DENNIS LAW KIM HSENG B. ENG (HONS) ELECTRICAL & ELECTRONIC ENGINEERING SEPTEMBER 2015

COOPERATIVE NETWORK FORMATION BETWEEN SWARM ROBOTS

DENNIS LAW KIM HSENG

CHEMICAL ENGINEERING

UNIVERSITI TEKNOLOGI PETRONAS

SEPTEMBER 2015

Cooperative Network Formation between Swarm Robots

by

Dennis Law Kim Hseng

15331

Dissertation submitted in partial fulfilment of

The requirements for the

Bachelor of Engineering (Hons)

(Electrical & Electronic)

SEPTEMBER 2015

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Cooperative Network Formation between Swarm Robots

by

Dennis Law Kim Hseng

15331

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
In partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(Electrical & Electronic)

Approved by,

_____

(Dr. Ho Tatt Wei)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

September 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

DENNIS LAW KIM HSENG

ABSTRACT

Swarm robot technology could be used in future daily applications. It does not require a lot of manpower once it is deployed into the field compared to doing large scale tasks manually done by humans. However, the downside of using swarm robots is that when the number of agents grow the communication between the swarms will also gradually get more complicated. Besides that, when moving large number of agents in a swarm towards completing a specific task together will also be a challenge due to the monitoring of each swarm agent's location. Lastly, the swarms should be able to reestablish their location with the large swarms if it ever gets lost from the pack. In order to overcome these problems, engineers have been exploring various ways of making robots work together and communicate with each other with different methods and tools. For this project, it will focus more on the type of wireless communication used, ranging from short range Bluetooth to long range Wi-Fi. With the wireless communication established, the swarms should be able to perform multi-hop communication with each agents through different network topologies. Finally, the swarm robots requires a cooperative algorithm in order for it to adapt to various situation in the different environments. When each agent is deployed into outdoor environments, it will have to adapt to the surroundings while maintaining a certain predetermined flight formation and constant communication with each agent and the base station.

ACKNOWLEDGEMENT

Firstly, I would like to take this opportunity to acknowledge and express my utmost gratitude to all the individual that have helped me to be able to complete my Final Year Project. I would not be able to achieve the accomplishment of my project without the supervision and guidance of my supervisor and colleagues. Sincere appreciation and gratitude to my respected supervisor, Dr. Ho Tatt Wei for guiding me throughout the project.

I would also like to thank Abu Bakar Sayuti Bin Hj Mohd Saman for granting me the access to use the components required for my Final Year Project such as the Intel Galileo Gen 2 board to be utilize as my microprocessor. Lastly, I would like to thank some of the postgraduates who gave some advices and guidance which allowed me to overcome some major problems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Over the past few decades, the advancement in technology lead to a lot of development in various unmanned robot which would allow users to replace human beings in hazardous or difficult to reach terrains by deploying a large number of robot or agents known as swarms. Swarms refers to a large group of individuals that interacts with its neighboring individual to achieve a common goal. It needs cooperative behaviors for the swarms to maneuver through certain obstacles together in order to achieve the common goals. This reduces the numbers of manpower, human error and also increase the productivity of the task. Essentially, a swarm of robots are able to perform tasks that a normal human is able to perform but also take it to a whole new level such as search and rescue in hazardous environment, able to explore large area of space to determine the environment status, climbing unreachable heights and etc. by using swarms of drones.

This all comes down to the wireless communication between the swarm robots. For the swarm to be able to perform any maneuver or execute the cooperative behavior, they must first relay any type of information to their neighboring robots so that they know their location or tasks at hand. The wireless communication is the foundation for any swarm to perform any task. Without it, most of the robots will be moving independently or blindly in an unknown environment. This will increase the chance of multiple robots to get lost or move into unwanted area. By implementing a multihop wireless network for this project, the robots would be able to relay any information or location to the base station or remote swarm robots far away.

1.2 Problem Statement

The agents of a swarm system is generally aware of the location of their own and neighboring agent's position or they possess the ability to relay information to a different agents. However in real-world application where the uncertainty of the environment is an issue, this type resource might not be achievable for the swarm agents [1]. Global positioning system (GPS) would be rendered inadequate in an environment with thick foliage or environment with metallic barriers that would interfere with the line of sight from the satellites which is required for the agents to locate their position.

Numerous circumstances requires the swarms to establish a monitoring stations that receives the information from the swarms and updates the station on their current position. This however restricts the area of movements of the swarms to a certain size and also increases the difficulty in establishing communication if the area of the swarms increases causing the communication nodes to be too far apart [2].

In a real-world situation, swarm of robots may geographically spread themselves across terrain with barriers to effective communication. The swarm behavior may suffer if communication between agents to be disrupted. In this project, a reliable multihop communication link should be investigated and established between a quadcopter and the ground swarms that are spread out over a terrain.

Other than that, the quadcopter swarm should also be able to maneuver through obstacle autonomously and also maintaining a certain flight formation that was predefined to the swarms itself. An inter-relationship physical structure that keeps the swarms in a predefined patters [3]. This pattern will also allow the swarms to keep them apart far enough from each other to optimize the whole area but not so far that the connection between them would break.

1.3 Objective and Scope of Study

The main **objectives** of this project is to investigate the cooperative behavior between swarms and also the flight formation optimization algorithm for the swarms to navigate themselves in different environments.

In order to achieve the objectives, the following will be the main focus for this project:

1. Develop and evaluate algorithms for a swarm of agents to position themselves to optimize formation of multihop communication network
2. Simulate the algorithms with increasing complexity due to environmental and reliability factors

The **scope of study** will be predicated around the means communication between the swarms of drones. The main focus will be on the wireless network communication of the swarms and simulate which of the wireless device that will optimize the swarm's communication.

CHAPTER 2
LITERATURE REVIEW

2.1 Wireless Protocols

For a short range wireless communication, there are 4 IEEE 802 protocol standards for the specification of the Physical PHY and Media Access Control MAC layers of the protocol that most wireless devices implement which is IEEE 802.15.1 commonly used for low-rate data transfer like Bluetooth to connect simple devices such as wireless mouse or headset, IEEE 802.15.3 for high-rate data transfer also known as Ultra-WideBand (UWB) which is used for multimedia links with high-bandwidth, IEEE 802.15.4 for low-rate data transfer use by ZigBee and usually used for wireless control networking and monitoring [4].

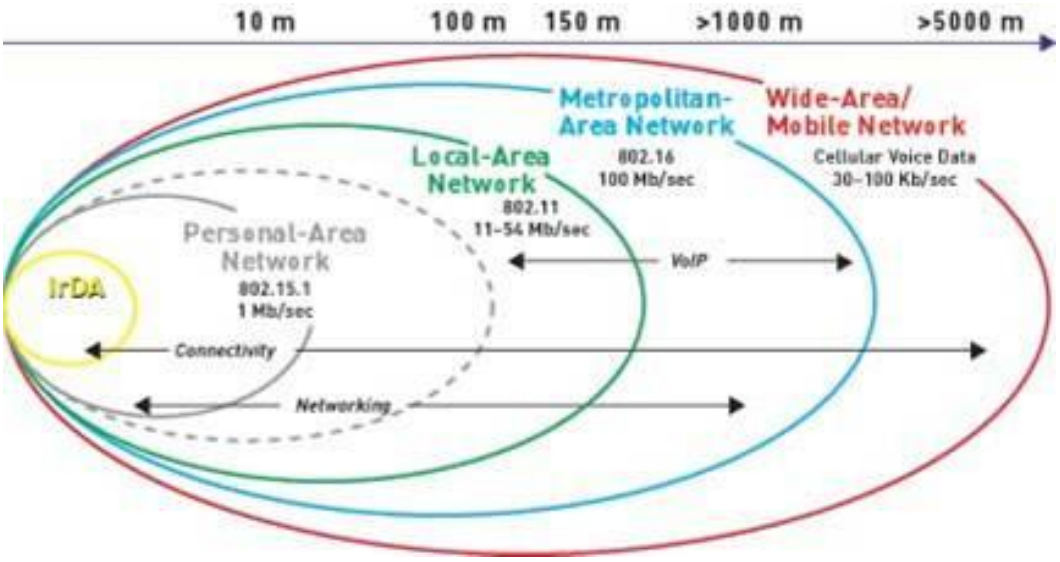


Figure 2.1 Range of Wireless Area Network (WAN)

2.1.1 IEEE 802.15.1

Bluetooth wireless devices generally use the protocol standard IEEE 802.15.1 [5]. It operates using short range Ultra High Frequency (UHF) of 2.4 GHz to 2.5 GHz in the Industrial, Scientific and Medical (ISM) radio band with a bandwidth of 100 MHz. Essentially it is a small, low powered radio device built onto a chip which uses the unlicensed 2.45 GHz radio band worldwide [6].

Bluetooth enables mobile objects to connect with other devices within short range without direct Line-Of-Sight. Conversely, the range for Bluetooth devices is very short ranging from 10m–100m. Hence, it is mostly implemented in Wireless Personal Area Network (WPAN) such as wireless mouse and headset [7]. However, the efficiency of the Bluetooth will drop when operating around Wi-Fi because Bluetooth signals operates at the same range of frequency as certain Wi-Fi standard such as IEEE 802.11b and IEEE 802.11g.

2.1.2 IEEE 802.15.3

For a high-rate data transfer over a short range, it is achievable using IEEE standard of 802.15.3 which is generally used by UWB. It provides the necessary quantity of service (QoS) for a real-time distribution of multimedia content such as audios and videos wirelessly over the spectrum of 3.1 GHz – 10.7 GHz radio band. Mostly suitable for home multimedia devices for audio and video transfer [8]. The advantages of UWB is that it has a wide bandwidth ranging from 100 MHz – 480 MHz for large data transfer which means it can replace the conventional high speed serial bus such as USB 2.0 and fire wire standards [9]. By overall comparison with other short to medium range wireless technology as IEEE 802.15.1 (Bluetooth) and IEEE 802.11 (Wi-Fi) it has a higher data rate transfer, lower cost and lower power consumption [10].

2.1.3 IEEE 802.15.4

IEEE 802.15.4 is a standard protocol for Low Rate Wireless Personal Area Network (LR-WPAN) for low powered devices such as ZigBee. This device was mainly developed for low-rates wireless monitoring and control [11]. It uses the same frequency band as the Bluetooth and UWB which is 2.4 GHz on the ISM radio band. The most distinctive advantages of the ZigBee module is that it only requires a very low power consumption during transmission and this translate into lower battery consumption. IEEE 802.15.4 standard protocol is usually more suitable for large number of heterogeneous sensor devices known as Wireless Sensor Network (WSN) spread out over a large area of field [12].

The ZigBee was also designed to support a standard WSN protocols which specifies the network layer for several network topologies shown in Figure 2. The network establish communication between devices through a central controller known as PAN coordinator [13]. The PAN controller will set up a network with the Full Function Devices (FFD) also known as router to communicate with the Reduced Function Devices (RFD) shown in Figure 3 [14].

