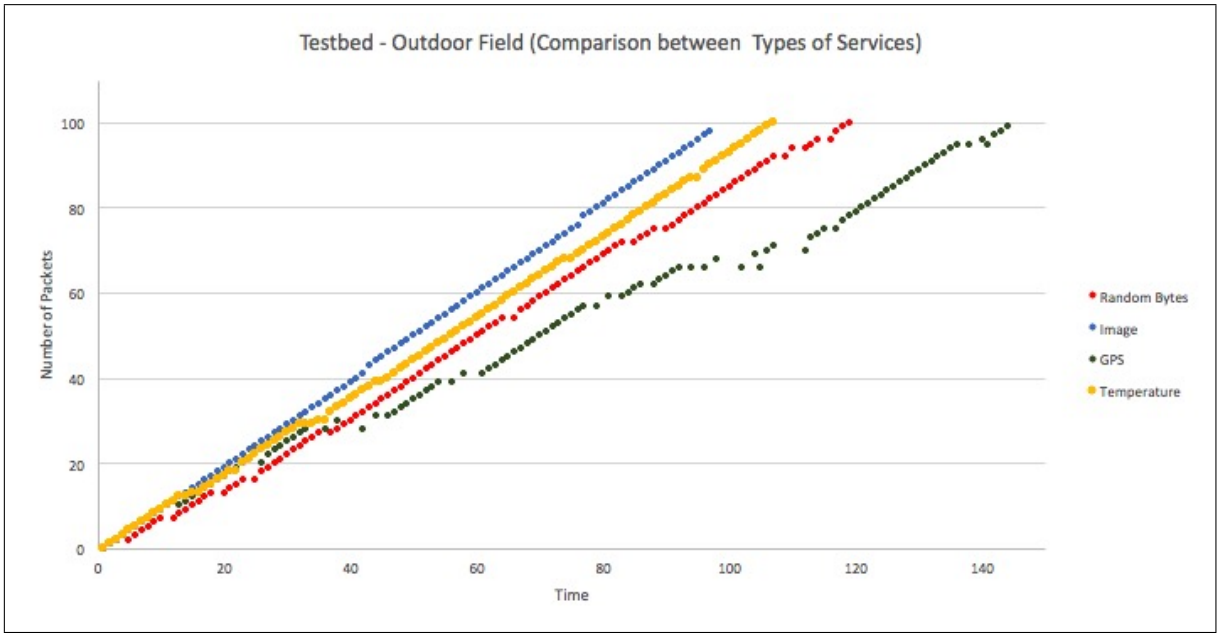


Figure 5.7: Comparison of Service Types: Outdoor (Open Field) Scenario

From the corresponding graph, we can find out that random bytes as the type of service overcome as the fastest service that is received in destination nodes. Our test scenario made a great impact of the outcome of the experimental results. The reason behind the image transmission process is done quicker than all other services because of stable transmission and reception time interval. The nodes need to be quickly receive these packet for image bytes as we designed it while keep it in my mind that it is the most important factor for any file processing protocol design. The second in position in the performance curve is the temperature service. Random bytes service also performed well until it starting to show the tendency of generating network delay quickly. The worst case scenario (not absolutely worse) is the GPS co-ordination service. We consider this service took the most time to receive as not only this service have to maintain a connectivity with the destination by transmitting and responding to the acknowledgements but also the sender node has to maintain a connection with the GPS module which is attached to its USB port. The

processing time for gathering data and creating the packets are included in this GPS curve (green dots) representation. Moreover, there is number of packet drops noticeable due to the distance the data need to travel to reach the radios. The PLR of these test results are for random bytes it is 3% for and temperature service it is also 3%. Surprisingly the Image service and the GPS both have a PLR of 4%.

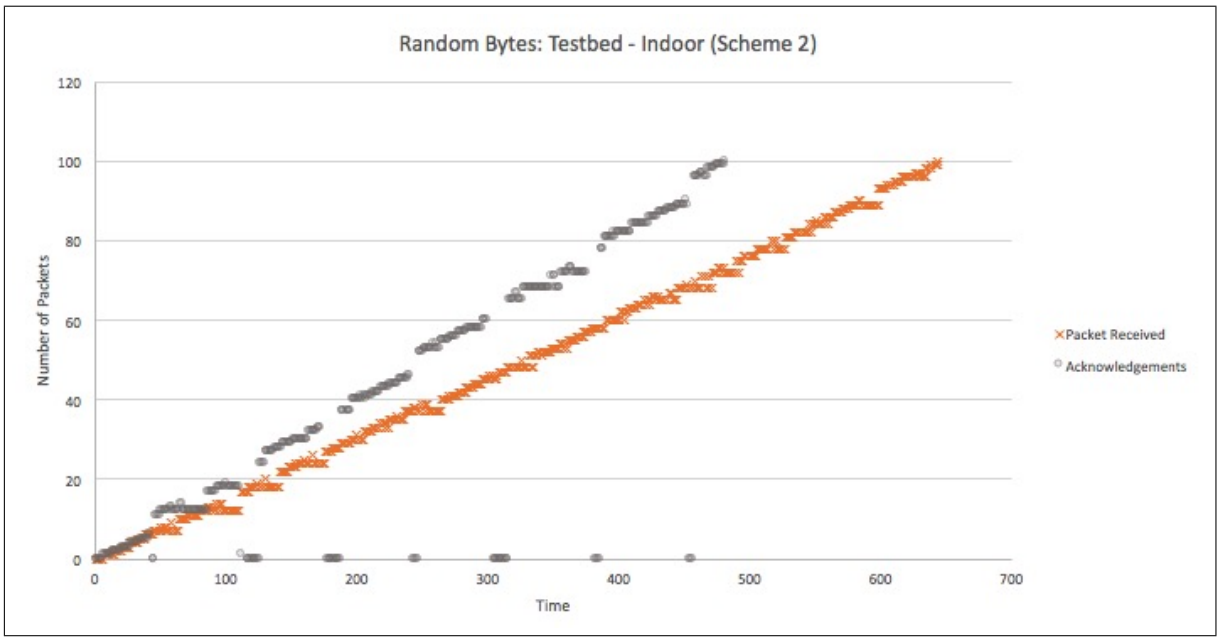
5.3 Experimental Results for Scheme II

Similar as the scheme I we implement scheme II in different test environments and compare the experimental results in form of the performance evaluation of the nodes. The explanation of these results are carried on these following sections.

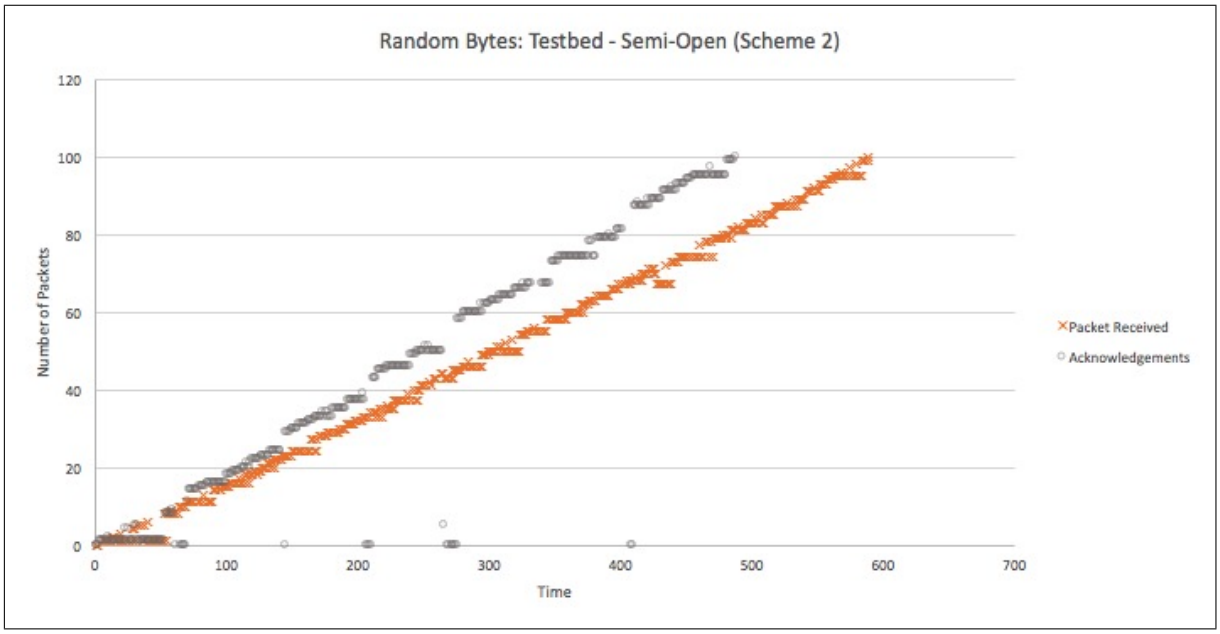
5.3.1 Scheme II: Transmission vs receiving time

We continue with our first test and implement our second transmission scheme in our test scenarios. This test procedure consists of three nodes which we refer as sender, relay and receiver nodes. Our scheme II design, the transmitter system will act as the sender node in this scheme and the forwarding system and the receiving system will act as the relay and destination node respectively. We implemented the same type of service as previous test (random bytes) and evaluate the results abstracted from the printed log files on sender and destination nodes. We chose to implement the scheme on test scenarios indoor, semi-open and outdoor (residential area) in this test procedure. The graphs in Fig. 5.8 and 5.9 illustrates the representation of the experimental results summarised from the test scenarios. The transmission delay is calculated as the same the scheme I implementation which is 75 ms in this case as well.

We conduct similar tests and assess the results in same way as the previous experiments. According to the Fig. 5.8(a), which shows the graph for results gathered by sending random bytes from sender (A) and forward the packets via relay node (B) towards the receiver node (C). We collect data from both nodes A and C to compare the PLR and delay between them for the transaction period. First we use the test scenario indoor to obtain our data. Then we create the graph with first 100 packets of the total transaction which ran quite flawlessly. After generating the graphs, we notice a huge amount of retransmission occurred in these transactions. Many packets have been dropped during this test on receiver node and number of retransmission in much higher than in scheme I. We can notice that the retransmission has been done for many packets in different moments of the transmission time period. We consider the reason behind is that all three nodes can hear each other during this periods and the response time for each node were set to be higher on both sender and receiver end. The relay node had to work much faster than other two nodes to maintain the relaying mechanism properly which also affect on timing of the data frames sending and receiving. Maintaining the connectivity and response according to the request between these two end systems is more difficult for the middle system (relay node). Since the response time gets higher (due to queuing and processing time) the packet forwarded may become corrupted or missing in the transaction as the sender and receiver node have their radio listening time intervals. Also on the sender and receiver side the amount of packets received (either ACK / NACK on sender and data frames with payloads on receiver) can become little hazardous due to the multiple packet receiving and calculating which packets to keep and which packets to discard. Due to the burst of many packets the nodes get exhausted very quickly or the connection disconnects faster in these scheme II of the protocol design. It is understandable from the graphs that the node A and C both missing many packets that meant for these systems Since the packets are missing not all the data are sent to receiver and not all the data packets with payload have been received successfully to the receiver. In the graph, the orange dots represents



(a)



(b)

Figure 5.8: Scheme II (Random Bytes): Implementation on Indoor and Semi-Open Test Scenarios

the time packet received in node C and grey dots represents the acknowledgements receive time by the sender A. We like to point out that the delay between the two time are quite more than the scheme I because this delay is included the network delay between the sender A and relay B and between B and receiver C. The PLR calculated in this test is almost 11% of the total 100 packets transaction. Our calculated network delay is about 20-162 ms for this 100 packets. Like we mentioned, this is including the processing and queuing delay of the relay node too. The difference is quite noticeable as number of packets increases on receiver side.

