Multi-Hop Wireless Sensor Network for Continuous Oxygen Tank's Level Detection

by

Saeed Ahmed Saeed Al-Haddad
15780

Dissertation submitted in partial fulfilment of
the requirements for the

Bachelor of Engineering (Hons)
(Electrical & Electronic Engineering)

JANUARY 2016

Universiti Teknologi PETRONAS,
32610 Seri Iskandar, Perak Darul Ridzuan,
Malaysia

CERTIFICATION OF APPROVAL

Multi-Hop Wireless Sensor Network for Continuous Oxygen Tank's Level Detection

by

Saeed Ahmed Saeed Al-Haddad
15780

A project dissertation submitted to the
Department of Electrical & Electronics Engineering
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the

BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONIC ENGINEERING)

Approved by,

____
Dr. Azlan Bin Awang

UNIVERSITI TECHNOLOGI PETRONAS
BANDAR SERI ISKANDER, PERAK
January 2016

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 reference and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

SAEED AHMED SAEED AL-HADDAD

ABSTRACT

Wireless sensor network technology is considered as one of the modern technologies that are used in a lot of areas to measure physical, chemical or environmental variables because of their low of cost and high efficiency in data transmission. This project aims to design a wireless sensor network that will be used to measure pressure or level of Oxygen gas inside the Oxygen tanks that are found in different places in hospitals in order to overcome the problem of the manual checking of tanks level, which may cause a lot of problems because of the lack of accuracy measurement and the absence of a continuous monitoring from the control room. The designing of this project will be based on the concept of multi-hop wireless sensor network using XBee modules. XCTU software will be used to update and configure XBee modules and PROCESSING software will be used to develop a program that reads data from XBee and show it in interactive way on the screen.

ACKNOWLEDGEMENT

First and foremost, all praises to Allah almighty for his blessings and guidance. Deepest gratitude to project supervisor Dr Azlan Bin Awang, for teaching me to have high standards and not to satisfy with less than the best, for giving me the chance to do something that I enjoy and most of all for his constant guide and support. Second, I would like to express my great gratitude and thankfulness to my co-supervisor AP Dr. Fawnizu Azmadi Hussin for his continuous help in the circuit design, his guidance and motivational spirit that pushed me to the maximum to payback his valuable time and knowledge. Third I would like to thank all of those who helped and made sure the project is a success and it will be a precious experience and I would like to specifically thank Mr Azhar Bin Zainal Abidin, for his assistance in control laboratory work. Last but not least I would like to thank my family and friends for standing beside me and supporting me until the end of the project. Thank you

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

API	Application Programming Interface
DPT	Differential Pressure Transmitter
IEEE	Institute of Electrical and Electronics Engineers
MAC	Media Access Control
OSI	Open System Interconnection
WSN	Wireless Sensor Network

CHAPTER 1
INTRODUCTION

1.1 Background Study

Wireless sensor network is an innovation that has been used in a wide range of wireless environment. Nowadays there is a lot of research that investigate the path to develop a new technology that has a low power, low data rate and low cost. The importance of this technology has been expanded by the announcement of the IEEE 802.15.4 standard and the anticipated of ZigBee standard. The Alliance of ZigBee has created a wireless communication standard that gives a low cost and low consumption of power compared to other available wireless technologies, this make the ZigBee alliance to be widely used in applications that depend on automation and control. At the end the committee of IEEE and the ZigBee Alliance agreed to unite and ZigBee was the announced name for this innovation [1].

ZigBee technology is used in applications that are mainly dependent on work for long periods of time from several days to several months but does not require a high data transmission. Also ZigBee wireless devices can be implemented in larger networks and operated in the unlicensed radio frequency worldwide at 2.4GHz with 250kbps data rate [1].

1.2 Problem Statement

The Oxygen tanks sporadically found in different places in hospital, which makes it difficult for the hospital administration to check the Oxygen gas remaining in each tank directly from the control room and this need someone to go and check all the pressure transmitters in order to know the status of the gas level in each tank.

Manual checking for Oxygen tanks level in the hospital is not feasible as it may cause a lot of serious problems because of the presence of the human error and lack of accuracy in the measurement add to this the lack of continuous measurement directly from the control room and so all of these reasons may lead to situations death and the deterioration of the level of health within the hospital.