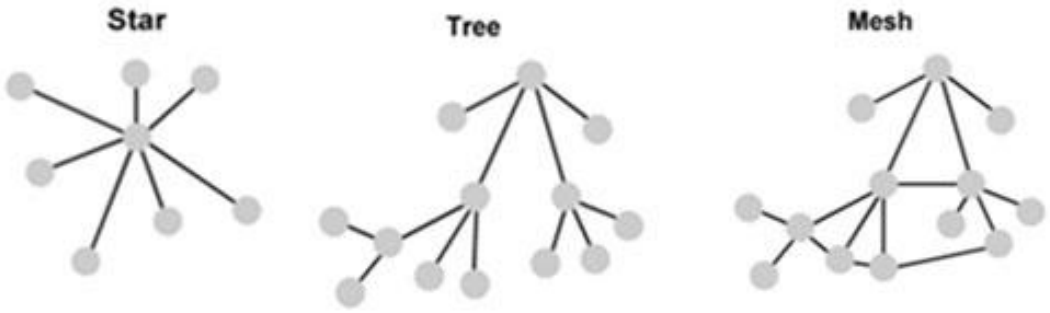


Figure 2.2 Star, Cluster-Tree and Mesh network structure

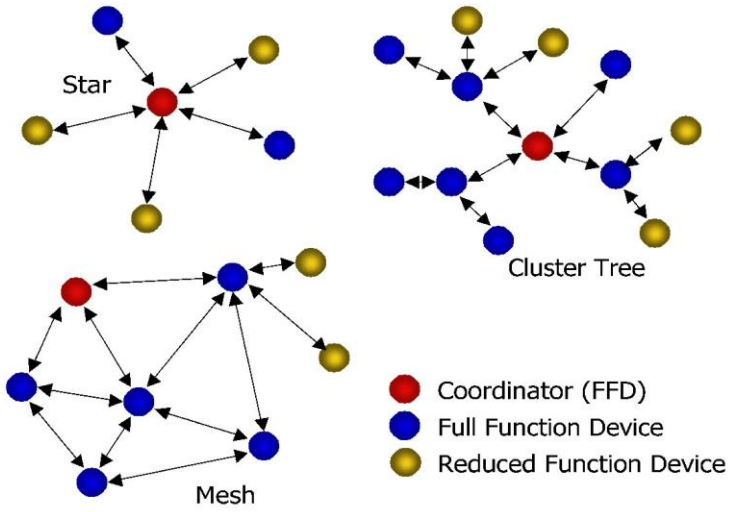


Figure 2.3 Network Topology

2.2 Multihop

Multihop networking has been part of our modern society for centuries. Most recognizably used by our cellular phone. The concept of multihoping network is to convey a specific data to a specified destination through multiple stations known as nodes before arriving. This networking system for a cellular phone is mostly static node which has a certain drawbacks which is the path loss and noise. The problem can be solved by using several fixed relay stations as the nodes. [15]

However, for this project, the nodes are dynamic and this requires Mobile Ad-hoc Networks System because the nodes are constantly moving or changing based on the environment. This requires optimization of the multihop wireless network such as, the distance between nodes, type of network topologies and the environments.[16] Each aspects will pose a certain drawbacks to the systems. Take the distance between the nodes for example. If the nodes are too far apart, the chances of not receiving data is higher.[17]

As for the topologies, each type of topologies gives different outcomes. As mention before, Mesh network topologies are more reliable but also more complex to implement compared to the Tree or Star network topologies. Mesh networks are able to establish multihop more easily if one of the nodes within the topology breaks down. This allows it to “self-heal” if there is any faulty nodes. [18]

2.3 Network formation

When working with a large number of robot like a swarm, there will be certain drawbacks such as the communication between individual robots in the swarm. For example, if one of the robots move too far away and lose its connection, the wireless network has to be reestablished. One way of doing it is to deploy quadcopters with Mobile Ad-HOC Network which consist of multiple mobile nodes such as the quadcopters. A mobile Ad-HOC Network is a self-organizing nodes that can communicate with each other using each other's transmission to perform a multihop communication. [19]

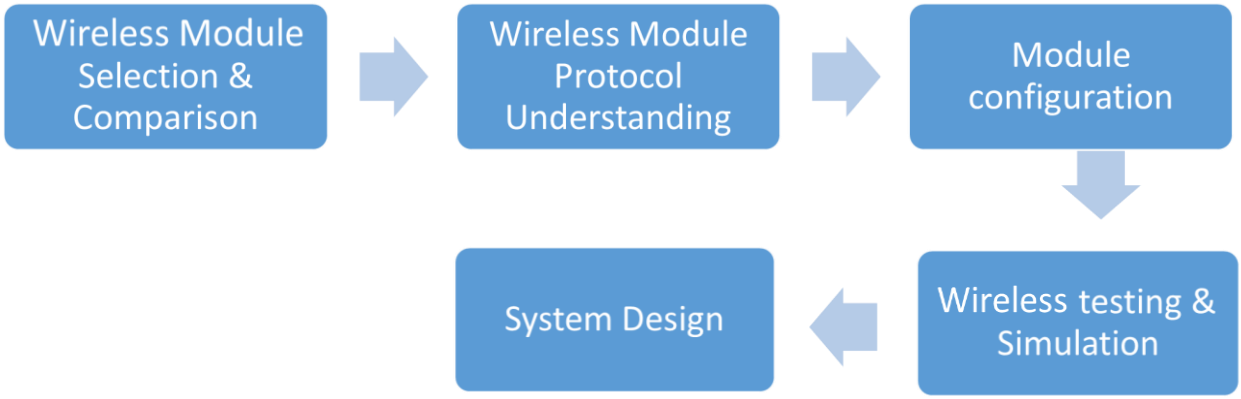
The network of this Mobile network requires a leader or a base station for the whole swarm. This leader will be required to have global information of the swarms such as the task and location. Using this information, the leader will be the one telling the swarms how to move throughout the environment. [20]

Another importance of a Mobile Ad-HOC Network is also the routing protocol of the data packet. Since the swarms will be constantly moving, the nodes have to have a certain knowledge of who and where to send the data packet. The network requires the information such as the location which is known as the unique address of each individual swarm robots. Using these addresses, the base station can choose where to send the data packets to. [21]

CHAPTER 3

METHODOLOGY

3.1 Flow Chart



3.1.1 Wireless Module Comparison & Protocol understanding

	Bluetooth	Ultra-Wideband (UWB)	ZigBee
Physical layer Protocol	IEEE 802.15.1	IEEE 802.15.3	IEEE 802.15.4
Bandwidth	720 Kbit/s	500 Mbit/s	20-250 Kbit/s
Range (Meter)	10-100	10-20	10-100
Network structure Architecture	Star	Mesh	Mesh
Advantages	Low cost, convenience	High Speed, large data transfer	Reliability, low cost, low power, scalability
Application	Cable Replacement	Video, Audio transfer	Monitoring, controlling

Table 3.1 Comparison between Bluetooth, UWB, ZigBee

From the Table 3.1, comparison between various wireless modules, it shows that, the Zigbee module is the most suitable in terms of range, network structure architecture and reliability. It is able to perform a mesh network topology, low power consumption, and enough range to cover a wide area.

It is also important to focus on the low powered feature that separate Zigbee protocols from other wireless protocols. The low data rates, low frame overhead, and power management is built directly into the protocol of the Zigbee modules. Since the wireless network will be mounted onto a quadcopter which is known to have low battery capacity, the low power consumption per individual node on the network is desirable. With the Zigbee module, the wireless network will be able to set up considerably longer in order to search and connect to other swarm robots.

3.1.2 Module Configuration

Using monitoring software such as "XCTU", users could control their XBee devices. **XCTU** is a free multi-platform application designed to enable developers to interact with Digi RF modules through a simple-to-use graphical interface.

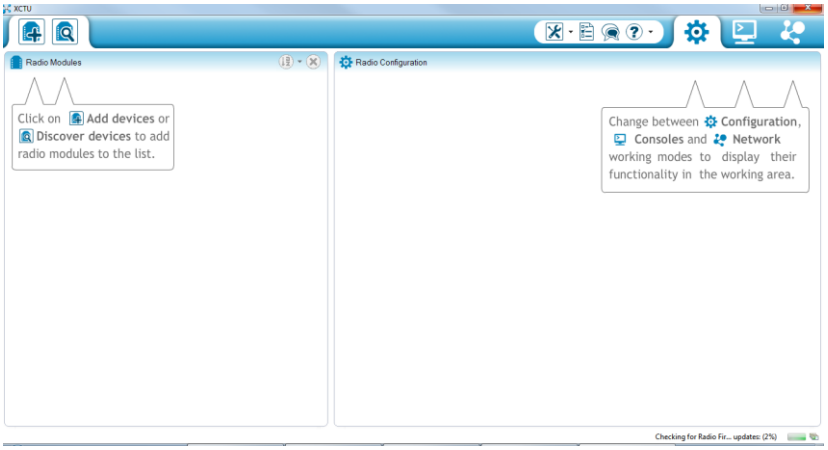


Figure 3.1 XCTU interface

XCTU allows users to control several XBee modules at a time. With this, the configuration and protocols can be written onto the modules based on the operations you require. There are a few settings that is required for every XBee modules to communicate with each other. Which is, the Personal Area Network ID (PAN ID), XBee modules unique source address (MY Address), and the destination address (DH & DL).

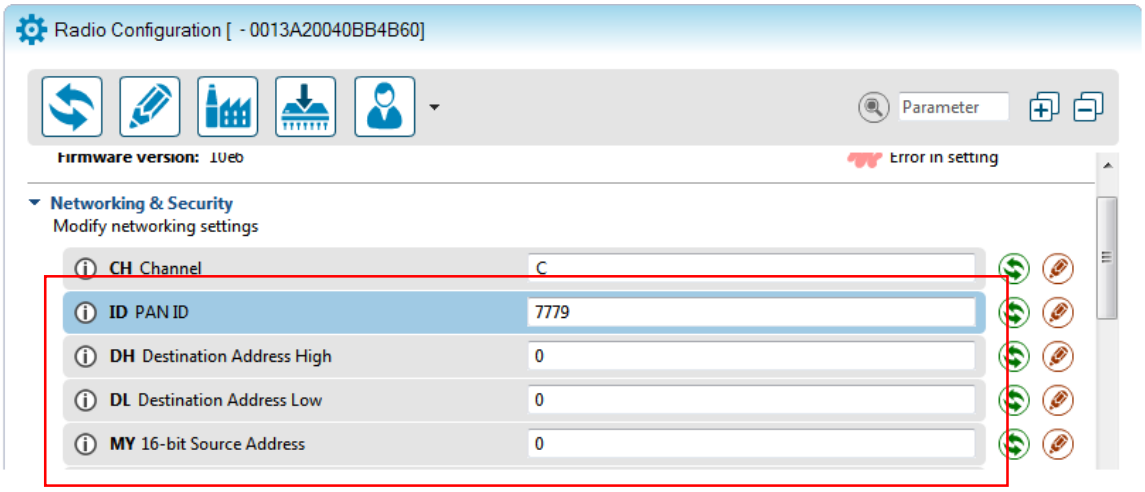


Figure 3.2 PAN ID, MY Address and destination address configuration

The PAN ID is used to configure a personal network for which only modules with the same ID could communicate. This is to prevent any interference from any other XBee modules in the vicinity. It uses a 16-bit HEX network ID ranging from 0000 to FFFF allowing users to have their own personal network lowering the chances of colliding with any other XBee. The MY Address of the XBee is to give each modules a unique ID for receiving data. It also uses a 16-bit HEX address ranging from 0000 to FFFF giving it 65535 unique address. If any 2 modules possess the same address ID, they will both receive the same data broadcasted to that address ID. Lastly the destination address DH &DL is the address that defines the destination of the XBee and which XBee is it talking to.