The Fig. 5.8(b) the graph also represents the performance evaluated on the test semi-open location by implementing scheme II transmission design. We notice a huge amount of packet retransmis-

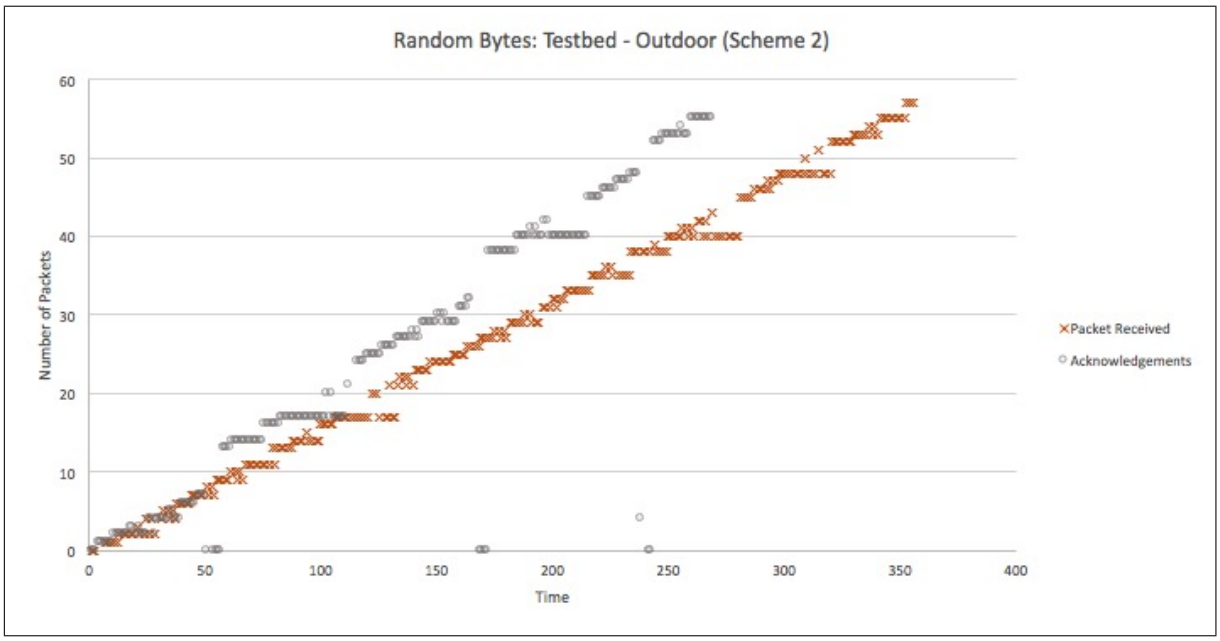


Figure 5.9: Scheme II (Random Bytes): Implementation on Outdoor Test Scenario

sion at the beginning of the transaction. This can occur due to the interferences of the equipments in the campus location which might generates interfering signals. There can be some 868 MHz enabled radios operating in the campus are which can create these interferences. The transmission became quite stable after some moments and the rest of 100 packets transaction period was quite similar to the indoor test result. It is also noticeable that the number of retransmission is less than the indoor scenario. A reader can easily notice the density of the packets received on both sender and receiver side is less than the previous case. It is possible due the increment of distance in this test as we placed the nodes further than indoor scenario in semi-open scenario which is about 50 m between sender and relay nodes and about 70 m between relay and the receiver nodes. This gives the opportunity for all the attached radios on the nodes to listen to their ports calmly and response more efficiently without being jammed with all the other unnecessary incoming packets. The network delay between the receiving times of the nodes are also little better in this case (11-101 ms). The PLR for this test is giving us the value of 13% for this test.

We continue with the test of implementing scheme II on outdoor (residential area) test scenario and gather important data. The Fig. 5.9 gives us the graph that shows the experimental result comparison of the systems. The distance between the nodes are more in this case (about 300 m between A,B and C). The range affect the transmission quality for this scenario which we can observe in the test results. We ran into issue of maintaining the connectivity between the nodes. The data traveling through the medium having issues to reach over the network to the each nodes. We observe the node getting exhausted quicker because of the connectivity failure between the nodes which we can say the closed environment of the scenario was one of the big issue. The obstacles around the test area must have some influence in this matter and the processing time get shorter than the queuing time of the nodes. In the graph, it is observed that the number of packet drops are much higher than previous two tests for transmission scheme II implementations. We evaluate the results of first 50 packets in this matter and we represent the graph accordingly. In Fig. 5.9, the network delay between sender and receiver nodes were similar as we observed the number of packets received in receiver was not many in numbers. When the network became stable the number of packets started to increase on node C. The delay is about 21-82 ms which is lot higher than the other two tests since we are only calculating this delay for 50 packets only.

The value PLR is almost 12% which is also high rate for such small amount of packet transfer.

Our test results shows that overall performance on two-hop transmission is much better when the radios (the nodes) are placed little further than 50 m from each other and this distance plays a big role for the packet drop ratio on the receiving node. Our semi-open test scenarios has better performance evaluation than the other two test scenarios.

5.4 Experimental Results: Various Test Scenarios

In this test, we put our nodes to test in various combination of schemes and data rate to get the best results on performance of the entire system. These following tests are not only achieved on different data rate but also with combination of different schemes with varied test scenarios. We explain the performance to the readers of these experimental test in the following sections.

5.4.1 Image comparison: Scheme I, II and III

This first test includes the comparison of performance of all the transmission schemes on every test. In this case we chose the image sending as a type of service and the image we are sending is the Apple (JPEG, 7.8 Kb). For the scheme I, we run the tests between two nodes. Here one node is the sender and the other one is the destination. Our comparison will focus on how fast our system works considering the test environments (indoor, semi-open and outdoor). We convert our log files in graphs format for a visual representation of the log files, which are generated on both end of the transmission. By comparing the PLR and the network delay we calculate the performance of our system integration along with our scheme designs. To send the image, we have to select the types of service as image sending and the image of the apple will be automatically selected. Then the image will be processed and divided into smaller bytes to insert them in the image data frames. 159 bytes per frame. According to our calculation the whole 7840 bytes of image will take approximately 99 frames to send over our radio module. This image will be collected on receiver end and merge them together by using the corresponding sequence numbers. These payloads need to be in right order and in exact amount as it is crucial in any file transfer protocol if a single bit of data missing while transacting, the entire file can be corrupted. So packet drops while transaction is a zero-tolerance issue for image processing service. We have to make a working solution that can send and receive the image bytes correctly and successfully no matter what is the scenario or radio status is. However, the implementation of an under-development protocol in the real-life scenario has its own impact on the result as we observe in our experimental results.

The Fig. 5.10 shows the graph representing the performance comparison of two nodes in the three tests scenarios. The indoor curve (red dots) gives the results of the receiving time of image bytes on receiver node / system (C) in indoor scenario. The other two curves are semi-open curve (blue dots) which showing the semi-open scenarios performance and the outdoor curve (green dots) showing the results on outdoor (residential area) scenario. The X and Y axis are as usual representing packet time received and the number of packets respectively. With this graph, we can clearly see that in all our test results are almost equivalent to each other in scheme I. Among all three the tests the outdoor scenarios shows the best results as it is noticeable that the packets receive time in this case are quicker than other scenarios. The outdoor tests are always giving us good performance throughout the previous tests and it is showing the same result in this test. The pictures recollected and analysed if they were received on the receiver end perfectly. In most cases we successfully received our test image on the destination node.

Figure 5.10: Image Receiving Performance Comparison: Scheme I (All Test Scenarios)

In the next test, we run the similar tests with the transmission scheme II. We are familiar with the number of packet loss on the receiver end while implementing this scheme. As previously we discussed, our scheme II will be implemented on three node which are sending, relaying and receiving nodes respectively. The sender node generates the packets with payload carrying small fragments of the chosen image and the relay node will forward this payload towards the receiving node. Receiving node then collects all the gathered data and combine and sort them in order to create the image file. We run the experiments and put the results in a graph representation for our readers to experience the performance achieved in each scenario. Fig. 5.11 represents the accumulated graph for scheme II.

By observing this graph, we can observe that the curve for indoor scenario receives the image bits lot faster than other two scenarios. It is also observable that this curve has more drops of packets and transmission overhearing time (the long gaps with no packets) that we discussed earlier in this chapter, are affect of other incoming packets that is not meant for the receiver node. The nodes are quite close to each other in this scenario which can affect more on the radio listening and process the right packet while transaction is running. Because of excessive amount of connection overhearing the receiver misses plenty amount of packets which is in this case a crucial factor. We repeated this particular test for many times and the test shows the almost similar results. We continue the testing process with the scheme II implementation on the semi-open and outdoor scenarios. The curves representing these two tests shows that the connection is more stable and PLR is much less in these tests. After repeating the process for some more turns we concluded with this test that the nodes seems to be holding well (performing better) while they are further from each other. The further the radios (and the node of course) from each other the better the performance it shows. Our graph (Fig. 5.11) shows that the nodes perform head to head on semi-open and outdoor (residential area) tests. This is a good step towards our project goal to achieve longer distance with these low data rate transmission schemes.

Our third scheme performance evaluation is little different than other two scheme evaluation in the tests scenarios. Instead this test is carried out the performance of our system with the different

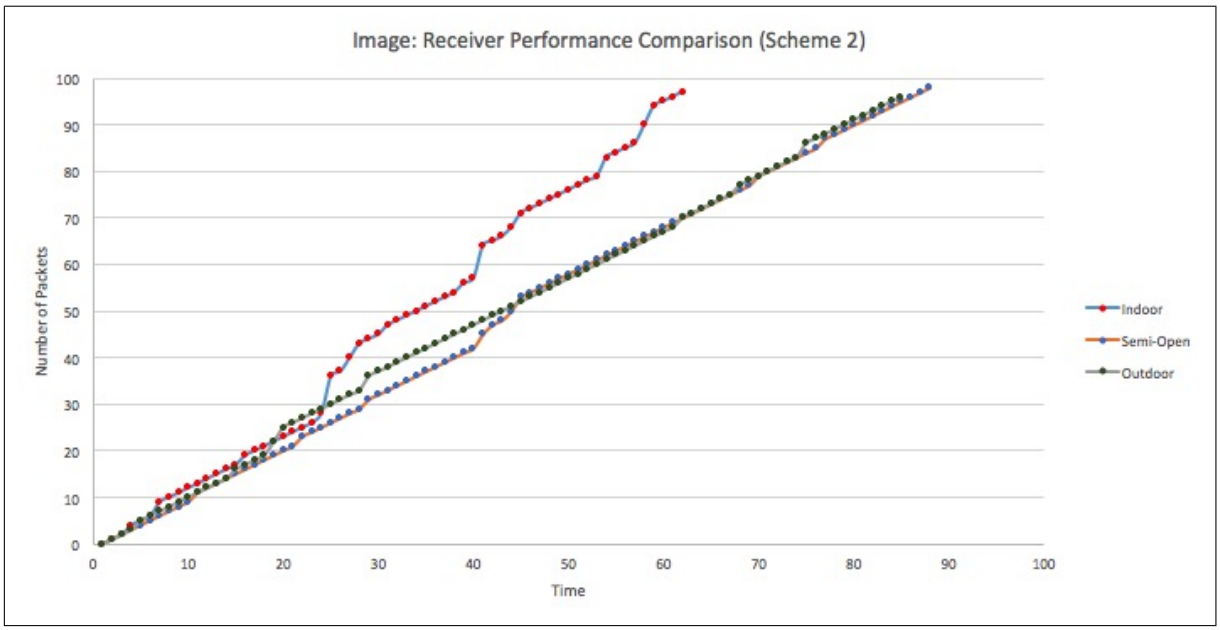


Figure 5.11: Image Receiving Performance Comparison: Scheme II (All Test Scenarios)

data rate in indoor scenario. Our scheme III is almost the same except we try to perfect our design by dividing the transmission into two phrases. First phrase just to transmit fresh copies on image bytes until no more packets to sent to destination and the second phrase to retransmit according to the missing packets requested by the destination. Same as first scheme, this scheme III also has only two nodes (sender and destination).