The idea of this project is to design a real time wireless sensor network to continuous monitor the level of all Oxygen tanks inside the hospital and transmit all data to central monitoring stations so that the administrations can check the status of all Oxygen tanks by only looking at a single screen in the control room.

1.3 Objectives

The objectives of this project are to:

1. Design and implement a point to point WSN using XBee modules.
2. Design and implement a multi-hop WSN using XBee modules.
3. Connect a differential pressure transmitter to a multi-hop WSN in order to transmit the data of Oxygen tanks level to a central node and show it on the screen.
4. Design a reporting system that can report the status of each Oxygen tank.

1.4 Scope of Study

The scope of this project is focusing on three main parts, firstly, pressure sensor with XBee end device node. In this part the Oxygen tank will be connected to a differential pressure transmitter using pressure input pipe to measure the pressure inside the tank. After that the pressure measuring signal wire will be connected to the input pin of XBee end device node to send measurement data continuously to a router node.

Secondly, router node and it will be used as a connection between the end device and sink node to receive and transmit data. this node will play an important role in the multi-hop WSN because it will save power consumption, cover large area and increase the efficiency of sending and receiving of data.

Lastly, sink node this node is designed to receive data from the router node and send it to the Arduino Uno board to make the necessary processing for the data before show on the computer screen.

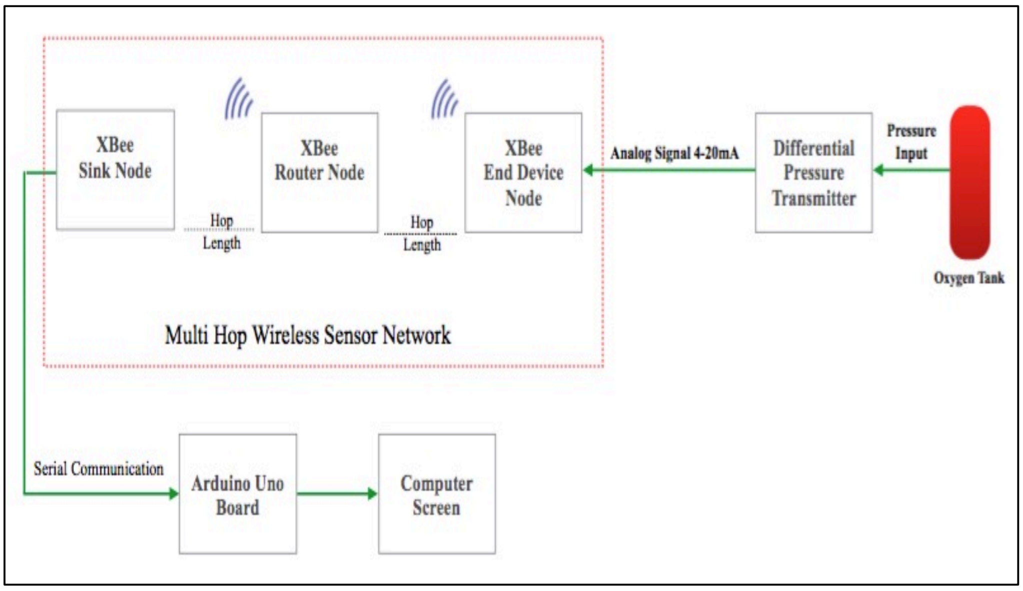


FIGURE 1.1 Oxygen tank pressure detection using multi-hop WSN

CHAPTER 2
LITERATURE REVIEW

2.1 Wireless Sensor Network

A WSN is comprise of various nodes that transmitting the data they designed to capture wirelessly. A WSN is normally consist of a power, processing, sensing and communication units. The consumption of power is the main issue that affects the use of an WSN. Hence, battery power is required in order to operate the WSN nodes independently for a long period of time with a low power consumption [2].

The nodes of WSN that are located over a wide range of area to monitor specific variables with their sensors and transmit the data between each other using wireless technology as the correspondence medium for the transmission of sensor's data to a central station [3].