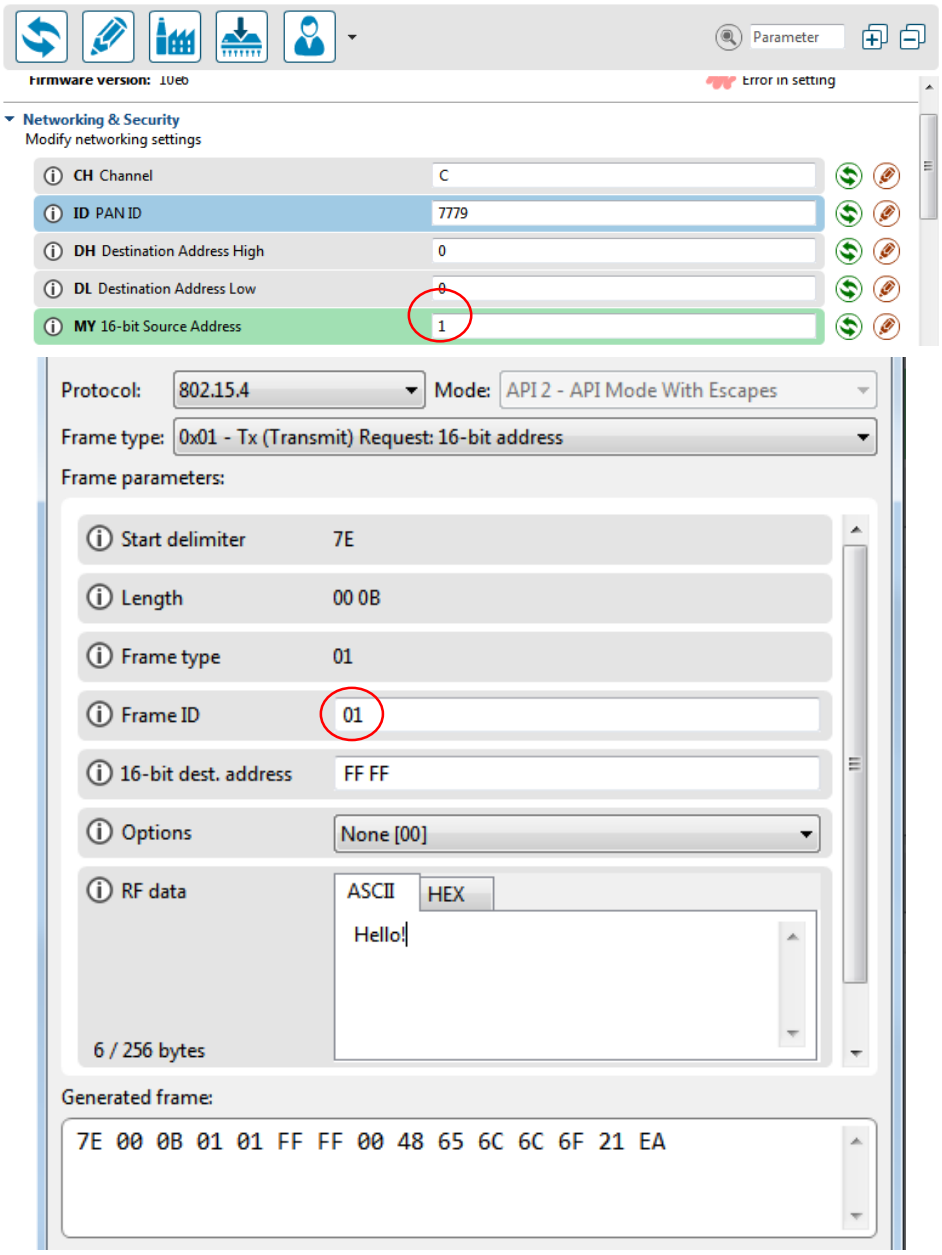


Figure 3.3 Packet with a specific ID address

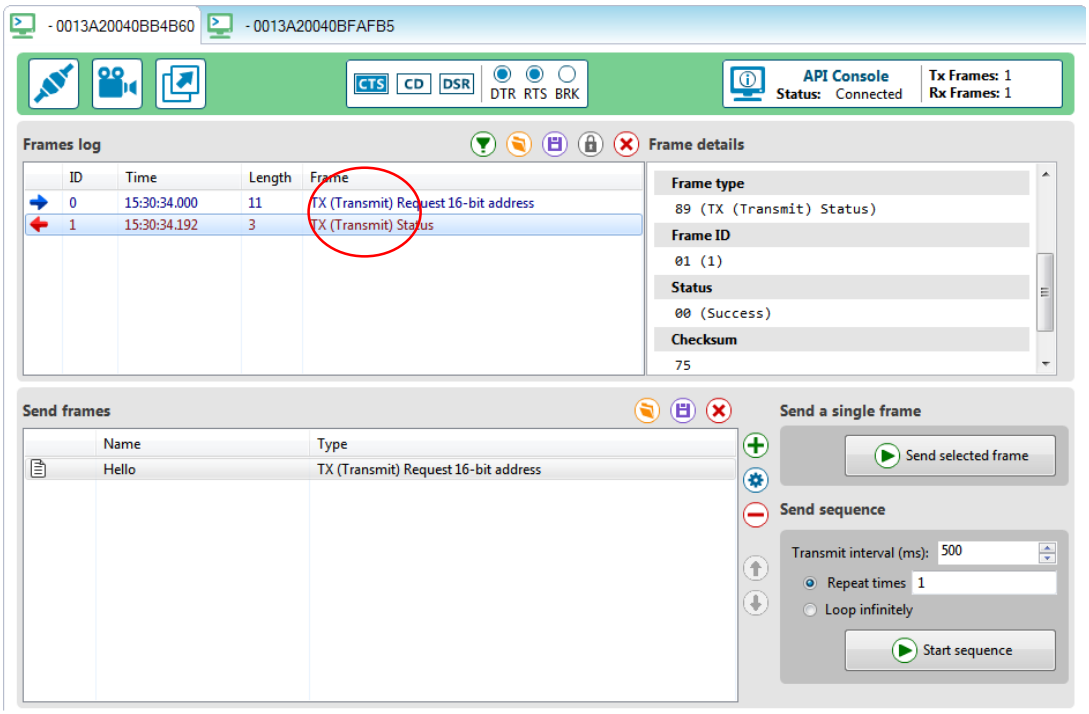


Figure 3.4 Transmitting data packet to a specific address ID

By giving one of the XBee modules a unique address ID of 01 and a packet of data with the specific destination to the address ID 01, only the modules with the address ID will receive it as shown in the diagrams.

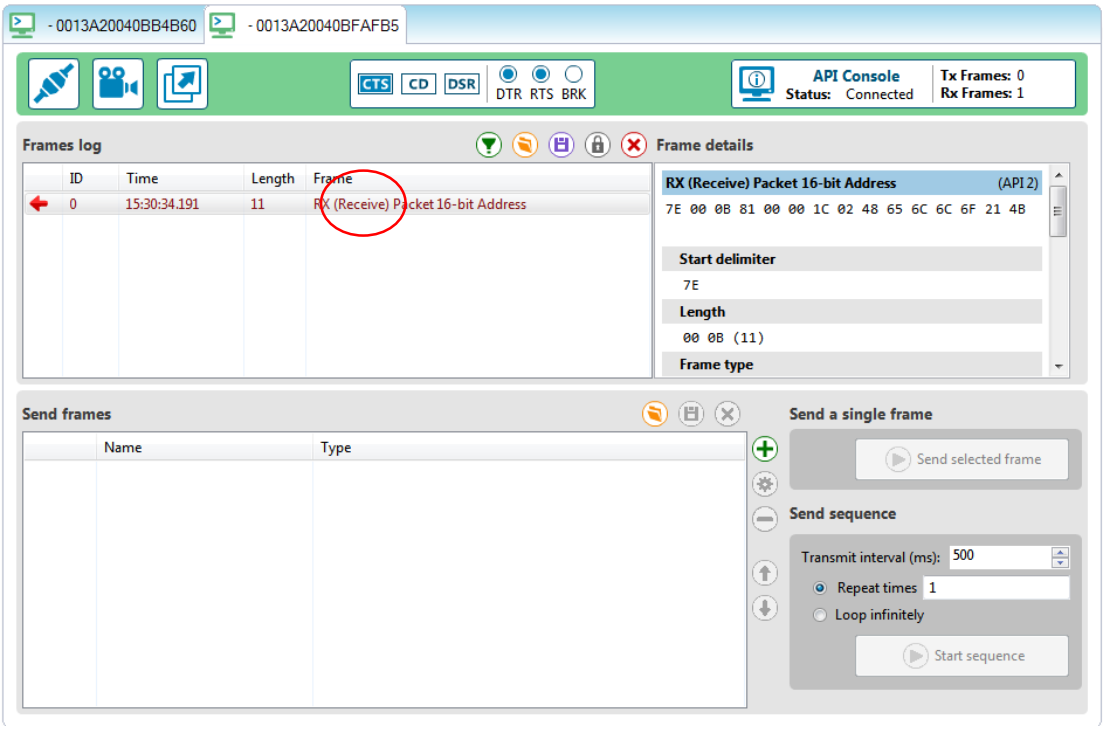


Figure 3.5 Receiving data packet

Other than that, the strength of the XBee modules within the range of the Coordinator or the base station can also be observed using the XCTU in terms of dBm (Decibel-milliwatts). It's the power ratio in decibels of the measured power in reference to one milliwatt. The power of the received signals can also be calculated using a formula which is : $x = 10 \log_{10} \frac{P}{1mW}$. P is the power in mW as x is in dBm.

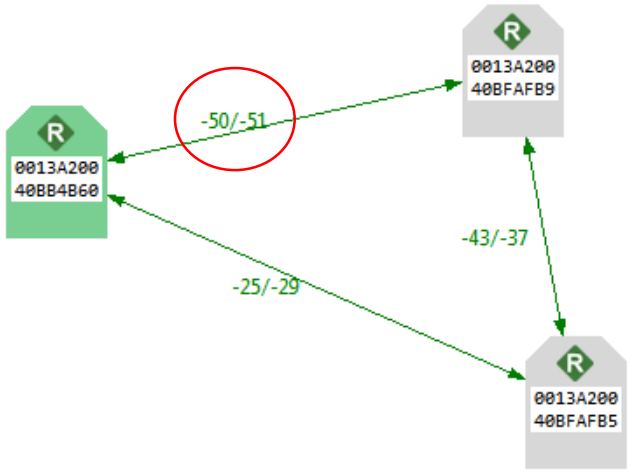


Figure 3.6 XBee modules signal strength

From the diagram, the strength of each XBee modules are shown. The strength of each modules are shown in different colors based on the power the XBee is receiving.