We observe astonishing result for the scheme 3 adaptation to all the different data rates. It is clearly visible that all the performance evaluation has the exact same receiving time intervals. The curves are almost linear and overlapping each other. This is a good indication that our schemes' working principal is quite nifty in this case and can be implemented with any data rate. We also notice some of the retransmission processes have been done by the sender and received on the destination node at the end of transmission. This is the main principal of our scheme 3 design and we can see that this principal has been implemented quite efficiently. Although the curves are almost overlapping to the naked eyes the system running on 19.2 Kbps performed better among all other with a thin margin. In this test experience, we have noticed that the image was successfully transferred to destination with this protocol (transmission scheme III) every single time. This is a promising step for our development of the system prototypes.

After running the tests we collect many different images received and printed successfully on the destination node (B). Some of these images are corrupted but many are received in perfect condition. Some images also received in a manner where the order of the bytes are scrambled and printed as disfigured images. We attach a few of many images in this report in order to represent the scrambled images received in different test scenarios. The Fig. 5.13(a) shows the image we received while performing the test for implementation of scheme II on semi-open test with 19.2 Kbps data rate as radio settings. Fig. 5.13(b) is collected from the outdoor (residential area) scenario with implementation of scheme I with 9.6 Kbps data rate. Additionally, we receive the image in Fig. 5.13(c) while running the scheme I implementation on indoor scenario with 192.2 Kbps data rate setting. All of these images received sometimes in disorderly fashion which made them appear as scrambled. Note that, even after receiving a distorted image, repeating of the same test (with same configuration, without alteration) can possibly receive a successful image such as

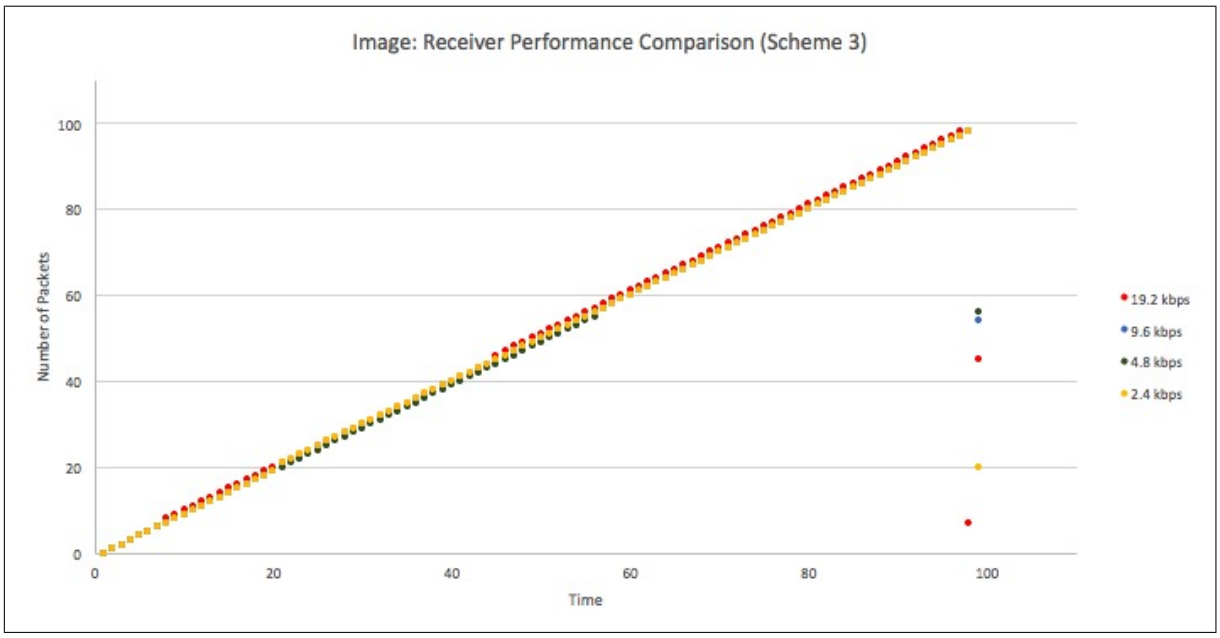
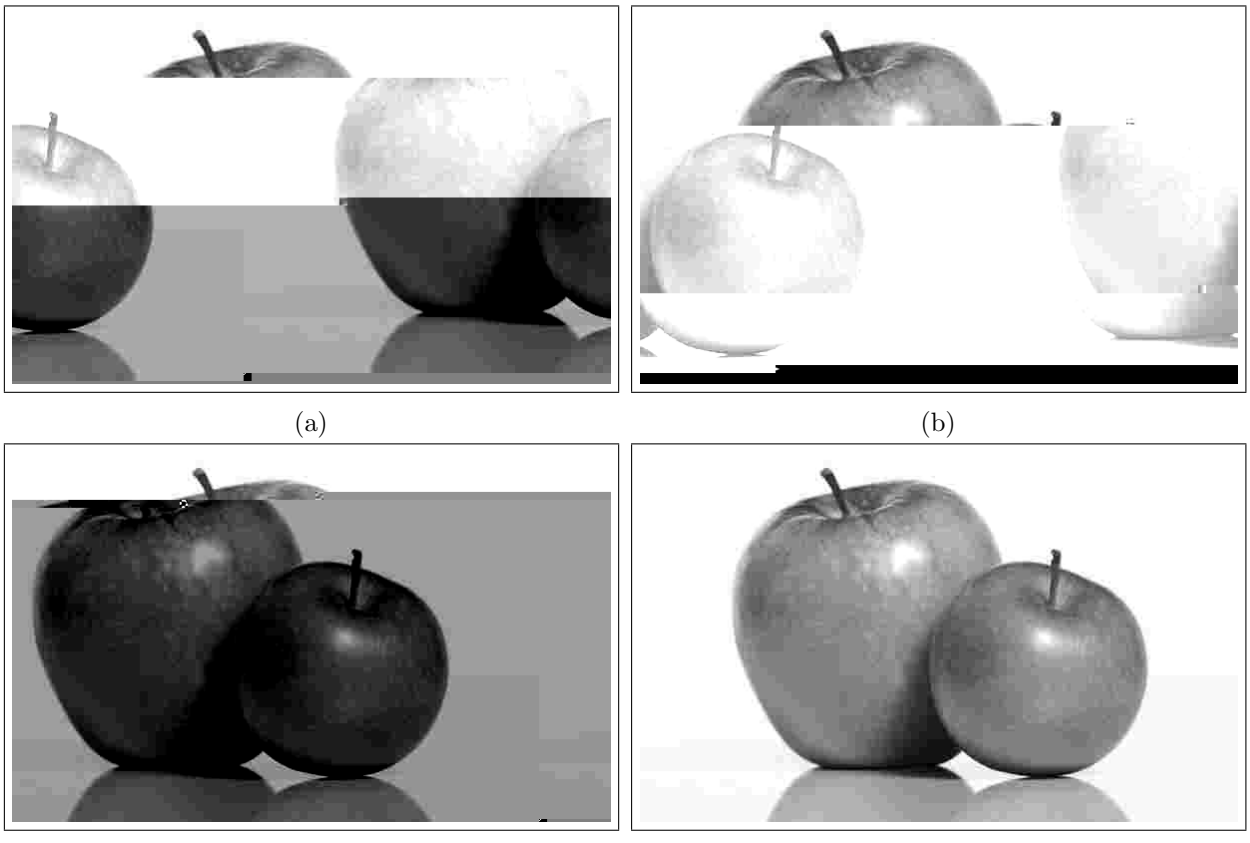


Figure 5.12: Image Receiving Performance Comparison: Scheme III (All Data Rates)



(a) (b)

(c) (d)

Figure 5.13: Image Received on Different Test Scenarios

Fig. 5.13(d). Some of the test results (log files) shows that the bytes are received correctly and in order, still they printed out incorrectly for some hardware glitches occurred while destination node is processing the bytes to print. This hardware glitches can happen to our micro-controller due

to RAM overflow, voltage fluctuations or when the system goes into halt mode. The last figure is the most commonly received image in all the tests especially for scheme III implementation for different data rates. We are only attaching the images that are distorted in the process in order to readers to have good perception of our test results. Finally we can successfully send our image via the Connect2Pi radio module which is reassuring the quality functionality of our system design and transmission schemes.

5.4.2 Different data rate on transmission scheme I and II : Test indoor (GPS)

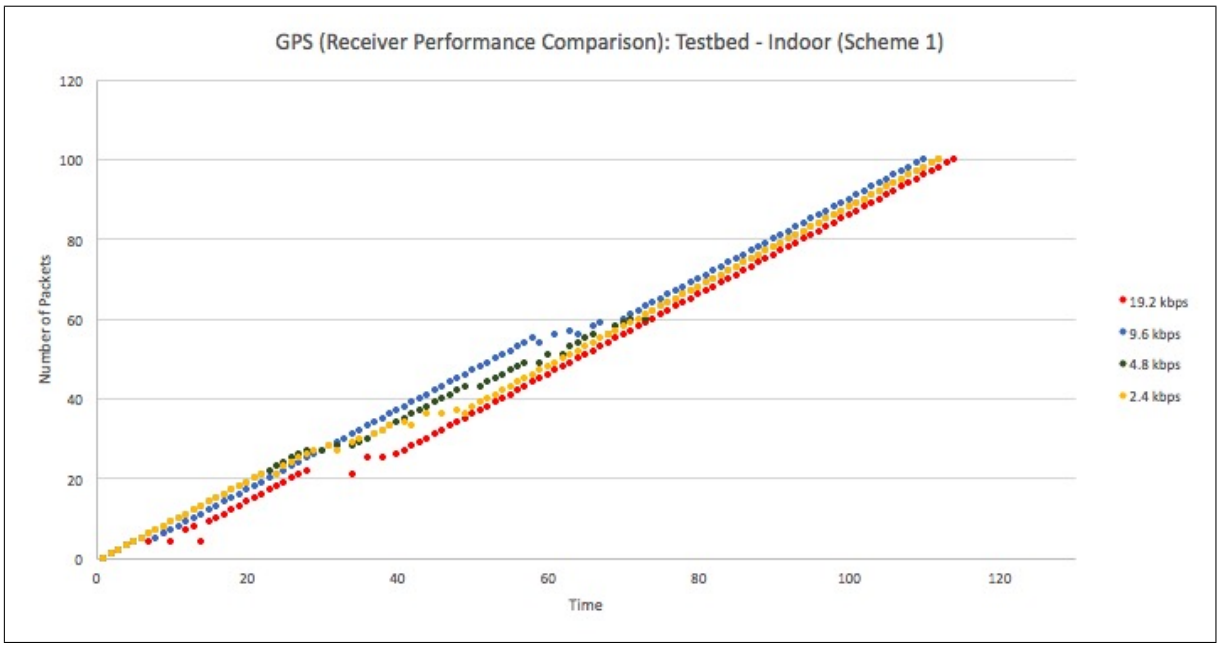
For this evaluation of performance comparison we implemented the transmission scheme I and II in the indoor test scenario. We chose the GPS co-ordinate transmission as type of service for this test and gather the results from both sender and receiver nodes. As usual we used only two nodes for this scheme I and they are assigned as sender and destination respectively. For scheme II we used three nodes which are referred as sender, relay and receiver nodes. The GPS module connected to the sender node can collect the information such as time, latitude, longitude, altitude, speed and etc. The packet size we sent from sender to receiver is 65 bytes (in scheme I) in total for each transmission. All the test node's radio in this test configured with data rate of 19.2, 9.6, 4.8 and 2.4 Kbps respectively scheme I and run our test with the following data rate of radio module on all nodes. We target to compare all the data rate available for our radio and put them to the test on indoor. We eventually tested all the data rate on both schemes and compared our results by using the graphs. By comparing the results we get a clear overview of the performance of our designed schemes with the different data rate. Table 5.2 gives us an understanding of the implemented tests we are conducting:

Table 5.2: Transmission Delay According to Various Data Rates: Indoor Test Scenario