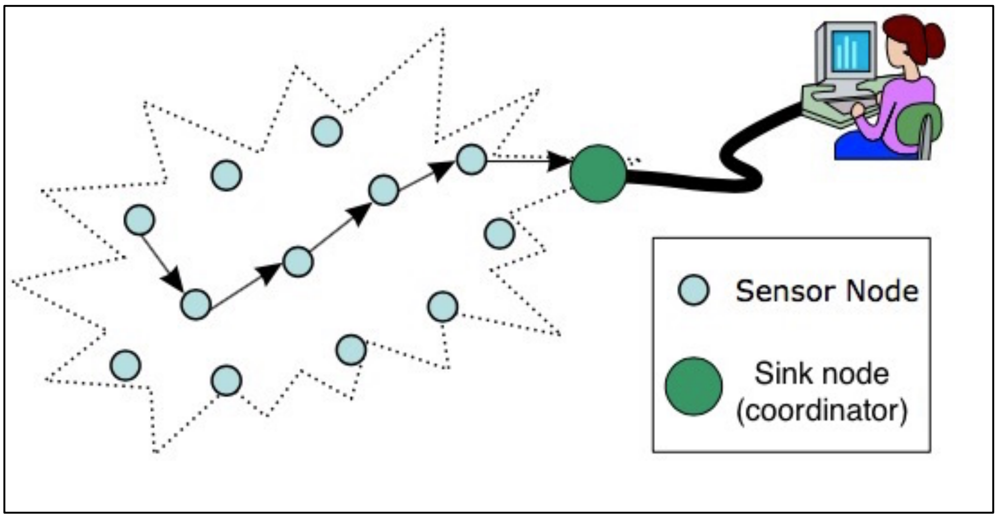


FIGURE 2.1 Typical multi-hop WSN architecture [4]

2.2 ZigBee Protocol

ZigBee is a protocol that has been developed based on an Open System Interconnection (OSI) layer model. ZigBee modules support three kinds of communication topologies; star, tree and mesh topology. ZigBee modules work with very low power consumption and low cost which makes it the most preferred wireless devices that had been used in Wireless Sensor Networks. ZigBee has the capability of multi-hop communication that supporting an unlimited range of wireless communications [5].

ZigBee promises dependable and self-configuring networks that offers low cost and low power consumption. These factors permit this technology to be specifically used the advantages of flexible mesh networking, short-range wireless protocol, strong security, a complete interoperability and well-defined application frameworks [6].

TABLE 2.1 Comparison between the existing wireless technologies [6]

	ZigBee/ IEEE 802.15.4	WLAN / IEEE 802.11b/ag	Bluetooth 1.2
Application focus	Monitoring & controlling	Web, Mail, Video	Cable Replacement
Stack size (Kbytes)	<64	>1000	>250
Battery life (days)	100 - 1000+	0.5 - 5	1-7
Network size (#nodes)	~Unlimited (65536)	Many	7
Bandwidth (kbps)	250	11000 / 54000	~1000
Range (meters)	100+	100	10+
Target BOM cost	$3	$9	$5

2.3 Multi-Hop Wireless Sensor Network and Energy Efficiency

High Advancement in the manufacturing of small, accurate and low cost sensors it becomes easier to design a multi-hop WSN for monitoring environmental or physical parameters.

In WSN there is a number of certain nodes called sensor nodes that is designed to capture a specific physical or environmental changes. Sensor nodes are found in different places from the central node (sink node) thus power management is needed for sensor nodes to minimizing power consumption because they will operate independently with a limited energy source. In order to maximize the network's life optimizing the length of the hop between the nodes will significantly extend the lifetime of the WSN [7].

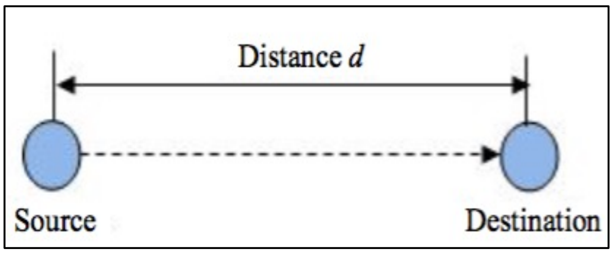


FIGURE 2.2 Transmission distance for n hops [7]

Every node is transmitting data with low power consumption required to receive the data at the receiver is equal to the sensitivity threshold P_t of the final node. Referring to figure 4 above P_t equal to [7]:

$$P_t = P_x \times (\frac{d_0}{x})^{\alpha} \qquad (1)$$

Thus, the total energy required to send the data from source to the destination equal:

$$P_x = P_t \times (\frac{x}{d_0})^{\alpha} \qquad (2)$$

Where:

d_0= Distance between source and Destination.

x = Length of the hop.