DigiMesh network

Modify the minimum values of the quality ranges and their colors:

Quality	Maximum	Minimum	Units	Color
Very strong	0	-70	dBm	(0,125,0)
Strong	-70	-80	dBm	(12,150,159)
Moderate	-80	-90	dBm	(212,105,0)
Weak	-90	-100	dBm	(231,0,0)

ZigBee network

Modify the minimum values of the quality ranges and their colors:

Quality	Maximum	Minimum	Units	Color
Very strong	256	195	LQI	(0,125,0)
Strong	195	130	LQI	(12,150,159)
Moderate	130	65	LQI	(212,105,0)
Weak	65	0	LQI	(231,0,0)

Figure 3.7 Strength classification and configuration

3.1.3 Wireless Protocol Testing & Simulation

The basis for swarm wireless network communication is based on the wireless protocol standard implemented onto the quadcopters. Using the wireless IEEE 802 standard protocols, a wireless network communication can be established within the swarms. This protocols allow individual swarm agents to set up their own network to communicate with the Base Station.

Using network simulation tools, such as "XCTU", Arduino IDE and X-ming monitor, it allows users to monitor and configure the wireless module to be able to perform multihop network.

3.1.4 System Design

Using the wireless multihop network, an adaptive network formation could be implemented. As shown in the Figure below, the network should be able connect to the lost swarm back to the base station using quadcopters using multihop wireless network.

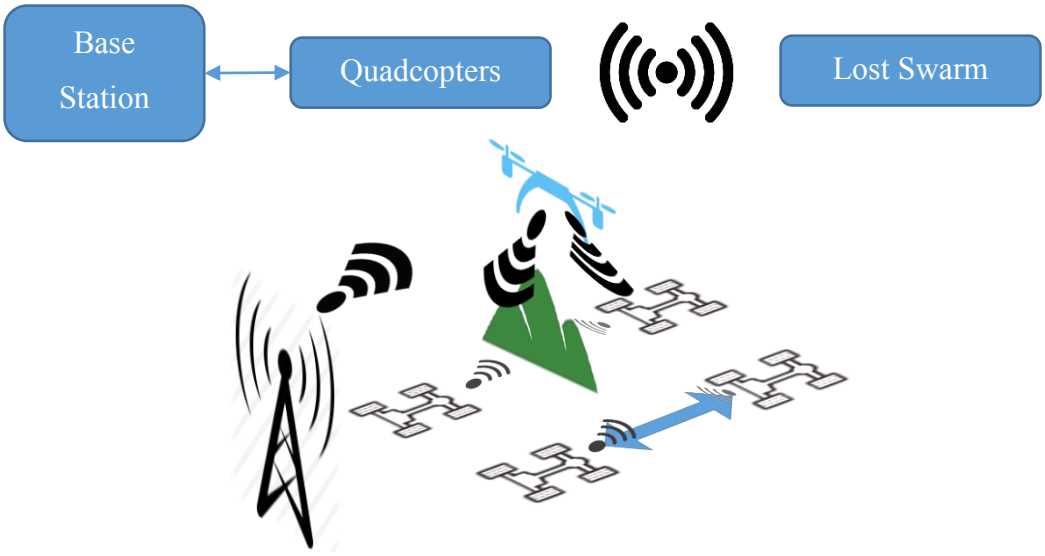


Figure 3.8 Multihop network between base station and swarm

3.2 Coverage Area of Wireless Network

In real life application of swarm robots, the main concern for the swarm is the wireless communication between each individual swarms and neighboring agents. Due to some natural environments such as barriers, trees or hills that would interfere with the Line-Of-Sight of some sensors that are required for wireless communication. Hence the coverage area of the each wireless module might be different depending on the location of each of the individual agents of the swarms.

To determine the area of the coverage over a specified area, the wireless module is tested in various environment conditions such as in-doors, outdoors and woods. The agents should also be tested with various transmission power to determine how much power should be needed to overcome the environment or barriers. The lost in the signal might be caused by path loss and shadowing. In an ideal case, path loss I cause by the inverse square law which means the power density is proportional to the inverse square of the distance. But in real world application it is caused by obstacles and transmission medium.

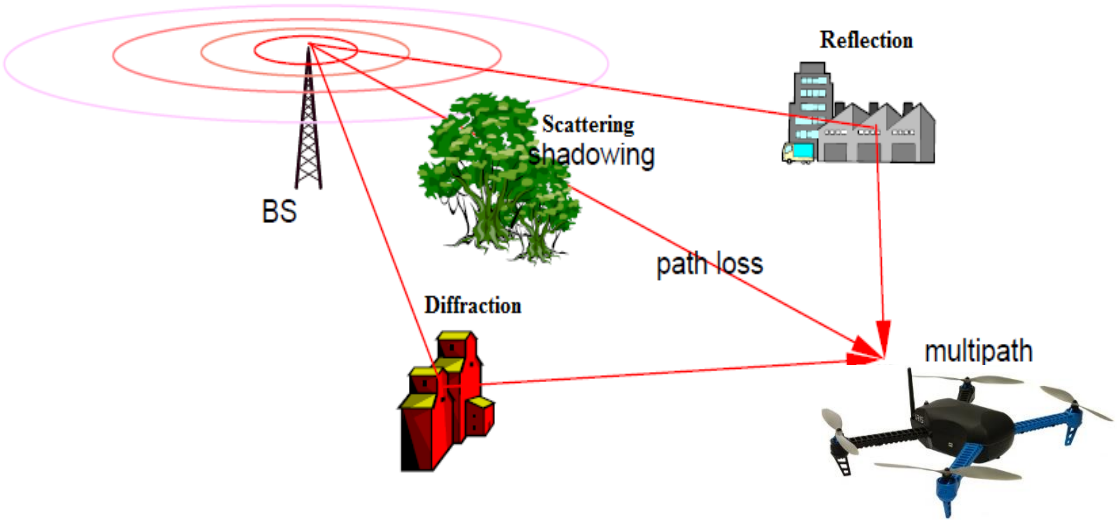


Figure 3.9 Network signal propagation through outdoor environment

3.3 Key Milestone

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14
XBee module & configuration understanding														
Arduino XBee Serial communication														
Transmitter and Receiver implementation - Packet sending - Packet Receiving - Acknowledgement														
Prototype testing and simulation														
Multihop communication testing and simulation														
Flight formation algorithm														

Table 3.2 Key Milesstone chart

3.4 Gantt Chart

Task		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
FYP 1	Literature review																												
	Extended proposal																												
	Project review																												
	Proposal defense																												
	Network protocol																												
	Prototype analysis																												
	Interim report																												
FYP 2	Network structure																												
	Swarm formation																												
	Progress Report																												
	Pre-Sedex																												
	Dissertation assessment																												
	Technical report																												
	Viva																												
	Final project report																												

Table 3.3 Gantt Chart

CHAPTER 4
RESULTS AND DISCUSSIONS

4.1 Wireless Network Design

To implement an adaptive wireless network, the system requires wireless multihop network to relay information to individual swarm robots. To do this, a network protocol has to be implemented such as how to detect the boundaries of transmission and how to grow a multihop network. One way of doing that is to use LED's to illustrate the connection between nodes.

XBEE MODULE	LED 1 (GREEN)	LED 2 (YELLOW)	LED 3 (RED)
Connected to base station	ON	OFF	OFF
Connected to remote Xbee	OFF	ON	OFF
Not connected to anything	OFF	OFF	ON

Table 4.1 XBee module connection using LED

When the XBee is connected to the base station, the LED will be green. However, when the XBee is not connected to the base station due to loss of connection or distance, the LED will be yellow and lastly, when the XBee module is neither connected to the base station or any remote XBee, the LED will be red.

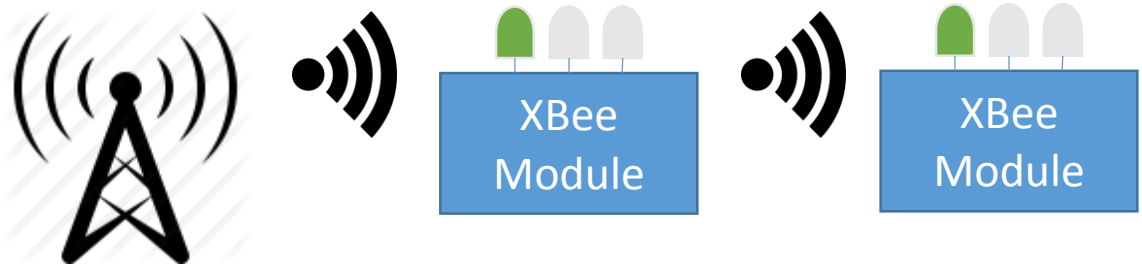


Figure 4.1 LED when XBee module is connected to base station

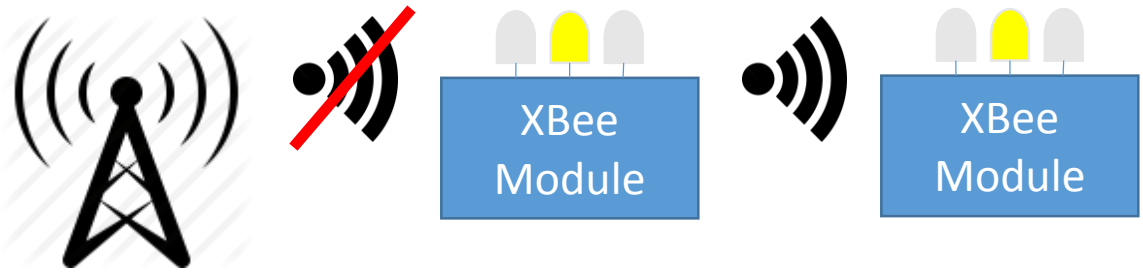


Figure 4.2 LED when XBee module is connected to remote XBee module

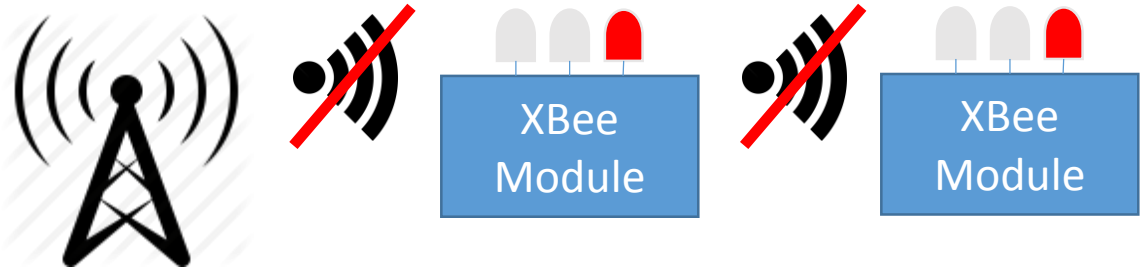


Figure 4.3 LED when XBee module is not connected to anything

4.2 Arduino (Intel Galileo Gen 2) – XBee interfacing

The XBee module is a versatile wireless network module. It is able to interface will multiple microcontroller such as Arduino, Raspberry Pi, Waspmote and etc. For this specific project, an Arduino microcontroller will be used. The XBee module will be interfacing with an Arduino microcontroller by Intel which is the Intel Galileo Gen 2 board as seen in the figure below.