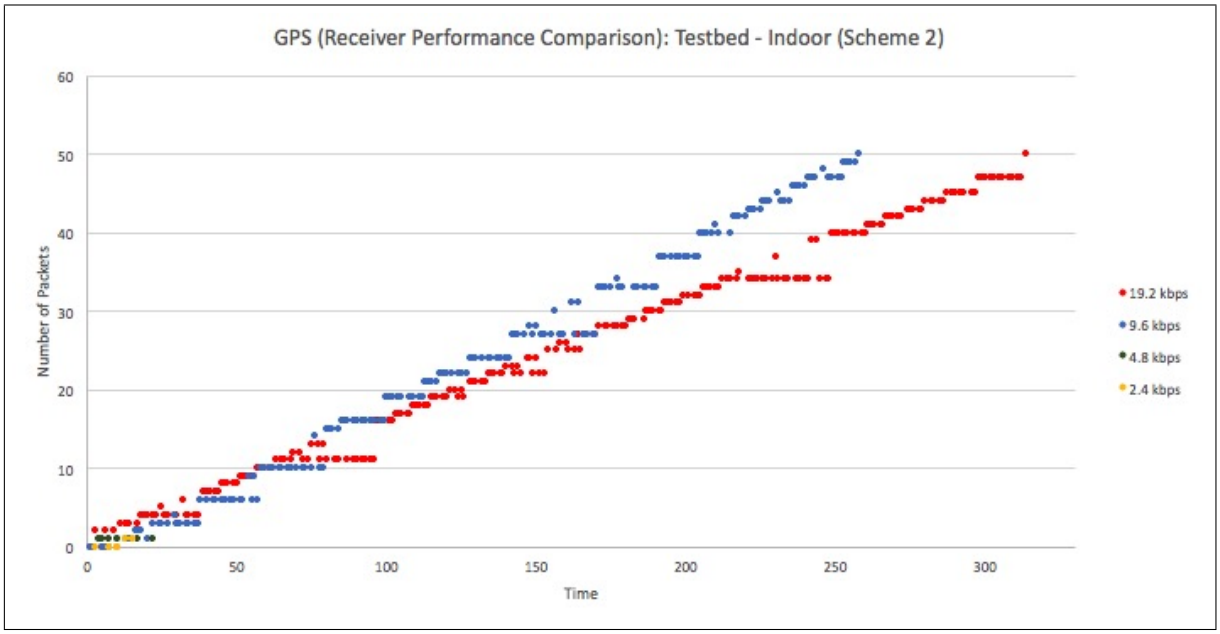
Schemes	**Data Rate**	**Packet Size**	**Transmission Delay**	**PLR**
Scheme I	19.2 Kbps	65 bytes	27.08 ms	0%
Scheme I	9.6 Kbps	65 bytes	54.17 ms	0%
Scheme I	4.8 Kbps	65 bytes	108.33 ms	4%
Scheme I	2.4 Kbps	65 bytes	216.67 ms	2%
Scheme II	19.2 Kbps	180 bytes	75 ms	14%
Scheme II	9.6 Kbps	180 bytes	150 ms	35%
Scheme II	4.8 Kbps	180 bytes	300 ms	N/A
Scheme II	2.4 Kbps	180 bytes	600 ms	N/A

After gathering data, we compare the performance of the different data rate by comparing receiving packets received on the node B for scheme I and on node C for scheme II. For the scheme I implementation, we observe that the performance of the system with all the data rate are almost similar in performance. At the start the comparison between the different data rate shows a little different than each other but once the connection got stable between the two nodes the performance of the nodes are also getting stable. As we can also see that the system on 9.6 and 4.8 Kbps does slightly better in performance than 19.2 and 2.4 Kbps data rate. Again on 19.2 Kbps it took the most time to receive all the 100 packets than rest of the data rate evaluation. Fig. 5.14(b) shows the performance results of data rate on scheme I indoor test. In this graph the radio rates are marked as red, blue, green and the yellow dots respectively.

We do the comparison of different data rate on three nodes by implementing our second transmission scheme on indoor environment. The result presented by the graph (Fig. 5.14(b)). By observing the scheme II's performance it is noticeable the data rate does not cooperate when it is lower in this scenario. Although the system performed very well on the 19.2 and 9.6 Kbps but for



(a)



(b)

Figure 5.14: Scheme I and II (GPS): Implementation on Indoor Test Scenario

the lowest two data rate the system gets exhausted quicker. On the highest data rate the system do perform quite efficiently as expected but for the 9.6 Kbps the transaction process are noticeably faster than the 19.2 Kbps data rate. This is very interesting in terms of two-hop communication protocols to adapt low data rate transmission to achieve a long distance or transmission range. Our design goal is to achieve long distance with this type of low data rate. This compares of data rate gave us good impression on our system design as it is progress towards our design goal. The PLR for all the combinations of data rates and schemes are showing on the Table 5.2. The PLR in the last two test is not applicable as the receiver could not continue to maintain the connection stability after a short time of transmission process start.

5.4.3 Different data rate on transmission scheme I and II : Test semi-open (GPS)

Similarly in this test applied the same test result as in Sec. 5.4.2 except we change our test scenario to semi-open. All other technical procedure are the same in this test as we are comparing the performance of scheme I and II according to the radio data rates available for our radio module attached to the nodes. We have two nodes in this scheme as it is scheme I and for scheme II it is three nodes. We run our application for scheme I implementation first on semi-open (university cafeteria area) with two nodes and gather our test data in log file forms. After comparing the log files we notice that our system design perform worst while the radios connected to them set to highest data rate 19.2 Kbps. It is also noticeable some packet retransmission and possible packet drop in 19.2 Kbps curve on the graph (Fig. 5.15(a)). The nodes performs almost in the same manner while they are in the lower data rates assigned for tests. Although the 9.6 Kbps data rate performs best in this case study. Our PLR for all four data rates (19.2 to 2.4 Kbps) are 7%, 0%, 2% and 0% respectively.

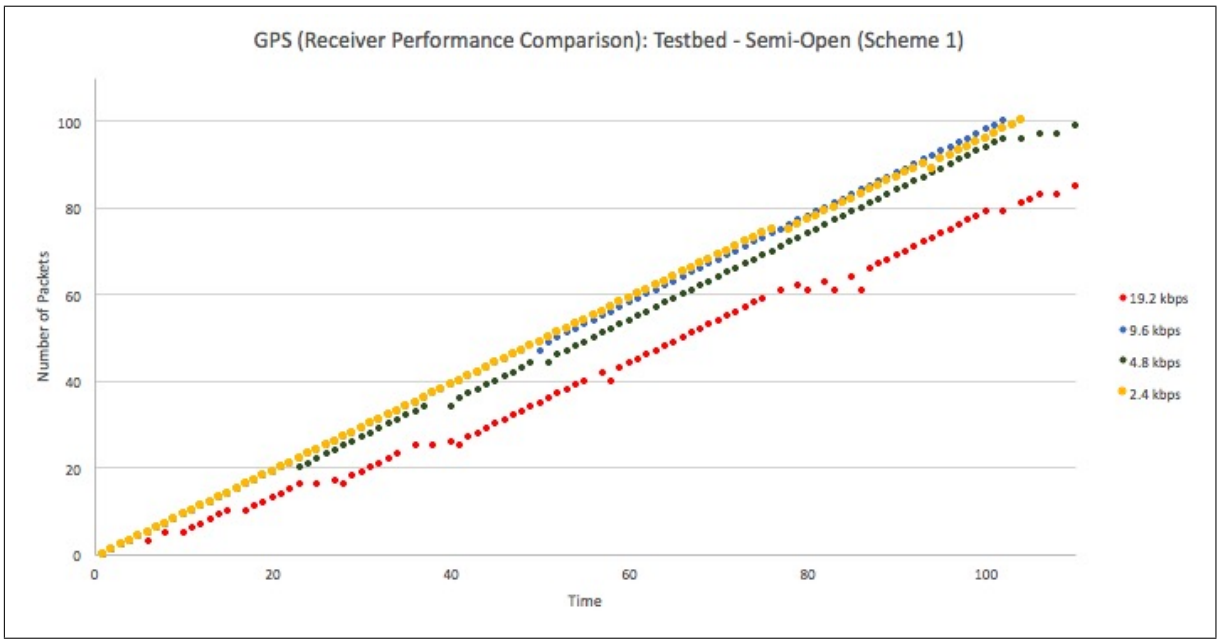
In the same test we run the scheme II and collect the experimental data from the experiments. This time we calculate the first 50 packets from our resulted log files and compare the effect of different data rate on semi-open scenario using scheme II. Our test result presented as graph in the Fig. 5.15(b). The graph presenting the data rate comparison between the three nodes shows almost the similar result that we conducted in indoor test scenario (scheme I). This time the performance of the node on 19.2 Kbps gets much higher than 9. 6 Kbps as the network delays for this data rate are much faster than the other radio data rates. As usual (same as indoor case) the system becomes exhausted quicker while running on data rate 4.8 and 2.4 Kbps. The PLR for 19.2 Kbps in this test is 20% for first 50 packets on receiver node C and for 9.6 Kbps its 35% on node C as well. The transmission delay for these tests are same as in Table 5.2

5.4.4 Different data rate on transmission scheme I and II : Test outdoor (GPS)

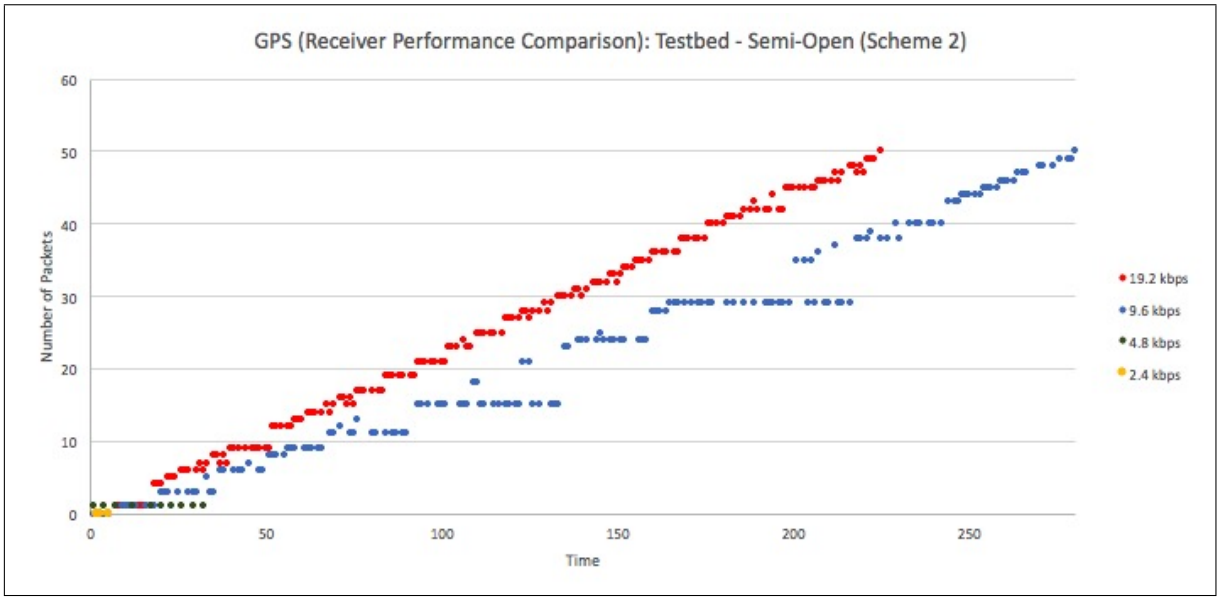
We run this final test of performance of our system design for low data rate long distance protocol design in outdoor test scenario by changing our radio data rate into four available alternatives and collect and compare the data between each other using graphs. We run both scheme I and II in this test scenario which gives us the overall performance of the system with single hop (for scheme I) and two-hop (for scheme II) transmission. From the graph (Fig. 5.16(a) we see the comparison of performance of the nodes where we notice that overall all the lower data rate enabled tests worked very well for outdoor environment except the highest data rate (19.2 Kbps) showed some weaker test results. The performance of the 19.2 Kbps data rate enabled scheme I is much lower than other tests performed on indoor and outdoor scenarios.