α = Path loss exponent.

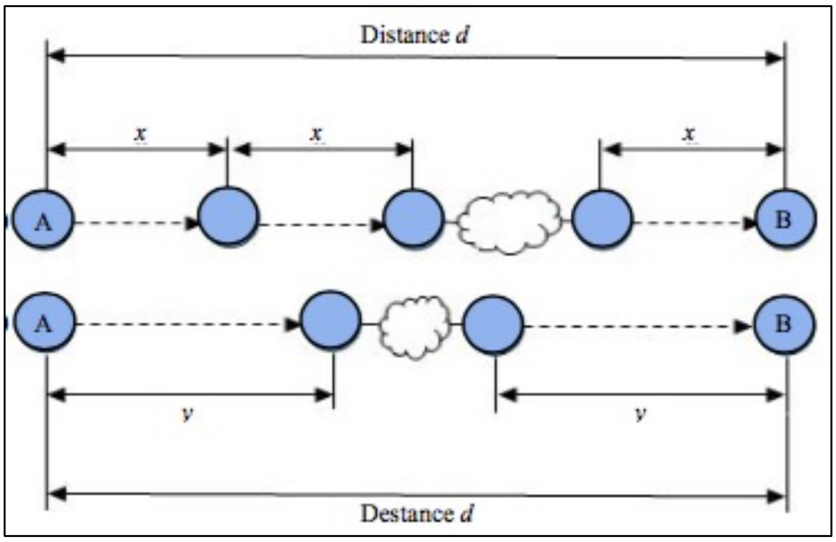


FIGURE 2.3 Transmission distance for n hops [7]

In figure 5 it shows a multi-hop WSN from source A to the destination B, in the first topology the data will be transmitted from A to B through four hops of length (x), while in the second topology the data will be transmit ted from A to B through two hops of length (y) with y=2x, thus the sensitivity threshold P_t becomes [7]:

$$P_t = P_x \times (\frac{d_0}{x})^\alpha = P_y \times (\frac{d_0}{y})^\alpha \qquad (3)$$

Since the length of y equal the double of x, the formula becomes:

$$P_y = P_x \times 2^\alpha \qquad (4)$$

Optimal length of hops in multi-hop WSN is required in order to control the consumption of power by WSN as designing a WSN with numerous short length of hops is more efficiently than longer hops [7].

2.4 Experiment Work on Multi-Hop WSN

In this experiment design of multi-hop WSN using Mica2 sensor nodes to transmit data from one node to another with a length of 760 meters and compute the energy consumption by the network with specific variety of hops and hop lengths [7].
The following table shows the calculation of the total consumption of power for different networks, with different length of hops.

TABLE 2.2 Power consumption vs hops length [7]

Hops number	Hop length (meter)	Total power consumed (mA)
6	126.666	662.7732
8	95.0	557.59
10	76.0	503.672
11	69.0909	488.4290
12	63.3333	475.3866
14	54.2857	466.9028
16	47.5	466.2968
18	42.2222	476.9244
20	38.0	491.836
21	36.1904	499.6533
22	34.4545	503.3204
24	31.6666	521.2308
26	29.2307	546.1027
28	27.1428	569.4514
30	25.3333	591.5625

From Table 2 we conclude that the most reliable range of hops that provide much less power consumption is 16 hops with hop length of 47.5 meters.