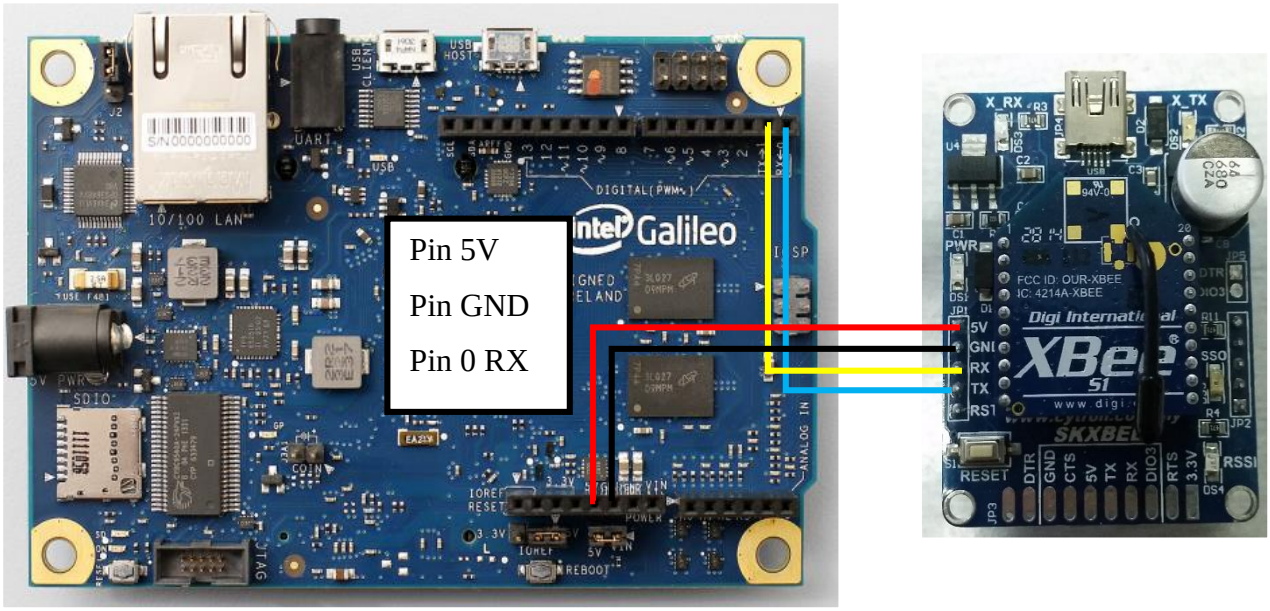


Figure 4.4 XBee Galileo connection

However, the XBee requires an adapter or a starter kit for the modules to interface. As seen in the figure, the starter kit used is a SKXBEE. The Galileo has 20 pins but only 4 pins is required for the interfacing which is just the Supply voltage (Pin 5V), Ground (Pin GND), Receiver (Pin 0-RX) and Transmitter (Pin 1-RX). The other pin could be used for other future application. The module can be easily configured using XCTU and Arduino IDE through serial interfacing. To configure the XBee module serially, a simple coding is required.

4.1.1 Serial Communication between XBee & Galileo Coding

```
TTYUARTClass* gSerialStdPtr = &Serial; // Galileo, /dev/ttyGS0, Tx pin
TTYUARTClass* gSerialTwoPtr = &Serial1;// Galileo, /dev/ttyS0, Rx pin
bool gGalileo = true;
String a = "Hello World";

void setup()
{
      int i;
      gSerialStdPtr->begin(9600); // Sender IDE
      gSerialTwoPtr->begin(9600); // Receiver
      waitForUser(5); // Give usr time to open serial terminal
      gSerialStdPtr->println("XBee-Sender-setup");
}
```

Figure 4.5 Coding for serial communication

For the XBee to communicate with the Galileo, first a serial communication must be set up using the Arduino IDE as seen in the figure above. First of all, the XBee and the microcontroller must be set with the same baud rate using **begin()** for both Transmitter (Tx) and Receiver (Rx). This is important because it allows the microcontroller and XBee to send and read at the same speed. If there is a mismatch, the two systems will move at a different speed causing the microcontroller to process the data wrongly. The baud rate from Arduino IDE ranges from 9600, 14400, 19200, 28800, 38400, 57600 or 115200 depending on the application of the microcontroller.

Other than that, the **gSerialStdPtr(Serial);** is to initiate Pin0 on the Galileo to TX of the XBee module and the **gSerialTwoPtr(Serial);** is to initiate Pin1 on the Galileo to the RX of the XBee module. This now allows the XBee to print out what the module has received from the remote XBee on to the serial monitor of the Arduino IDE or transmit the data coming from the microcontroller to the remote XBee. To monitor this, the Arduino IDE serial monitor is used.

4.1.2 Sending

```
TTYUARTClass* gSerialStdPtr = &Serial; // Galileo, /dev/ttyGS0, Tx pin
TTYUARTClass* gSerialTwoPtr = &Serial1;// Galileo, /dev/ttyS0, Rx pin
bool gGalileo = true;
String a = "Hello World";

void setup()
{
      int i;
      gSerialStdPtr->begin(9600); // Sender IDE
      gSerialTwoPtr->begin(9600); // Receiver
      waitForUser(5); // Give usr time to open serial terminal
      gSerialStdPtr->println("XBee-Sender-setup");
}

void loop()
{     // Send data in 1 sec increments
      gSerialTwoPtr->println(a);
      if(gGalileo) gSerialStdPtr->println("Hi World");
      delay(1000*1);
}
```

Figure 4.6 Coding for transmitter

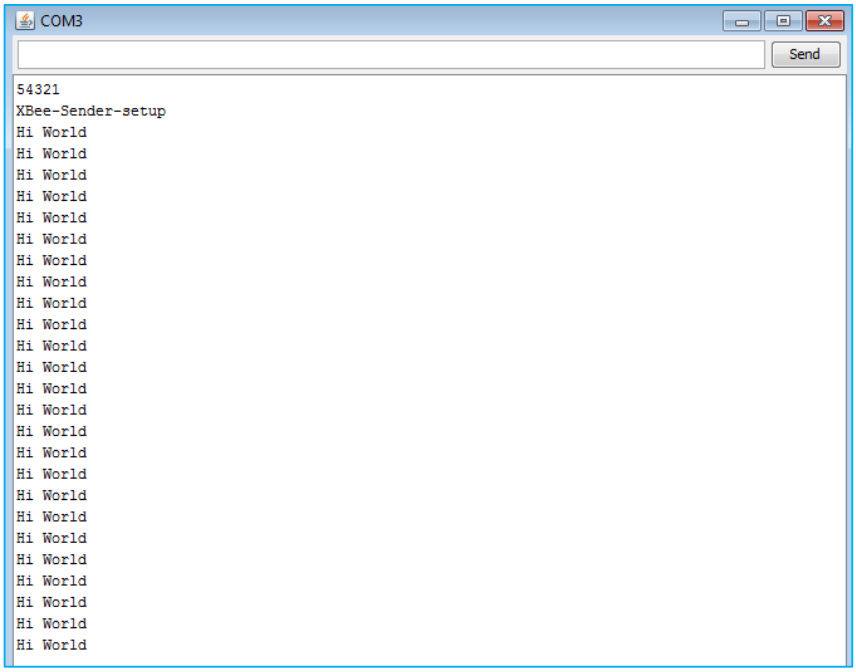


Figure 4.7 Results for sending on serial monitor of the Arduino IDE

Once the XBee is able to communicate with the Arduino serially, it is now able to transmit or receive data based on the given program. For example seen from the figure, this program is to send a simple data packet to the remote XBee. In this case, this example is sending a string of data which is “Hi World”. It will constantly broadcast throughout the same PAN ID set that was configured previously for the mesh network. Any XBee module with the same ID will be receiving this data packet.

To allow the module to receive specific data packet, a unique Address ID is required. The packet could include the address of the module and any module that receive the packet will just keep on broadcasting the data packet until it reaches the module with the ID of the packet. Once this is achieve, an acknowledgement will be sent back to the base station and the broadcasting will stop

4.1.3 Receiving

```
void loop() {

  // put your main code here, to run repeatedly:
 // Give indication that no data has been received
 if(qData == false) gSerialStdPtr->println("XBee-Receiver-waiting");
 //gSerialStdPtr->println(gSerialTwoPtr->available());
 // Get data from Sender and print to Receiver serial port

 while(gSerialTwoPtr->available())
 {
 c= gSerialTwoPtr->read(); // Read XBee data
 gSerialStdPtr->write(c); // Write local
 qData = true;
 }
 gSerialStdPtr ->println(c);
 delay(1000);
 if(qData == false) delay(1000*1); // Slow down until data is rec
}
```

Figure 4.8 Coding for receiver

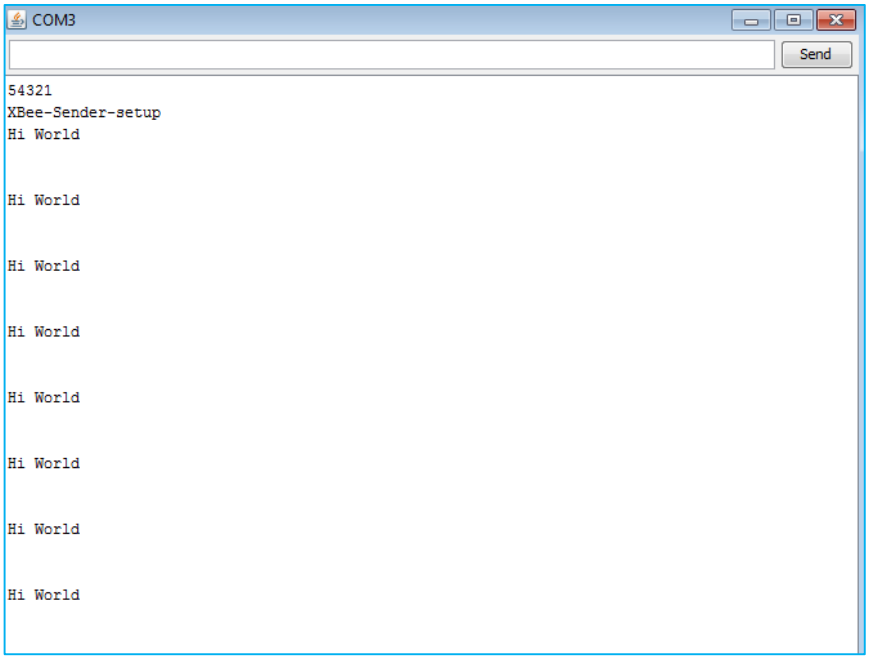


Figure 4.9 Results for receiving

The figure above is the coding for the receiver, the initialization of the pins and configuration of the XBee is the same as the transmitter. Once the receiver XBee detects the data packet sent by the XBee with the same PAN ID, the receiver will accept the data packet and print it onto the serial monitor as shown in the figure.