According to the result gathered from outdoor data rate comparison testing with scheme II, we notice an interesting change on performance of nodes. Although with the lowest data rate (2.4 Kbps) the system get exhausted but for the 4.8 Kbps data rate our system worked very well with two-hop transmission. Considering the semi-open and indoor test, outdoor test results gives better performance of the system while using lower data rates. Especially for 4.8 the systems inter-exchange the data quite faster and receiver node receives the packets in quicker manner. For the 9.6 data rate the performance is the almost similar as the other two tests but for highest data rate (19.2 Kbps) the performance got much lower than the previous two settings. The PLR for outdoor test on different data rate are as follows:

From all these tests, it is agreeable that our system design along with the transmission schemes performed better in the lower data rates in all the different types of test environment and distances. Although we only carried on with one type of service (GPS) in this particular experiment, but we



(a)



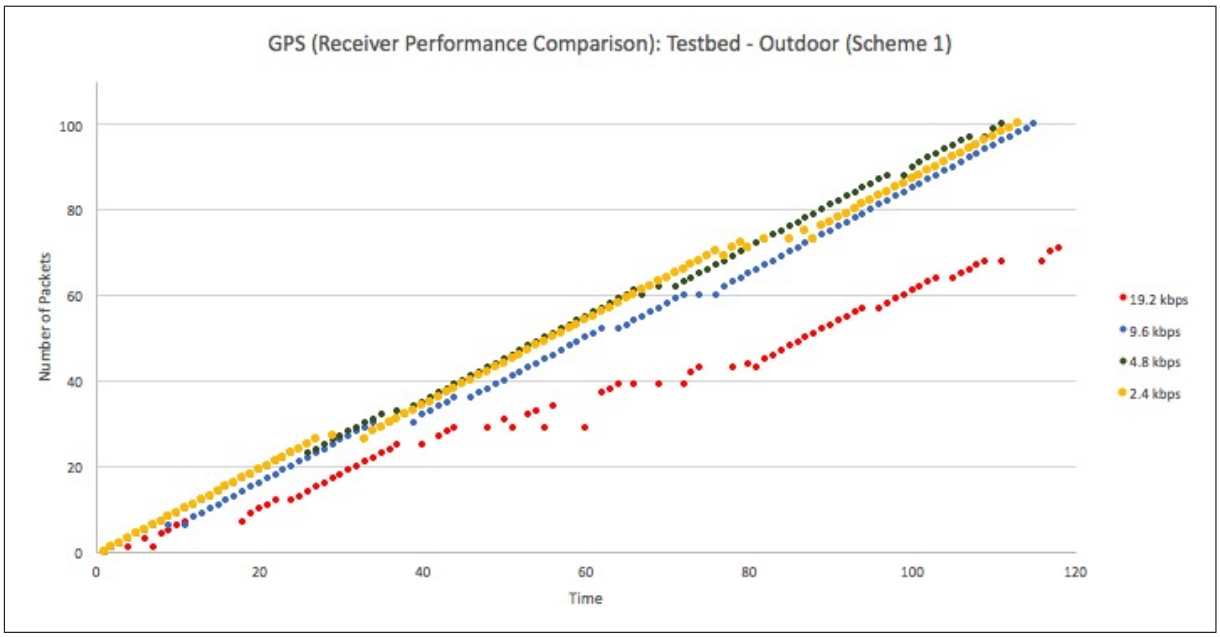
(b)

Figure 5.15: Scheme I and II (GPS): Implementation on Semi-Open Test Scenario

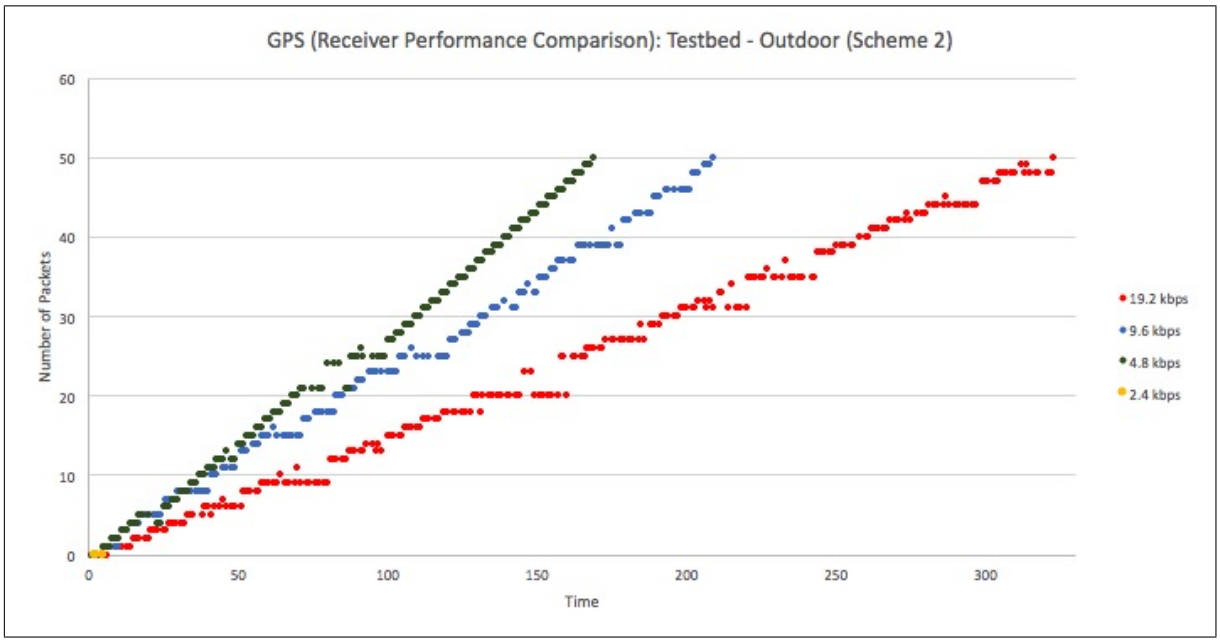
ran several other tests with different types of services in this test scenarios and the results showed the similar behaviour in those cases as well.

To summarise we gathered the data for 19.2 and 9.6 Kbps enabled results for all three tests for scheme I, and try to show a graph representation of the performance scale of our system. Figs. 5.17(a) and 5.17(b) will give the readers of this thesis report a thorough view of the collected results.

As our readers can observe that our system performs very well in the all the scenarios with scheme I and the best performance are conducted when the nodes were placed indoor. The performance of the nodes on outdoor scenario is little laggier than in other two scenarios. For 19.2 Kbps the nodess shows the similar performance for both indoor and semi-open scenarios. It is also



(a)



(b)

Figure 5.16: Scheme I and II (GPS): Implementation on Outdoor Test Scenario

noticeable that in outdoor the amount of retransmitting is quite often than other two tests. With the Fig. 5.17(b)) representing the graph for the performance comparison of the scheme I with 9.6 data rate, shows us that with the lower data rate our system performance in quite stable and linear fashion and it is very similar performance for any test scenarios. Although the performance evaluation shows that the node with 9.6 Kbps data rate in outdoor scenario is taking slightly longer time to receive a packet and send an ACK / NACK accordingly, still there is no sign of packet drop noticeable in this curve and the test results. The flow of the system is very stable in every test with lower data rate packet transmission.

Table 5.3: PLR According to Different Data Rates: Outdoor Test Scenario

Schemes	Data Rate	PLR
Scheme I	19.2 Kbps	14%
Scheme I	9.6 Kbps	3%
Scheme I	4.8 Kbps	2%
Scheme I	2.4 Kbps	0%
Scheme II	19.2 Kbps	12%
Scheme II	9.6 Kbps	19%
Scheme II	4.8 Kbps	4%
Scheme II	2.4 Kbps	N/A

5.4.5 Additional test results

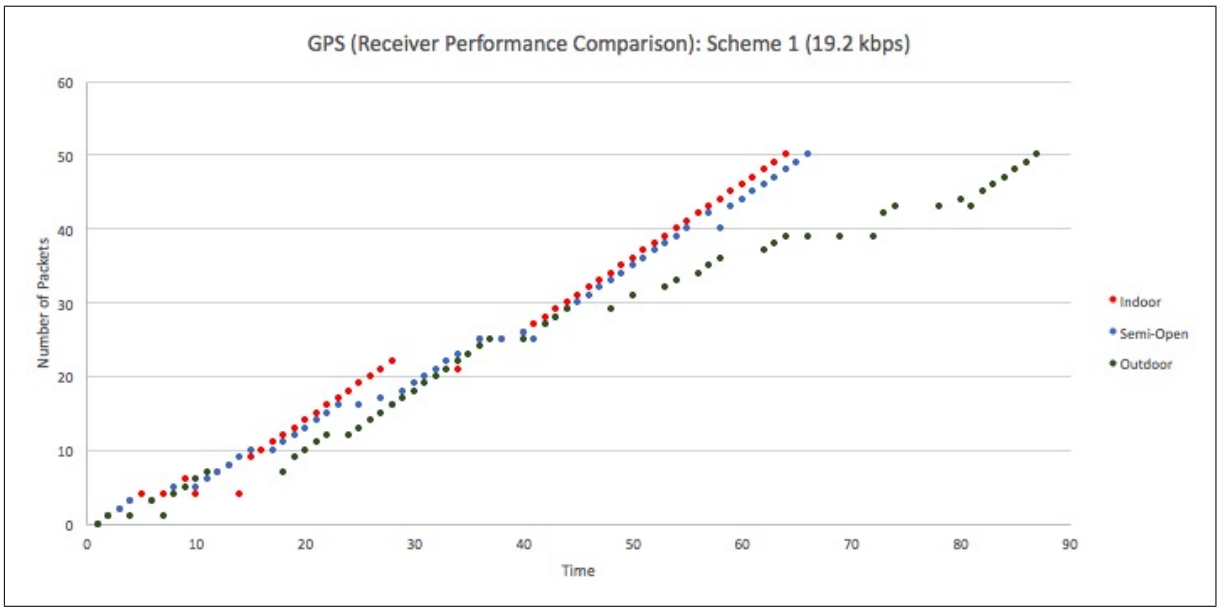
In this quick test we obtain our experimental data collection procedure on all the scenarios and choose to check the performance of the designed system by implementing the service type temperature for every scenarios combination with transmission schemes. The temperature service has the lowest size of the packet in each transaction which is about 25 bytes in total in each data frame. We are considering that this service type can the fastest type of service that can be implemented on any scenario. We gathered our data on every test scenario and the graph shows us the performance evaluation of all the packet received time on destination node in each scenario.

It is very interesting to see that our temperature service is working exactly similar pattern for each and every scenario. We consider, this is happening due to the short size of the total packet size in this cases has an advantage of sending and receiving quickly to avoid the network delay in between the sender and destination nodes. The graph in Fig. 5.18(a) representing the results for this case study.