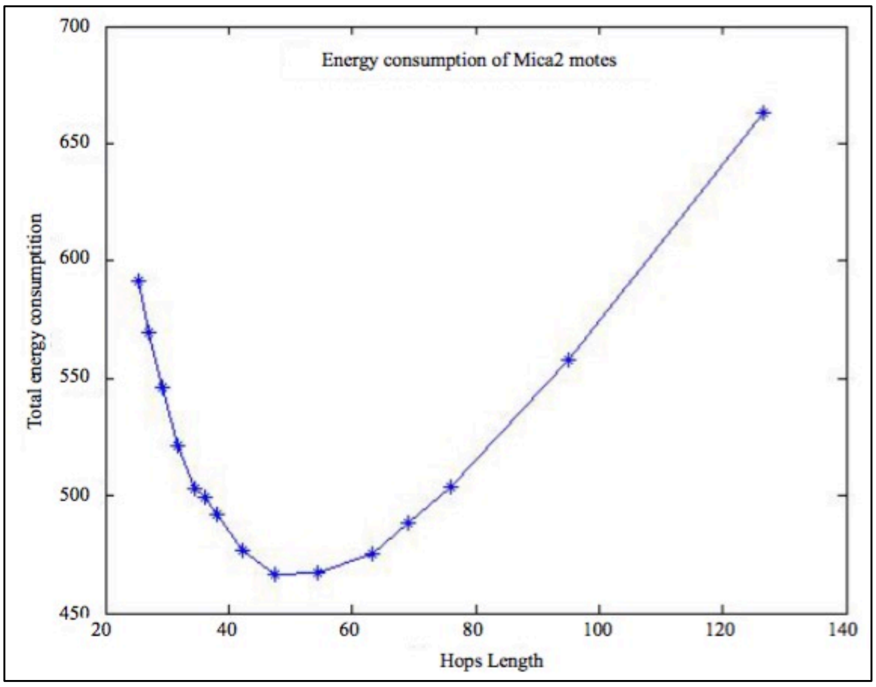


FIGURE 2.4 Mica2 energy consumption Vs hops length [7]

From the figure above we can conclude that the choice of optimal length of the hop is necessary because it provide more energy efficiency and extend the network's life.

CHAPTER 3
METHODOLOGY

The methodology of this project refers to a set of techniques that will be used to complete this project, the methodology techniques includes:

- Project Flow Chart
- Project Activities
- Gantt chart and key milestone
- Tools

3.1 Project Flow Chart

The Flow chart for this project as shown in figure 6 below:

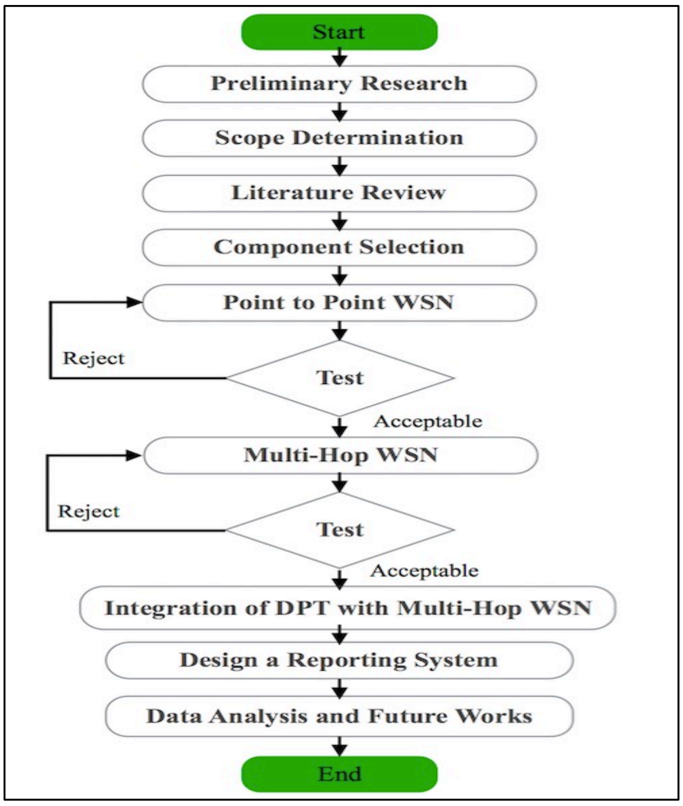


FIGURE 3.1 Project flow chart

3.2 Project Activities

The project tasks are briefly described in the table below:

TABLE 3.1 Project activities

Activity	Description
Preliminary research	Gathering information that related to the area of this project and understanding preceding research papers that involve in the field of this topic and their results.
Scope determination	Figure out the scope of the project and how it will be accomplished within the specified time.
Literature review	Understanding of the current literatures, findings and principles of the projects that relies on pressure/ level sensor and multi-hop WSN implementation.
Component selection	Research on the different XBee modules and select the suitable XBee module that support multi-hop WSN communication.
Point to point WSN implementation	Design and implement a point to point WSN.
Multi-hop WSN implementation	Design and implement a multi-hop WSN.
Connect differential pressure transmitter with multi-hop WSN	Connect the differential pressure transmitter with multi-hop WSN in control lab.
Design a reporting system	Design of reporting system using Processing software to show the data transferred on the screen.
Data analysis and future work	Evaluation of the project outcomes, compare and make a suggestion and recommendations for future development works.