To detect whether if the remote XBee is on stand-by or receiving data, two LED's are added onto the circuit as shown in **Figure 4.15.**

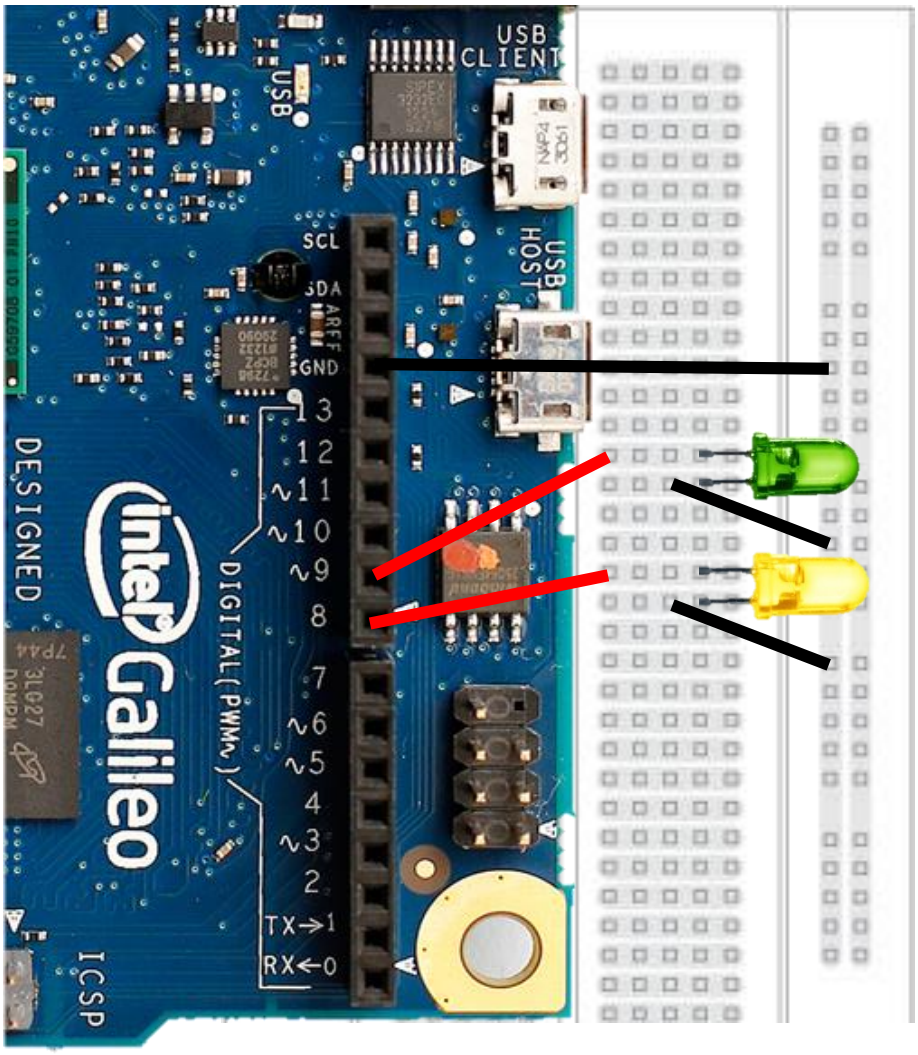


Figure 4.10 LED connection

The yellow LED is to signify that the XBee module is waiting. It means that the XBee is either too far away or the base station is not sending any data. Once the remote XBee is close enough to connect to the base station and it starts receiving data, the green LED will light up.

```
void flashLed(int pin, int times, int wait) {

    for (int i = 0; i < times; i++) {
      digitalWrite(pin, HIGH);
      delay(wait);
      digitalWrite(pin, LOW);

      if (i + 1 < times) {
        delay(wait);
      }
    }
}
void loop()
  {
      // put your main code here, to run repeatedly:
      // Give indication that no data has been received

      if(qData == false) gSerialStdPtr->println("XBee-Receiver-waiting");
        flashLed(notconnectLED, 1, 100);

      //gSerialStdPtr->println(gSerialTwoPtr->available());
      // Get data from Sender and print to Receiver serial port

      while(gSerialTwoPtr->available())
        {
            flashLed(connectLED, 1, 1);
```

Figure 4.11 LED coding

Figure above shows the coding for the LED to light up when the Xbee is waiting or receiving data.