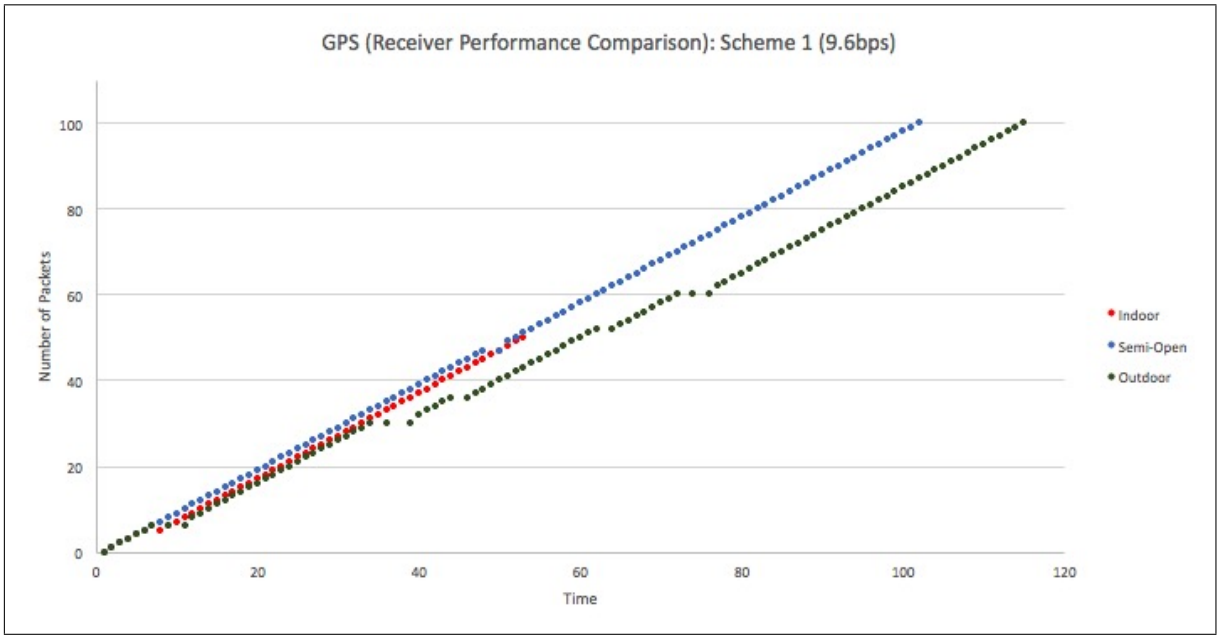
According to the graph in Fig. 5.18(b), we ran another experiment by implementing scheme II on the all the test scenarios. We again chose to transfer temperature data among two running nodes. This graph shows that packets are received in very different pattern than of scheme I. Many packet drops are noticeable and retransmission of packet is also easily viewable with this graph. We are considering the reason behind this huge change between the scheme is that our radio module need to listen to all the incoming packets sent from sender or relay nodes. The technique we implemented to collect temperature on sender by sending ER_CMD command (followed by ACK) and listen for the echo generated by the radio. The temperature will be echoed back to the sender which the sender collects and send to the relay node to forward. This echo can also be heard by the relay node and in the semi-open and indoor situation the receiver also hears these echoes. This creates a massive amount of overhearing while scheme II is running with three nodes. Although all the data in the graph looks scatter and disoriented but it is not hard to notice that the receiver still keeps its connection stability and stable network delay in semi-open test situation.

5.5 Summary

These valuable experimental results proof that our system has a promising performance in the different test scenarios and it can be a positive step towards the research field for WSN. Our research outcomes can give the readers to have good perception of an embedded system design.



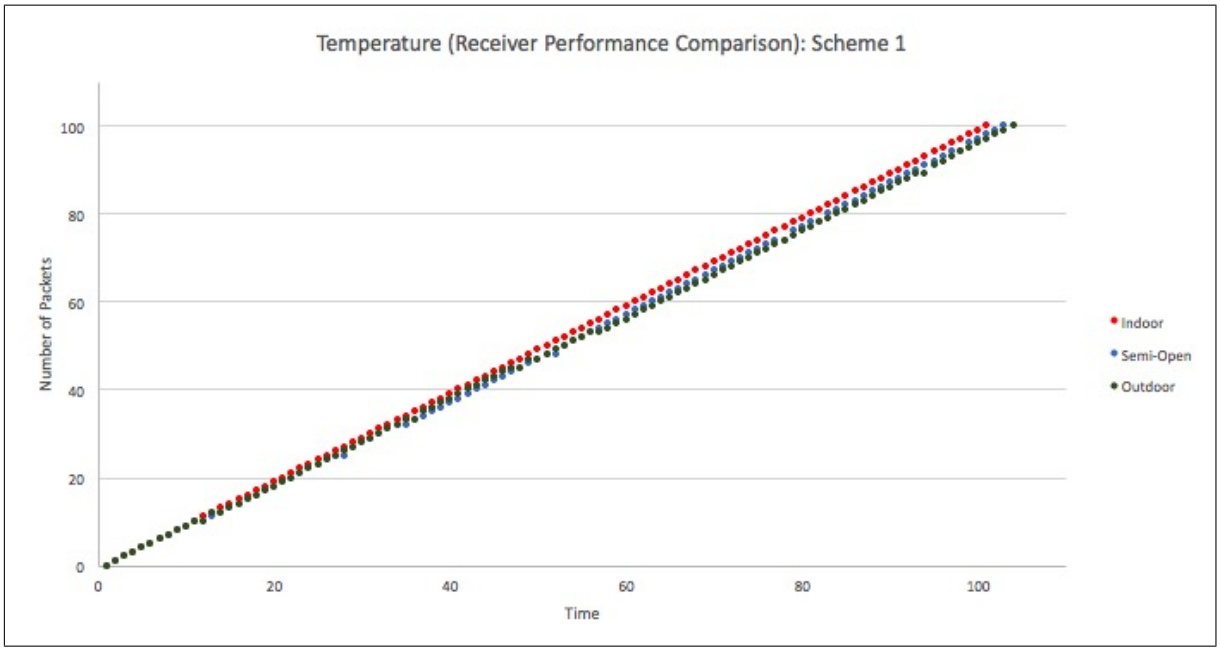
(a)



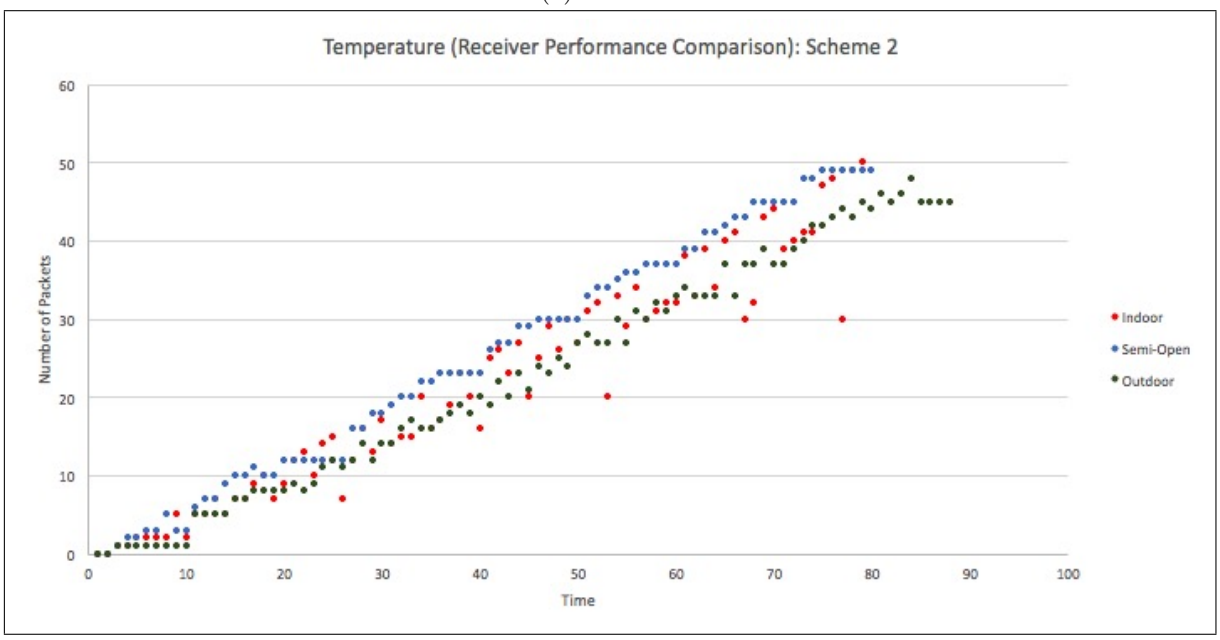
(b)

Figure 5.17: Scheme I and II (GPS): Implementation on All Test Scenarios

We also have good perspective of the performance evaluation of our designed system in a real-life scenario based tests. We observe that with lower data rate of the radios we can achieve longer distance of packet transmission. The nodes perform better when they are placed further from each other (within the transmission range of the radios) and better with the lower data rate. Additionally the height of the radios are also important in the low data rate packet transmission. In our test scenarios, height of our radios was only 1.8 m above the ground. By raising the radios higher than that always gives better results on the node's performances. With the scheme III implementation on the system, we observe the quality of the service type gets finer on the low data rate packet transmission at 868 MHz band and the reliability of our transmission process becomes more improved with this scheme.



(a)



(b)

Figure 5.18: Scheme I and II (Temperature): Implementation on All Test Scenarios

Chapter 6

Conclusions and Future Work

In this emerging world of WSN, designing the anatomy of an embedded system is a very sophisticated field of work. Not only a developer needs to construct the architecture of the design but also has to implement a right protocol that can co-operate with that system. The functionality of a wireless networking system is very much depended on transmission connectivity, bit balancing, packet constructing flexibility, transmission precision and many other factors of communication network. The study of WSN is continually growing, as these networks have the advantage of easy deployment for a number of different applications. Protocols, Routing and Management of this network focus on wireless sensor and their operation, routing, data frame handling, time management and synchronisation and etc. The quality of packets transferring schemes between the nodes can have an impact on the connection stability and longevity of the entire network. The range of WSN is also an essential factor for a network model design. By using a low frequency band such as 868 MHz ISM band (to transmit low data rate packets) a developer can accomplish this long range of transmission which can benefit the industry of WSN development.

6.1 Conclusion

We studied the concept behind developing an embedded system and design principal of building and implementing a protocol design in this project which can transmit and receive low data rate packets in the long range of distance. By utilising different ideas behind creating a working scheme, we understood and learned the techniques of maintaining a better network connectivity to increase performance of a WSN system. With our embedded design together with our designed transmission schemes, we transferred low data rate packets for several types of services between the nodes by using the 868 MHz enabled radio modules. We obtained comprehensive experiments on real-life based test scenarios and gathered our experimental data of both ends of the network. These results are then evaluated by comparing the performance of nodes in different combinations of test scenarios combined with our designed transmission schemes. This particular thesis allowed us to gain a good view into the complexities that lies into the sensor node design and their real-life implementations in WSN, which can help towards the development of this noble field of research.

6.2 Contribution

Our contribution in this project consists of these major parts:

- We have designed an embedded system with three nodes (transmitter, forwarder and the receiver) which can transmit low data rate packets using the radio modules which operate on 868 MHZ ISM band.
- We implemented four different types of services in our system in order to transfer data packets via the network such as randomly generated bytes, image, GPS information and temperature. We constructed the data frames for these services and used them for services in our designed schemes.
- Then we proposed three transmission schemes (Scheme I, II and II) which can maintain the balance between nodes while in ongoing transmissions. Using the scheme I, we sent low data rate packets over long distance in real-time. By implementing scheme II, we relay the low data rate packets by using the forwarding node in between the transmitting and receiving system to reach further distance. We developed the scheme I and implemented a new transmission scheme III which gives us a better quality of data transmission for single hop network. Scheme III allows the first phrase to only send fresh packets and the second phrase to re-collect the dropped packet on receiver node.
- Finally we perform numerous real-life based tests in our various test scenarios such as indoor, semi-open and outdoor (residential area and open field) scenarios and collect the experimental data from these tests. Our designed system can reach up to 900 m in distance and keeps stable transmission rate efficiently throughout the transmission period. In addition, we summarise by extracting the data collected and put them in the graph models, we evaluate the performance of the entire network along with the transmission schemes implementation on the systems, which gives us a thorough overview of our working protocol's performance in many given conditions. This gives the readers of this thesis a detailed perception of our system's performance evaluation.

6.3 Future Work

We include several potential work that can be investigated in the future. These future topics are mentioned below:

- The radio's RSSI can be used to dynamically choose the closest node as the forwarding / relay node.
- A method can be implemented in the protocol where a receiver can determine the missing packets by calculating the information from the previous and the next packets.
- Can also implement a mechanism to request other nodes in the network to send their GPS location, ambient temperature, etc.

Bibliography

[1] ETSI, *Short Range Devices*, [online] Available: http://www.etsi.org/website/Technologies/ShortRangeDevices.aspx

[2] ITU, "*ITU Page on Definitions of ISM Bands*", [online] Available: http://www.itu.int/net/ITU-R/terrestrial/faq/index.html#g001

[3] ITU, "*ITU Page on Radio Regulations*", *ITU Radio Regulations, Frequencies, Frequency allocations, Table of Frequency Allocations*, [online] Available: http://www.itu.int/en/publications/Pages/default.aspx, chap. 2, art. 5, sec. 4.