3.3 Gantt Chart and Key Milestone

The two tables below show the Gantt chart and key milestone for FYP1 and FYP 2:

TABLE 3.2 Gantt chart of FYP1

Item/Week (FYP1)	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Project title selection	🟧	🟧												
Preliminary research and scope determination		🟧	🟧	🟧										
Literature review				🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
Extended proposal						🟥 *								
Component selection							🟧	🟧	🟧					
Proposal defense presentation									🟥 *					
Point to point WSN implementation									🟧	🟧	🟧	🟧		
Submission of FYP1 Draft Report													🟥 *	
Submission of FYP1 interim report														🟥 *

*** Key milestone**

TABLE 3.3 Gantt chart of FYP2

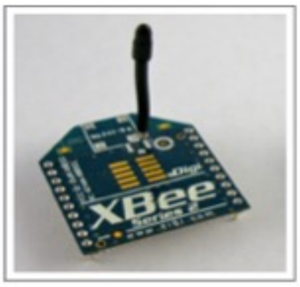
Item/Week (FYP2)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Multi-hop WSN implementation	🟧	🟧	🟧	🟧											
Connect differential pressure transmitter with multi-hop WSN			🟧	🟧											
Design a reporting system					🟧	🟧	🟧								
Progress report submission								🟥 *							
Results analysis									🟧	🟧					
Pre-SEDX presentation										🟥 *					
Future work recommendation											🟧	🟧			
Draft report submission													🟥 *		
Final Report submission														🟥 *	
VIVA															🟥 *

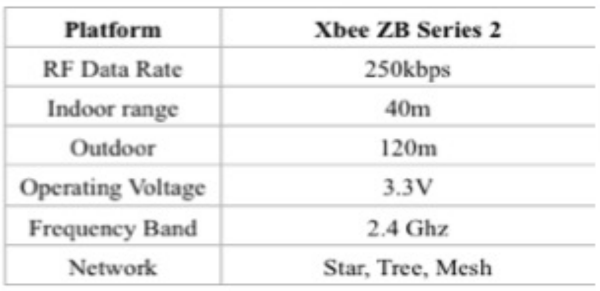
*** Key milestone**

3.4 Tools

3.4.1 Hardware

1. Three XBee modules with antenna series 2.





Platform	Xbee ZB Series 2
RF Data Rate	250kbps
Indoor range	40m
Outdoor	120m
Operating Voltage	3.3V
Frequency Band	2.4 Ghz
Network	Star, Tree, Mesh

FIGURE 3.2 XBee series 2 and it is specification

2. One Arduino Uno board.
3. One USB A to Mini B cable.
4. Two breadboards.
5. Two XBee Explorers.
6. Three breakout boards.
7. One external microcontroller.
8. One temperature sensor.
9. Two 3.3V voltage regulator.
10. One 5V voltage regulator.

3.4.2 Software

1. XCTU software to update and configure XBee modules.
2. Fritzing software to simulate the circuit of prototype.
3. Processing software to develop a reporting system.
4. Arduino software to design codes to process the data at router and sink nodes.

3.5 Procedure

3.5.1 Point to Point WSN Implementation

Point to point WSN is the first step in the implementation of this project. In this chapter we discuss the configuration of two XBee modules to make them communicate to each other using XCTU software.