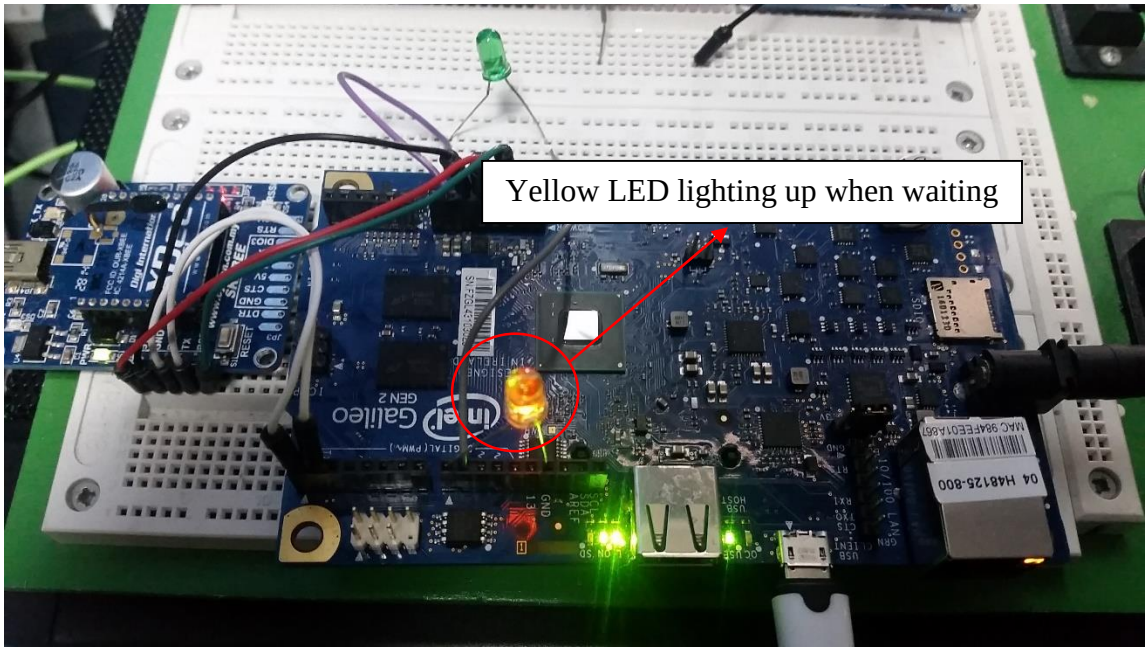


Figure 4.12 LED when XBee is waiting

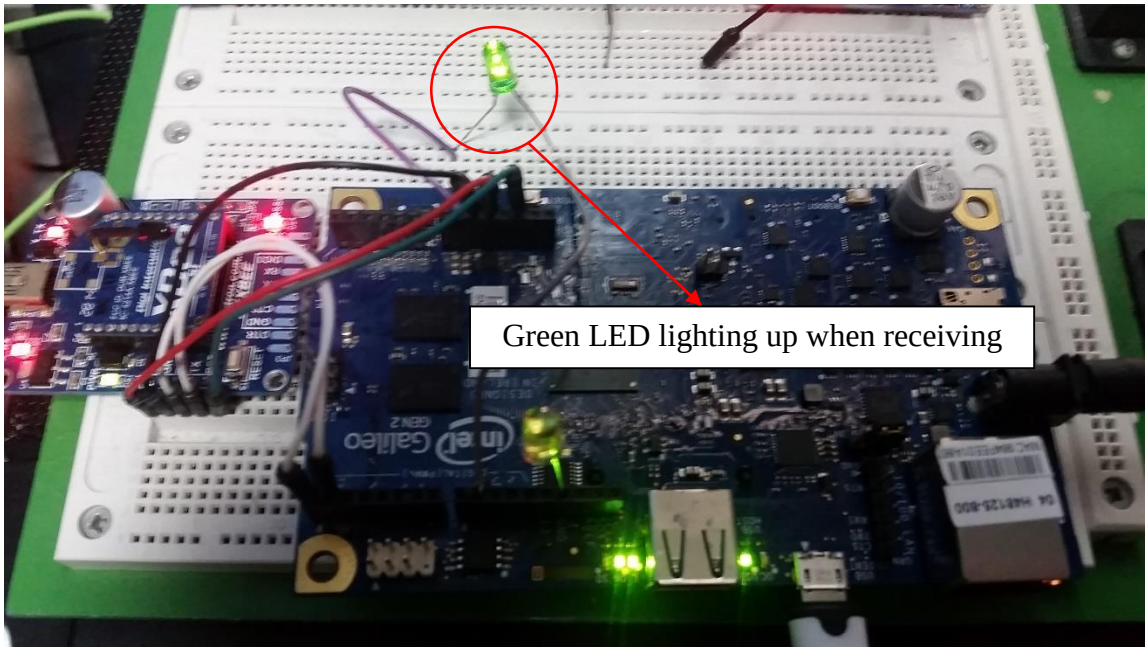


Figure 4.13 LED when XBee is receiving

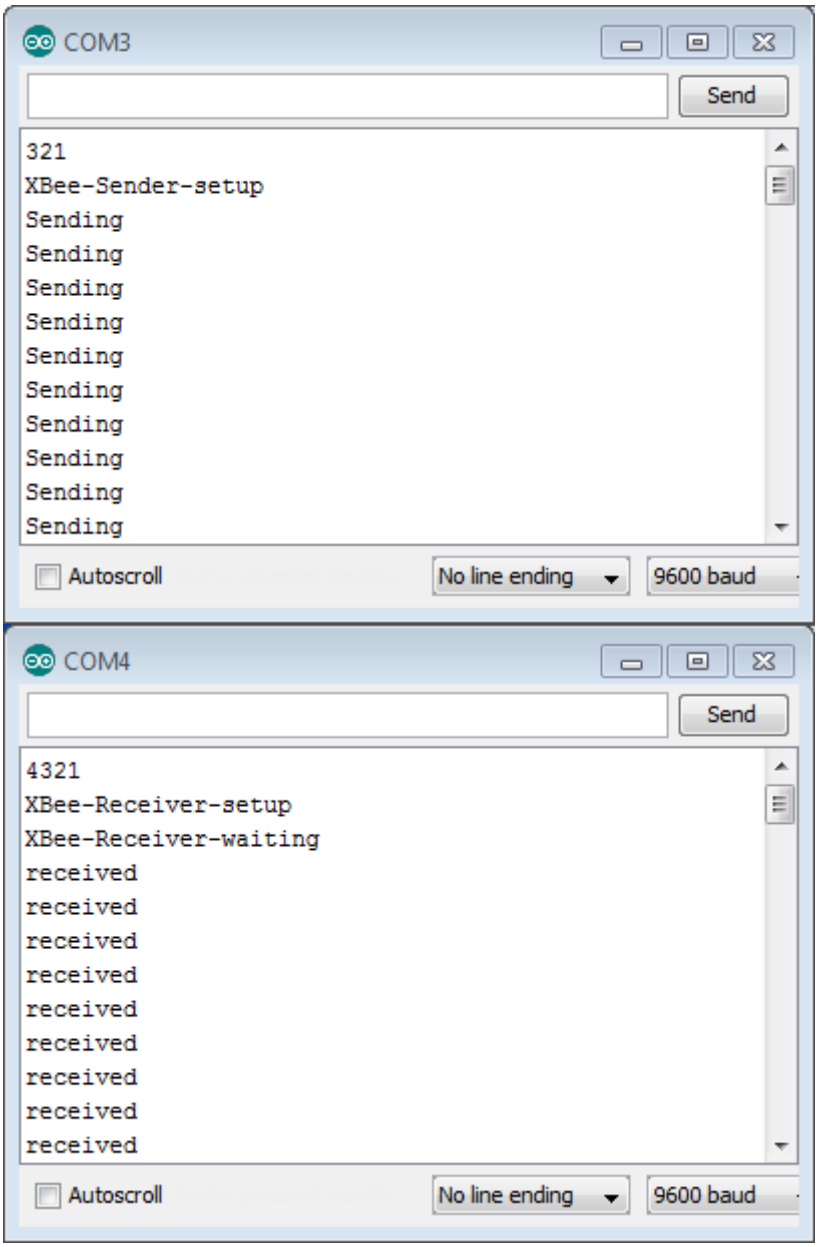


Figure 4.14 Serial monitor showing Base station and remote XBee

4.1.4 Simple Architecture of XBee Transmitter and Receiver

Based on the codes

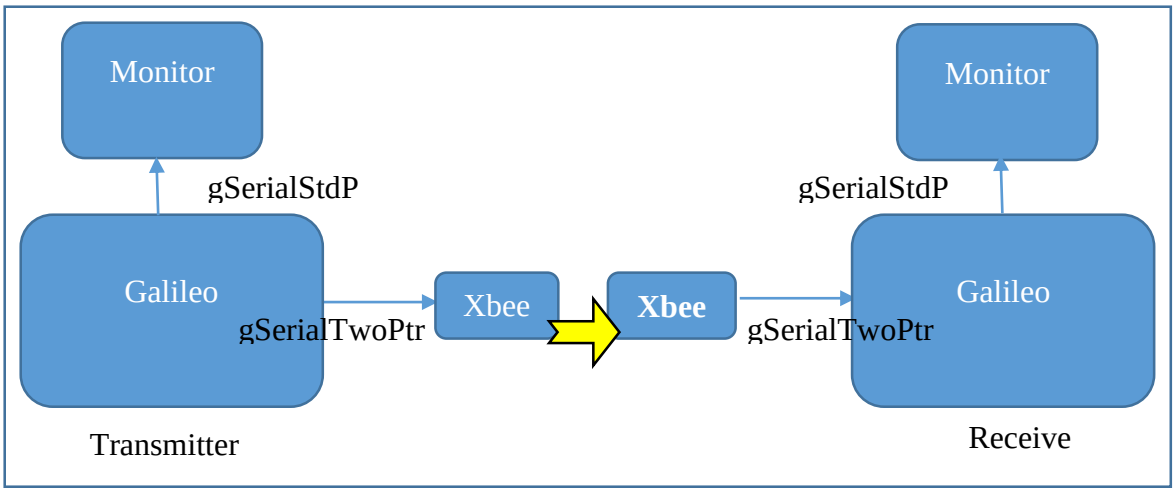


Figure 4.15 Transceiver architecture

4.1.5 Algorithm for Distance Control

Base Station and Single Node

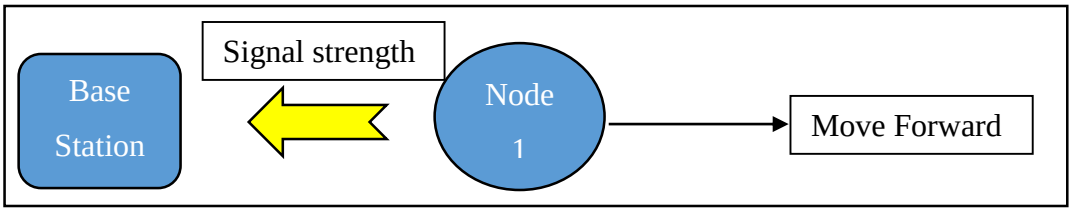


Figure 4.16 1 Node Communication

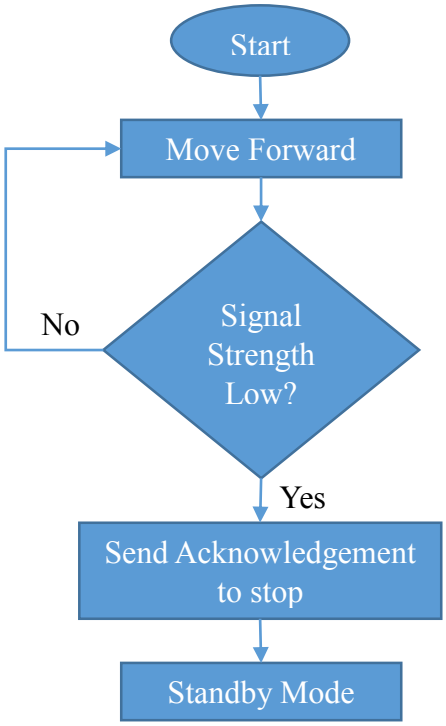


Figure 4.17 1 Node Communication Flow Chart

Base Station and 2 Nodes

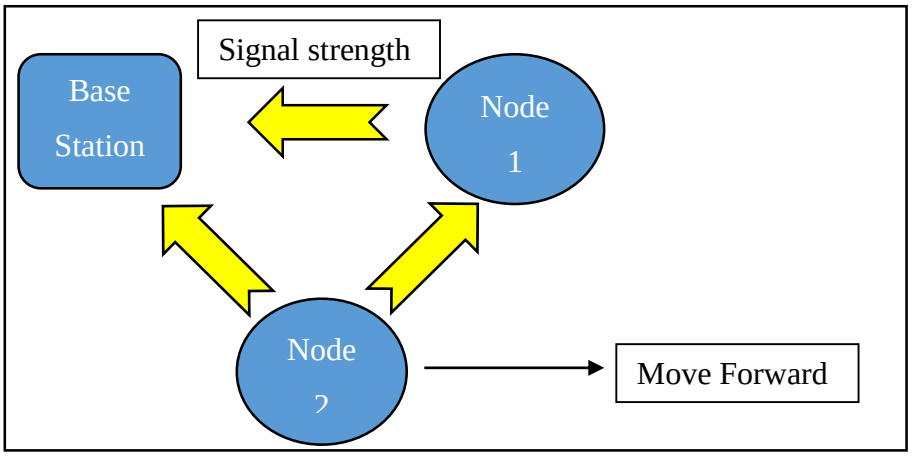


Figure 4.18 2 Nodes Communication

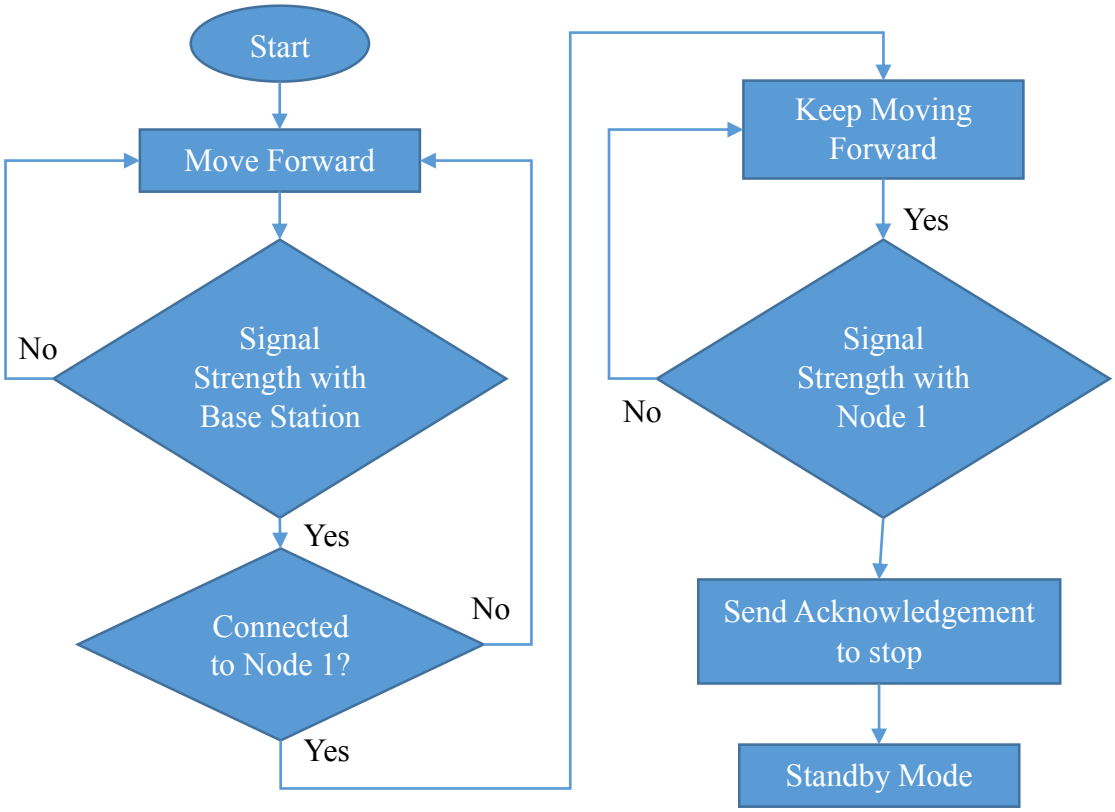


Figure 4.19 2 Nodes Communication Flow Chart

CHAPTER 5
CONCLUSION AND RECOMMENDATION

In conclusion, the main aim of this project research is to figure out an optimum wireless network communication between individual agents of flying drones to perform a multihop network system. The drones will also have to establish a network connection while taking the environment into account such as trees, hills and barriers. This requires the swarms to have a collective behavior algorithm for the agents to form a stable formation while maintaining communication allowing the drones to form a wireless network communication infrastructure for the ground swarms to communicate also. This project can be further improved by implementing a more versatile and robust routing protocols for all the nodes within the swarms for a better communication.

Overall, the objectives of the project have been met and the multihop network was implemented. Using the ZigBee as the wireless module, the Intel Galileo as the microcontroller, the data was sent to specific remote wireless module further away as desired.

CHAPTER 6
REFERENCES

1. Hauert, S., J.-C. Zufferey, and D. Floreano, *Evolved swarming without positioning information: an application in aerial communication relay.* Autonomous Robots, 2008. **26**(1): p. 21-32.