[4] Wikipedia, "*Ultra High Frequency*", [online] Available: https://en.wikipedia.org/wiki/Ultra_high_frequency#cite_note-ieee-1

[5] R. Nilsson, "*What is the Difference Between 2.4 GHz and 5 GHz Wireless in Industrial Application*", [online] Available: http://www.connectblue.com/press/articles/what-is-the-difference-between-24ghz-and-5ghz-wireless-lan-in-industrial-applications/

[6] Wikipedia, "*868 MHz frequency band*", [online] Available: https://en.wikipedia.org/wiki/800_MHz_frequency_band

[7] LPRS, "*Connect2Pi USB Dongle*", [online] Available: http://www.lprs.co.uk/wireless-modules/era-connect2-pi.html

[8] Raspberry Pi Foundation, "*Raspberry Pi Official Website*", [online] Available: https://www.raspberrypi.org

[9] Python Software Foundation, "*The Python Standard Library*", [online] Available: https://docs.python.org/2/library/binascii.html

[10] J. Oller, I. Demirkol, J. Casademont, J. Paradells, G. U. Gamm and L. Reindl, "*Has Time Come to Switch From Duty-Cycled MAC Protocols to Wake-Up Radio for Wireless Sensor Networks?*", *IEEE/ACM Transactions on Networking*, pp. 674-687, vol. 24, no. 2, Apr 2016.

[11] M. Woehrle, M. Bor and K. Langendoen, "*868 MHz: A noiseless environment, but no free lunch for protocol design*", *Networked Sensing Systems (INSS), Ninth International Conference on, Antwerp*, pp. 1-8, 2012.

[12] Z. Aqachmar, P. Acco, J. Y. Fourniols, G. Auriol and C. Escriba, "*Why don't we use free 868 MHz band for geolocation?*", *Electrical Electronics Engineers in Israel (IEEEI), IEEE 27th Convention of, Eilat*, pp. 1-4, 2012.

[13] D. Dobrilovic, B. Odadzic and Z. Stojanov, "*Approach in planning the deployment of 868 MHz RF modules in WSN scenarios*", *Applied Computational Intelligence and Informatics (SACI), IEEE 10th Jubilee International Symposium on, Timisoara*, pp. 387-392, May 2015.

[14] G. Håland, E. R. Jahren and J. K. Joreid, "*GPS-basert posisjonerings- og identifiseringssystem for fritidsbåter*", Bachelor Thesis Project, Department of Information and Communication Technology, University of Agder, May, 2013

[15] J. Drozdowicz and G. Mazurek, "*IEEE 802.15.4 compliant in-building wireless sensor network*", *Signal Processing Symposium (SPSympo), Debe*, pp. 1-5, 2015.

[16] OFCOM, "*OFCOM IR 2030 - License Exempt Short Range Devices*", [online] Available: `http://stakeholders.ofcom.org.uk/binaries/spectrum/spectrum-policy-area/spectrum-management/research-guidelines-tech-info/interface-requirements/IR_2030.pdf`

[17] M. Loy, R. Karingattil and L. Williams, *ISM-Band and Short Range Device Regulatory Compliance Overview*, [online] Available: `http://www.ti.com/lit/an/swra048/swra048.pdf`

[18] BB Electronics Mfg. Co. Inc., "*The Ten Commandments of Wireless Communications*", [online] Available: `http://www.bb-elec.com/Learning-Center/All-White-Papers/Wireless-Cellular/10-Commandments-of-Wireless-Communications.aspx`, 2009.

[19] Adafruit, "*Adafruit Ultimate GPS Breakout Version 3*", [online] Available: `https://www.adafruit.com/products/746`

[20] Future Technology Devices International Ltd, "*FTDI D2XX Driver*", [online] Available: `http://www.ftdichip.com/Drivers/D2XX.htm`

Appendices

We have several python codes to operate each nodes in the transmission system and many corresponding log files collected from the test scenarios, which is represented in the Table 6.1. This table gives us an overview of amount of codes and sample logs collected in our thesis work.

Table 6.1: Application Codes and Log Files on Nodes

	Node A	**Node B**	**Node C**
Scheme	I, II, III	I, II, III	II
Codes: System Start	1	1	1
Codes: Random Bytes	2	2	1
Codes: Image	3	3	1
Codes: GPS	2	2	1
Codes: Temperature	2	2	1
Corresponding Log Files	About 106	About 40	About 44

Instead of attaching all the files, we give some of the sample corresponding log files and codes in this chapter. All the application codes and the log files are available for the readers on request.

- In Appendix A, we add the log file which is collected on the receiver node after running the temperature service type with scheme I implementation in 19.2 Kbps data rate. This is a sample log file of experimental result in indoor test scenario.
- In Appendix B, we attach the collected GPS information on receiver node by implementing scheme I. The test scenario is outdoor and the data rate of the radios were set to 9.6 Kbps while conducting this particular experiment.
- In Appendix C, we include the application code that runs the image sending procedure on the sender node in scheme III implementation.

Appendix A

Log File: Temperature (Indoor, 19.2 Kbps)

2016-04-23 18:25:16.933880	0111	0A	0B	0B	0000	31.5	b001000
2016-04-23 18:25:19.144749	0111	0A	0B	0B	0001	31.8	b000100
2016-04-23 18:25:21.336153	0111	0A	0B	0B	0002	31.8	b000100
2016-04-23 18:25:23.539305	0111	0A	0B	0B	0003	31.8	b000100
2016-04-23 18:25:25.813566	0111	0A	0B	0B	0004	31.8	b000100
2016-04-23 18:25:28.010546	0111	0A	0B	0B	0005	31.8	b000100
2016-04-23 18:25:30.202968	0111	0A	0B	0B	0006	31.8	b000100
2016-04-23 18:25:32.394630	0111	0A	0B	0B	0007	31.8	b000100
2016-04-23 18:25:34.670210	0111	0A	0B	0B	0008	31.5	b001000
2016-04-23 18:25:36.992055	0111	0A	0B	0B	0009	31.5	b001000
2016-04-23 18:25:39.207430	0111	0A	0B	0B	0010	31.5	b001000
2016-04-23 18:25:41.398059	0111	0A	0B	0B	0011	31.5	b001000
2016-04-23 18:25:43.590086	0111	0A	0B	0B	0012	31.5	b001000
2016-04-23 18:25:45.874884	0111	0A	0B	0B	0013	31.5	b001000
2016-04-23 18:25:48.067730	0111	0A	0B	0B	0014	31.5	b001000
2016-04-23 18:25:50.265716	0111	0A	0B	0B	0015	31.5	b001000
2016-04-23 18:25:52.552029	0111	0A	0B	0B	0016	31.5	b001000
2016-04-23 18:25:54.870941	0111	0A	0B	0B	0017	31.5	b001000
2016-04-23 18:25:57.178892	0111	0A	0B	0B	0018	31.5	b001000
2016-04-23 18:25:59.374572	0111	0A	0B	0B	0019	31.5	b001000
2016-04-23 18:26:01.605615	0111	0A	0B	0B	0020	31.5	b001000
2016-04-23 18:26:03.916733	0111	0A	0B	0B	0021	31.1	b000100
2016-04-23 18:26:06.110397	0111	0A	0B	0B	0022	31.1	b000100
2016-04-23 18:26:08.304398	0111	0A	0B	0B	0023	31.1	b000100
2016-04-23 18:26:10.534791	0111	0A	0B	0B	0024	31.1	b000100
2016-04-23 18:26:12.863555	0111	0A	0B	0B	0025	31.5	b001000
2016-04-23 18:26:15.064106	0111	0A	0B	0B	0026	31.5	b001000
2016-04-23 18:26:17.278893	0111	0A	0B	0B	0027	31.1	b000100
2016-04-23 18:26:19.483770	0111	0A	0B	0B	0028	31.1	b000100

Appendix B

Log File: GPS (Outdoor, 9.6 Kbps)

2016-04-27 06:10:15.392950	0110	0A	0B	0B	0000	2016-04-27T14:33:23.000Z	58.333	8.5775	40.8	0.17	b000110
2016-04-27 06:10:16.776205	0110	0A	0B	0B	0001	2016-04-27T14:33:24.000Z	58.333	8.5775	40.9	0.35	b111110
2016-04-27 06:10:18.218216	0110	0A	0B	0B	0002	2016-04-27T14:33:25.000Z	58.333	8.5775	41.0	0.34	b010001
2016-04-27 06:10:19.658882	0110	0A	0B	0B	0003	2016-04-27T14:33:26.000Z	58.333	8.5775	41.1	0.27	b110100
2016-04-27 06:10:21.754619											
2016-04-27 06:10:22.039753	0110	0A	0B	0B	0003	2016-04-27T14:33:26.000Z	58.333	8.5775	41.1	0.27	b110100
2016-04-27 06:10:24.130408											
2016-04-27 06:10:24.262525	0110	0A	0B	0B	0005	2016-04-27T14:33:29.000Z	58.333	8.5775	41.5	0.20	b100100
2016-04-27 06:10:26.360863											
2016-04-27 06:10:26.629831	0110	0A	0B	0B	0005	2016-04-27T14:33:29.000Z	58.333	8.5775	41.5	0.20	b100100
2016-04-27 06:10:28.748493	0110	0A	0B	0B	0006	2016-04-27T14:33:32.000Z	58.333	8.5775	41.5	0.19	b011101
2016-04-27 06:10:30.200390	0110	0A	0B	0B	0007	2016-04-27T14:33:33.000Z	58.333	8.5775	41.5	0.15	b011101
2016-04-27 06:10:31.582196	0110	0A	0B	0B	0008	2016-04-27T14:33:34.000Z	58.333	8.5775	41.5	0.09	b001010
2016-04-27 06:10:32.979330	0110	0A	0B	0B	0009	2016-04-27T14:33:35.000Z	58.333	8.5775	41.5	0.18	b000010
2016-04-27 06:10:34.393750	0110	0A	0B	0B	0010	2016-04-27T14:33:36.000Z	58.333	8.5775	41.5	0.22	b010111
2016-04-27 06:10:36.491602											
2016-04-27 06:10:36.760604	0110	0A	0B	0B	0010	2016-04-27T14:33:36.000Z	58.333	8.5775	41.5	0.22	b010111
2016-04-27 06:10:38.881124	0110	0A	0B	0B	0011	2016-04-27T14:33:38.000Z	58.333	8.5775	41.7	0.15	b011110
2016-04-27 06:10:40.335568	0110	0A	0B	0B	0012	2016-04-27T14:33:39.000Z	58.333	8.5775	41.9	0.05	b011110
2016-04-27 06:10:41.726626	0110	0A	0B	0B	0013	2016-04-27T14:33:40.000Z	58.333	8.5775	42.1	0.04	b100101
2016-04-27 06:10:43.202053	0110	0A	0B	0B	0014	2016-04-27T14:33:41.000Z	58.333	8.5775	42.2	0.08	b000000
2016-04-27 06:10:44.592752	0110	0A	0B	0B	0015	2016-04-27T14:33:42.000Z	58.333	8.5775	42.2	0.14	b100010
2016-04-27 06:10:45.986404	0110	0A	0B	0B	0016	2016-04-27T14:33:43.000Z	58.333	8.5775	42.2	0.17	b001100
2016-04-27 06:10:48.084684											
2016-04-27 06:10:48.354630	0110	0A	0B	0B	0016	2016-04-27T14:33:43.000Z	58.333	8.5775	42.2	0.17	b001100
2016-04-27 06:10:50.448441											
2016-04-27 06:10:50.606897	0110	0A	0B	0B	0017	2016-04-27T14:33:45.000Z	58.333	8.5775	42.2	0.21	b110101
2016-04-27 06:10:50.792504	0110	0A	0B	0B	0016	2016-04-27T14:33:43.000Z	58.333	8.5775	42.2	0.17	b001100
2016-04-27 06:10:52.908738	0110	0A	0B	0B	0018	2016-04-27T14:33:47.000Z	58.333	8.5775	42.2	0.08	b111111
2016-04-27 06:10:54.004982	0110	0A	0B	0B	0019	2016-04-27T14:33:48.000Z	58.333	8.5775	42.2	0.31	b101111
2016-04-27 06:10:55.203087	0110	0A	0B	0B	0020	2016-04-27T14:33:49.000Z	58.333	8.5775	42.5	0.13	b101101
2016-04-27 06:10:56.362553	0110	0A	0B	0B	0021	2016-04-27T14:33:50.000Z	58.333	8.5775	42.7	0.03	b101001
2016-04-27 06:10:57.486746	0110	0A	0B	0B	0022	2016-04-27T14:33:51.000Z	58.333	8.5775	42.9	0.05	b011110
2016-04-27 06:10:58.620574	0110	0A	0B	0B	0023	2016-04-27T14:33:52.000Z	58.333	8.5775	43.0	0.01	b110001