2. Perreault, L., M.P. Wittie, and J. Sheppard. *Communication-aware distributed PSO for dynamic robotic search*. in *Swarm Intelligence (SIS), 2014 IEEE Symposium on*. 2014. IEEE.

3. Barca, J.C., A. Sekercioglu, and A. Ford, *Controlling formations of robots with graph theory*, in *Intelligent Autonomous Systems 12*. 2013, Springer. p. 563-574.

4. Lee, J.-S., Y.-W. Su, and C.-C. Shen. *A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi*. in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*. 2007. IEEE.

5. *IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).* IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002), 2005: p. 0_1-580.

6. Frazier, R., *Bluetooth a boon for wireless devices.* Trade Publication, 2000. **Vol. 17**(Issue 16): p. 43.

7. Ferro, E. and F. Potorti, *Bluetooth and Wi-Fi wireless protocols: a survey and a comparison.* Wireless Communications, IEEE, 2005. **12**(1): p. 12-26.

8. Kay, R. *Ultrawideband*. 2006 Apr 10 [cited 2015 June 22]; Available from: http://www.computerworld.com/article/2563195/mobile-wireless/ultrawideband.html.

9. Wideband, U., *Poised to Transform the Wireless Industry*. 2003, On World Report.

10. Zhuang, W., X. Shen, and Q. Bi, *Ultra-wideband wireless communications.* Wireless Communications and Mobile Computing, 2003. **3**(6): p. 663-685.

11. Krasteva, R., et al., *Application of Wireless Protocols Bluetooth and ZigBee in Telemetry System Development.* Problems of Engineering, Cybernetics, and Robotics, 2005. **55**: p. 30-38.

12. Ting, K.S., et al. *The performance evaluation of IEEE 802.11 against IEEE 802.15. 4 with low transmission power*. in *Communications (APCC), 2011 17th Asia-Pacific Conference on*. 2011. IEEE.

13. Cuomo, F., et al. *Topology formation in IEEE 802.15. 4: cluster-tree characterization*. in *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*. 2008. IEEE.

14. Jung, S., A. Chang, and M. Gerla. *Comparisons of ZigBee personal area network (PAN) interconnection methods*. in *Wireless Communication Systems, 2007. ISWCS 2007. 4th International Symposium on*. 2007. IEEE.

15. Lee, T.B. *Multi-hop matters: the state of wireless mesh networking*. LAW & DISORDER / CIVILIZATION & DISCONTENTS 2009; Available from: http://arstechnica.com/tech-policy/2009/12/mesh-networks-come-of-age/.

16. Zorzi, M. and A. Armaroli. *Advancement optimization in multihop wireless networks*. in *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*. 2003. IEEE.

17. Jr., R.W.H. *Multi-Hop Networking*. Research in wireless communication and signal processing 2015 [cited 2015; Available from: http://www.profheath.org/research/multi-hop-networking/.

18. Hu, L., *Topology control for multihop packet radio networks.* Communications, IEEE Transactions on, 1993. **41**(10): p. 1474-1481.

19. Raghavendran, C.V., G.N. Satish, and P.S. Varma, *Intelligent Routing Techniques for Mobile Ad hoc Networks using Swarm Intelligence.* International Journal of Intelligent Systems and Applications (IJISA), 2012. **5**(1): p. 81.

20. Mohemmed, A.W. and N. Kamel, *Particle swarm optimization for Bluetooth scatternet formation.* 2005.

21. Zungeru, A.M., L.-M. Ang, and K.P. Seng, *Classical and swarm intelligence based routing protocols for wireless sensor networks: A survey and comparison.* Journal of Network and Computer Applications, 2012. **35**(5): p. 1508-1536.

CHAPTER 7

APPENDICES

Full coding for Transmitter (Base Station)

```
// S E N D E R
// This code demonstrates how to run the shield on Arduino and Galileo.
// A R D U I N O
//HardwareSerial* gSerialStdPtr = &Serial; // Arduino Uno, Tx(D1) Rx(D0)
//HardwareSerial* gSerialTwoPtr = &Serial; // Arduino Uno, Tx(D1) Rx(D0)
//bool gGalileo = false;
// G A L I L E O
TTYUARTClass* gSerialStdPtr = &Serial; // Galileo, /dev/ttyGSO, Tx pin
TTYUARTClass* gSerialTwoPtr = &Serial1;// Galileo, /dev/ttySO, Rx pin
#define num 48
bool gGalileo = true;
String a = "MMMMMMMMMMMMMMMMMMMMMMMMMM";
String temp = "0000";
String data = "A0A0A0A0A0";
int add1= 0,add2= 0,add3=0,add0=0;
char Pack[28];
char ACK;
int Var =1;
int Ccount = 1;
void setup()
 {
      int i;
      gSerialStdPtr->begin(9600); // Sender IDE
      gSerialTwoPtr->begin(9600); // Receiver
      waitForUser(5); // Give usr time to open serial terminal
      gSerialStdPtr->println("XBee-Sender-setup");

      for (i=0; i<20; i++)
        {
          a[i] = 65+i;
        }
void loop()
 {
    // Send data in 1 sec increments
    // gSerialStdPtr->println(Ccount);
    if(gGalileo)
      {
```

```
switch(Var)
 {
    case 1:
     //gSerialStdPtr->println("case 1 = 01"); //
     a[0] = num+0;
     a[1] = num+1;
     Pack[0] = a[0];
     Pack[1] = a[1];
     //gSerialTwoPtr->println(a);
    break;
    case 2:
     //gSerialStdPtr->println("case 2 =address");
     //temp = temp << 1;//String(add);
     //temp = String(add);
     //gSerialStdPtr->println(temp);
     for(int i=2;i<6;i++)
      {
         a[11-i] = temp[5-i];
         Pack[11-i] = a[11-i];
      }
     for(int i=6;i<=9;i++)
      {
         a[11-i] = num+0;
         Pack[11-i] = a[11-i];
      }
     //gSerialTwoPtr->println(a);
    break;
    case 3:
     //gSerialStdPtr->println("Case 3");
     for(int i=10;i<=20;i++)
      {
         a[30-i] = data[20-i];
         Pack [30-i] = a[30-i];
      }
     for(int i=20;i<=23;i++)
      {
         a[i] = 'x';
         Pack [i] = a[i];
      }
    break;
    case 4:
     //gSerialStdPtr->println("Case 4");
     a[24] = num+1;
     a[25] = num+0;
     Pack [24] = a[24];
     Pack [25] = a[25];
     break;
```

```
      }
  }
gSerialTwoPtr->println(a);
gSerialStdPtr->println("Sending");
delay(50*1);
Ccount = Ccount+1;
if(Ccount<=1)
  {
     Var=1;
  }
else if(Ccount<4)
  {
     Var=2;
     add0 = add0 +1;
     if(add0==10)
       {
          add0 = 0
          add1 = add1+1;
          if(add1==10)
            {
               add1 = 0;
               add2 = add2+1;
            }
          if(add2==10)
            {
               add2 = 0;
               add3 = add3+1;
            }
          if (add3 == 0 && add2 == 0 && add1 == 2 && add0 == 0)
            {
               add3 = 0;
               add2 = 0;
               add1= 0;
               add0 = 0;
            }
       }
     temp[3] = 48+add0;
     temp[2] = 48+add1;
     temp[1] = 48+add2;
     temp[0] = 48+add3;
  }
else if(Ccount<5)
  {
     Var=3;
  }

else if(Ccount<8)
```

```
      {
         Var=4;
      }
    if(Ccount == 10)
      {
         Ccount = 1;
         Var=1;
      }
 }
void waitForUser(unsigned int aSec)
 {
    // Give user time to bring up the serial port
    for(int i=aSec; i>0; i--)
      {
         delay(1000*1);gSerialStdPtr->print(i);
      }
    gSerialStdPtr->println("");
 }
```

Full Coding for Receiver

```
// R E C E I V E R
// This code demonstrates how to run the shield on Arduino and Galileo.
// A R D U I N O
//HardwareSerial* gSerialStdPtr = &Serial; // Arduino Uno, Tx(D1) Rx(D0)
//HardwareSerial* gSerialTwoPtr = &Serial; // Arduino Uno, Tx(D1) Rx(D0)
//bool gGalileo = false;
// G A L I L E O
TTYUARTClass* gSerialStdPtr = &Serial; // Galileo, /dev/ttyGSO, Tx pin
TTYUARTClass* gSerialTwoPtr = &Serial1; // Galileo, /dev/ttySO, Rx pin
bool gGalileo = true;
char c='A';
char add[]= [5];
char packet [26];
String STOP = "S";
int rcount=0;
int connectLED = 9;
int notconnectLED = 8;
bool qData;

void flashLed(int pin, int times, int wait)
 {
    for (int i = 0; i < times; i++)
     {
        digitalWrite(pin, HIGH);
        delay(wait);
        digitalWrite(pin, LOW);
        if (i + 1 < times)
         {
            delay(wait);
         }
     }
 }
void setup()
 {
    pinMode(connectLED, OUTPUT);
    pinMode(notconnectLED, OUTPUT);
    qData = false;                  // Initialize on reset
    gSerialStdPtr->begin(9600);       // Receiver
    gSerialTwoPtr->begin(9600);       // Sender
    waitForUser(5);                 // Give usr time to open serial terminal
    gSerialStdPtr->println("XBee-Receiver-setup");
}
void loop()
 {
    // put your main code here, to run repeatedly:
    // Give indication that no data has been received
    if(qData == false) gSerialStdPtr->println("XBee-Receiver-waiting");
```

```
      flashLed(notconnectLED, 1, 1);
    //gSerialStdPtr->println(gSerialTwoPtr->available());
    //Get data from Sender and print to Receiver serial port
    while(gSerialTwoPtr->available())
     {
        flashLed(connectLED, 1, 1);
        c= gSerialTwoPtr->read(); // Read XBee data
        packet[rcount]=c;
        rcount++;
        if(rcount>27)
         {
            rcount=0;
            gSerialStdPtr->println("received");
            for(int i=0;i<=27;i++)
             {
                /*if ((packet[9]=='0') && (packet[10]=='9'))
                 {
                    gSerialTwoPtr->println(c);
                    gSerialTwoPtr->println(STOP);
                    gSerialStdPtr->println("Received");
                    gSerialStdPtr->println("Sending ACK back to Transmitter");
                    gSerialStdPtr->println("STOP");
                 } */
                flashLed(connectLED, 1, 1);
                gSerialTwoPtr->println(c);
                packet[i]='\0';
             }
         }
        //gSerialStdPtr->write(c); // Write local
        qData = true;
     }
    rcount =1;
    // gSerialStdPtr ->println(lala);
    delay(1000);
    if(qData == false) delay(1000*1); // Slow down until data is rec
 }
void waitForUser(unsigned int aSec)
 {
    // Give user time to bring up the serial port
    for(int i=aSec; i>0; i--)
     {
        delay(1000*1);gSerialStdPtr->print(i);
     }
    gSerialStdPtr->println("");
 }
```