Appendix C

Code: Scheme III Implementation for Image Service Type

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import time as t
import serial
import thread
import os
import random
import math
import struct,binascii
import csv
import datetime
from random import randrange


def listen(name,timeout):
    listen_txt = ''
    exitflag = 0
    while True:
        while ser.inWaiting()>0:
            t.sleep(0.005)
            listen_txt += ser.read(1)
        if listen_txt !='':
            print listen_txt
            listen_txt = ''
        if (exitflag == 1):
            exitflag = 0
            thread.exit()
# end of listen thread function

def __init__(self):
    self.seq = 0
def getNextSeqNo(self):
    self.seq +=1
    return format(self.seq,'010d')

def CONNECTION_on(dst_addr):
    acknowledge = str(packet_type[5]+src_addr+dst_addr+real_dst_addr)#+mock_seq_num
        +mock_payload+mock_crc)
    return acknowledge

def CRC_calculate(payload):
    k = binascii.crc32(payload)
    h = abs(k) % (1<<32)
    s = bin(abs(h))
    crc="".join(value for index, value in enumerate(s[:26]) if index % 2 == index %
        4 ==1)
```

```
46      return crc
47
48  def PACKET_send(payload):
49      i = 0
50      while i < 10000:
51          seq_num = str(format(i,'010d'))
52          k = binascii.crc32(payload)
53          h = abs(k) % (1<<32)
54          s = bin(abs(h))
55          crc="".join(value for index, value in enumerate(s[:26]) if index % 2 ==
                index % 4 ==1)
56          send = (packet_type[2]+src_addr+dst_addr+seq_num+serv_type[0]+payload+crc)
57          i = i+1
58          return send
59
60  def PACKET_retransmit(payload):
61      resend = str(packet_type[2]+src_addr+dst_addr+seq_num_recv+payload+crc)
62      return resend
63
64  transmission_random = [4,7]
65  random_index = randrange(0,len(transmission_random))
66
67  # configure the serial connections (the parameters differs on the device you are
        connecting to)
68  ser = serial.Serial(
69      port='/dev/ttyUSB0',        # Either something COM1 for Windows or /dev/ttyUSB0
            for Linux/RasPi
70      baudrate= 19200,
71      parity=serial.PARITY_NONE,
72      stopbits=serial.STOPBITS_ONE,
73      bytesize=serial.EIGHTBITS,
74      writeTimeout = 0,
75      timeout = 0,
76      rtscts=False,
77      dsrdtr=False,
78      xonxoff=False
79  )
80
81  f = open('/home/pi/Desktop/Wait_and_Send_A/services/image_B/apple_B.jpg','rb')
82  byte_read_image = f.read()
83  t.sleep(0.5)
84  nf = open('/home/pi/Desktop/Wait_and_Send_A/services/image_B/temp_image_byte_B.txt'
        ,'wb')
85  nf.write(byte_read_image)
86  t.sleep(0.005)
87  f.close()
88  nf.close()
89  rf = open('/home/pi/Desktop/Wait_and_Send_A/services/image_B/temp_image_byte_B.txt'
        ,'rb')
90  byte = rf.read()
91  t.sleep(0.5)
92
93  #packet_type = [invite,ready, ACK,  NACK,rand_bytes,image, GPS, temperature]
94  packet_type = ['0000','0001','0010','0011','0100','0101','0110','0111','1111']
95  src_addr = "0A"
96  dst_addr = "0B"
97  seq_num = format(0,'04d')
98  mock_seq_num = '0000'
99  mock_payload = format(0,'159d')
100 mock_crc = format(0,'07d')
101
102 real_dst_addr = "0B"
103 input = CONNECTION_on(real_dst_addr) # packet_type[5] (image) with conn. on
```

```
104 ser.write(input)
105 t.sleep(1.5)
106
107 os.system("clear")
108 print 'Image is Sending to ', real_dst_addr
109
110 input = 1
111 exitflag =0
112
113 i = 0
114 a = 0
115 x = 0
116 n = 200
117
118 packet_type_sent_array = []
119 seq_num_array = []
120 payload_array = []
121 crc_array = []
122 src_addr_array = []
123 dst_addr_array = []
124 real_dst_addr_array = []
125 time_sent_array = []
126 total_packet_send = 0
127
128 packet_type_recv_array =[]
129 src_addr_recv_array = []
130 dst_addr_recv_array = []
131 real_dst_addr_recv_array = []
132 ack_recv_array = []
133 seq_num_recv_array = []
134 time_recv_array = []
135
136 while 1:
137     try:
138         numbytes = ser.inWaiting()
139         msg = ''
140         msg += ser.readline(17)
141         t.sleep(0.001)
142         packet_type_recv = msg[:4]
143         src_addr_recv = msg[4:6]
144         dst_addr_recv = msg[6:8]
145         real_dst_addr_recv = msg[8:10]
146         ack_recv = msg[10:13]
147         seq_num_recv = msg[13:17]
148
149         time_recv = datetime.datetime.today()
150         packet_type_recv_array.insert(n,packet_type_recv)
151         packet_type_recv_array = packet_type_recv_array[-n:]
152         src_addr_recv_array.insert(n,src_addr_recv)
153         src_addr_recv_array = src_addr_recv_array[-n:]
154         dst_addr_recv_array.insert(n,dst_addr_recv)
155         dst_addr_recv_array = dst_addr_recv_array[-n:]
156         real_dst_addr_recv_array.insert(n, real_dst_addr_recv)
157         real_dst_addr_recv_array = real_dst_addr_recv_array[-n:]
158         ack_recv_array.insert(n, ack_recv)
159         ack_recv_array = ack_recv_array[-n:]
160         seq_num_recv_array.insert(n, seq_num_recv)
161         seq_num_recv_array = seq_num_recv_array[-n:]
162         time_recv_array.insert(n,time_recv)
163
164         with open('/home/pi/Desktop/Wait_and_Send_A/services/image_B/
                packets_recv_frm_BtoA.csv','w') as tf:
165             writer = csv.writer(tf,delimiter='\t');
```

```
166                 writer.writerows(zip(time_recv_array,packet_type_recv_array,
                        src_addr_recv_array,dst_addr_recv_array,
                        real_dst_addr_recv_array,seq_num_recv_array))
167
168         if packet_type_recv == packet_type[3]:
169             if seq_num_recv in seq_num_array:
170                 os.system("clear")
171                 t.sleep(0.005)
172                 v = seq_num_array.index(seq_num_recv)
173                 print 'NACK received. Retransmitting'
174                 l = binascii.crc32(payload_array[v])
175                 m = abs(l) % (1<<32)
176                 p = bin(abs(m))
177                 crc_retransmit = "".join(value for index, value in enumerate(p
                        [:26]) if index % 2 == index % 4 ==1)
178                 input = str(packet_type[5]+src_addr+dst_addr+real_dst_addr+
                        seq_num_recv+payload_array[v]+crc_retransmit)
179                 ser.write(input)
180                 print 'NACK response : ', input
181                 t.sleep(1)
182                 total_packet_send = total_packet_send + 1
183                 print 'Sent ', total_packet_send, ' time', '\n'
184                 random_value = transmission_random[random_index]
185                 while total_packet_send == random_value:
186                     print 'Connection Lost / System Exhausted. Re-establishing
                            Connection','\n'
187                     t.sleep(0.5)
188                     tf.close()
189                     input1 = 'ER_CMD#B3'
190                     ser.write(input1)
191                     t.sleep(0.005)
192                     input2 = 'ACK'
193                     t.sleep(0.005)
194                     # Radio changed to default
195                     execfile('/home/pi/Desktop/Wait_and_Send_A/main.py')
196                 pass
197             pass
198         if packet_type_recv == packet_type[0]:
199             if src_addr_recv == dst_addr:
200                 if dst_addr_recv == src_addr:
201                     input1 = 'ER_CMD#B3'
202                     ser.write(input1)
203                     t.sleep(0.005)
204                     input2 = 'ACK'
205                     t.sleep(0.005)
206                     # Radio changed to default
207                     execfile('/home/pi/Desktop/Wait_and_Send_A/main.py')
208         t.sleep(0.05)
209         os.system("clear")
210         print 'Transaction in Progress...'
211         total_packet_send = 0
212         seq_num = str(format(i,'04d'))
213         b = a + 159
214         hexadecimal = binascii.hexlify(byte)
215         payload = str(hexadecimal[a:b])
216         if payload != '':
217             a = b
218             k = binascii.crc32(payload)
219             h = abs(k) % (1<<32)
220             s = bin(abs(h))
221             crc_fresh="".join(value for index, value in enumerate(s[:26]) if index
                    % 2 == index % 4 ==1)
222             seq_num_array.insert(n,seq_num)
```

```
223             seq_num_array = seq_num_array[-n:]
224             payload_array.insert(n,payload)
225             payload_array = payload_array[-n:]
226             list = zip(seq_num_array,payload_array)
227             time_sent = datetime.datetime.today()
228             src_addr_array.insert(n,src_addr)
229             src_addr_array = src_addr_array[-n:]
230             dst_addr_array.insert(n,dst_addr)
231             dst_addr_array = dst_addr_array[-n:]
232             real_dst_addr_array.insert(n,real_dst_addr)
233             real_dst_addr_array = real_dst_addr_array[-n:]
234             crc_array.insert(n,crc_fresh)
235             crc_array = crc_array[-n:]
236             packet_type_sent_array.insert(n,packet_type[5])
237             packet_type_sent_array = packet_type_sent_array[-n:]
238             time_sent_array.insert(n,time_sent)
239             with open('/home/pi/Desktop/Wait_and_Send_A/services/image_B/
                    packets_sent_frm_AforB.csv','w') as tf:
240                             writer = csv.writer(tf,delimiter='\t');
241                             writer.writerows(zip(time_sent_array,
                                    packet_type_sent_array,src_addr_array,
                                    dst_addr_array,real_dst_addr_array,seq_num_array,
                                    payload_array,crc_array))
242             input = str(packet_type[5]+src_addr+dst_addr+real_dst_addr+seq_num+
                    payload+crc_fresh)
243             ser.write(input)
244             t.sleep(1)
245             i = i+1
246             pass
247
248         if payload == '':
249             print 'Image sending successful. Waiting for NACKs (if there is any)
                    for ', x, ' secs. \n'
250             rf.close()
251             tf.close()
252             x = x + 1
253             if x == 10:
254                 input1 = 'ER_CMD#B2'
255                 ser.write(input1)
256                 t.sleep(0.005)
257                 input2 = 'ACK'
258                 t.sleep(0.005)
259                 print 'Radio changed to B2 mode (9.6 Kbps)'
260                 pass
261             print x
262             t.sleep(0.5)
263             if x == 300:
264                 input1 = 'ER_CMD#B3'
265                 ser.write(input1)
266                 t.sleep(0.005)
267                 input2 = 'ACK'
268                 t.sleep(0.005)
269                 print 'Radio changed to default (19.2 Kbps)'
270                 execfile('/home/pi/Desktop/Wait_and_Send_A/main.py')
271             pass
272
273     except KeyboardInterrupt:
274         t.sleep(1)
275         tf.close()
276         input1 = 'ER_CMD#B3'
277         ser.write(input1)
278         t.sleep(0.005)
279         input2 = 'ACK'
```

```
280             t.sleep(0.005)
281             # Radio changed to default
282             execfile('/home/pi/Desktop/Wait_and_Send_A/main.py')
```