3.5.1.1 Configuration of XBee Modules

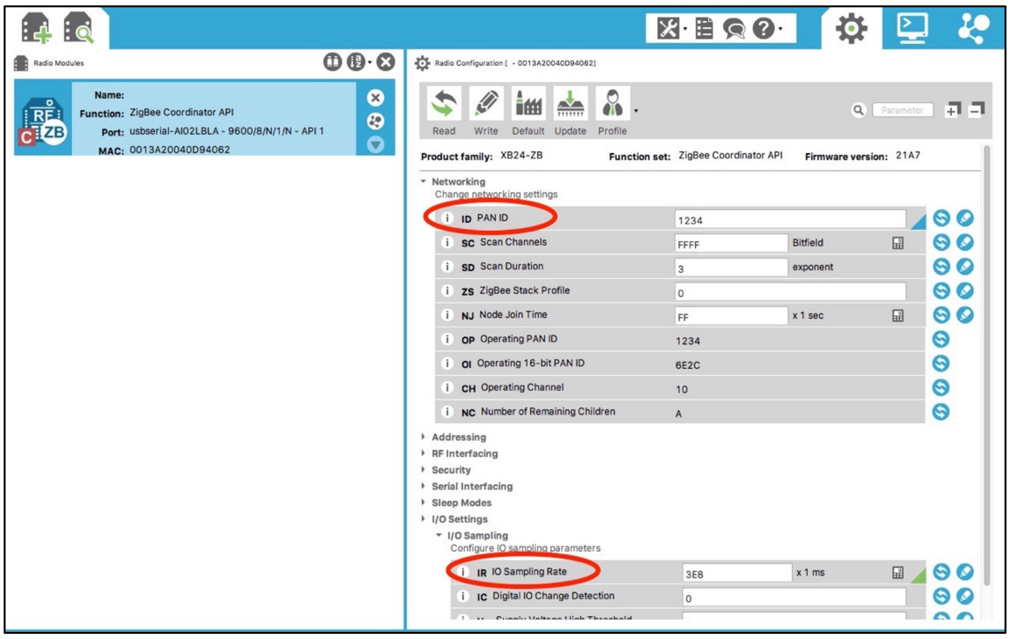
XBee is a wireless communication device that is most widely used in the formation of an WSN using ZigBee protocol. All XBee modules can work on two operation modes which is transparent mode and API mode [8]. In this project XBee's series 2 chosen to design multi-hop because they consume less power and support all kind of ZigBee communication topologies.

3.5.1.1.1 Sink Node (Coordinator)

The Sink node (Coordinator) must be configured in API (Application Programming Interface) mode because all input and output data only delivered in API mode. The sink node sends all necessary data immediately to all nodes in the same WSN to establish connection and respond to the terminal command. Once the connection is established the Sink node can receive the data from all sensor nodes in the WSN and transfer the received data using serial port to show it on the screen after making the necessary processing for the data.

Steps to configure Sink XBee using XCTU software:

1. Set the mode of the XBee as coordinator API mode.
2. Set the PAN ID with 64-bit address (for example 1234).
3. Set IR-IO sampling rate with 20 milliseconds (hex =14).
4. Click WRITE to save changes.



Name:
Function: ZigBee Coordinator API
Port: usbserial-AI02LBLA - 9600/8/N/1/N - API 1
MAC: 0013A20040D94062

Radio Configuration [- 0013A20040D94062]

Product family: XB24-ZB Function set: ZigBee Coordinator API Firmware version: 21A7

Networking
Change networking settings

ID	PAN ID	1234	
SC	Scan Channels	FFFF	Bitfield
SD	Scan Duration	3	exponent
ZS	ZigBee Stack Profile	0	
NJ	Node Join Time	FF	x 1 sec
OP	Operating PAN ID	1234	
OI	Operating 16-bit PAN ID	6E2C	
CH	Operating Channel	10	
NC	Number of Remaining Children	A	

Addressing
RF Interfacing
Security
Serial Interfacing
Sleep Modes
I/O Settings
I/O Sampling
Configure IO sampling parameters

IR	IO Sampling Rate	3E8	x 1 ms
IC	Digital IO Change Detection	0	

FIGURE 3.3 Configuration of XBee coordinator using XCTU software

3.5.1.1.2 Router Node

Before configuring the router node, we need to write down the XBee Sink node 64-bit serial number (high and low). The router node must be configured in AT mode to make it able to receive sensing data and communicate to other nodes.

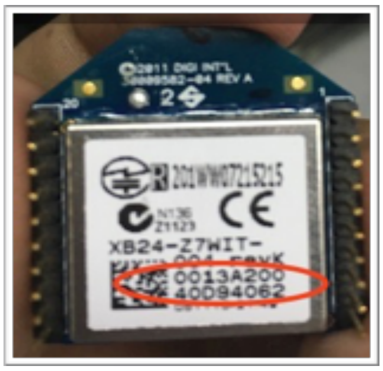
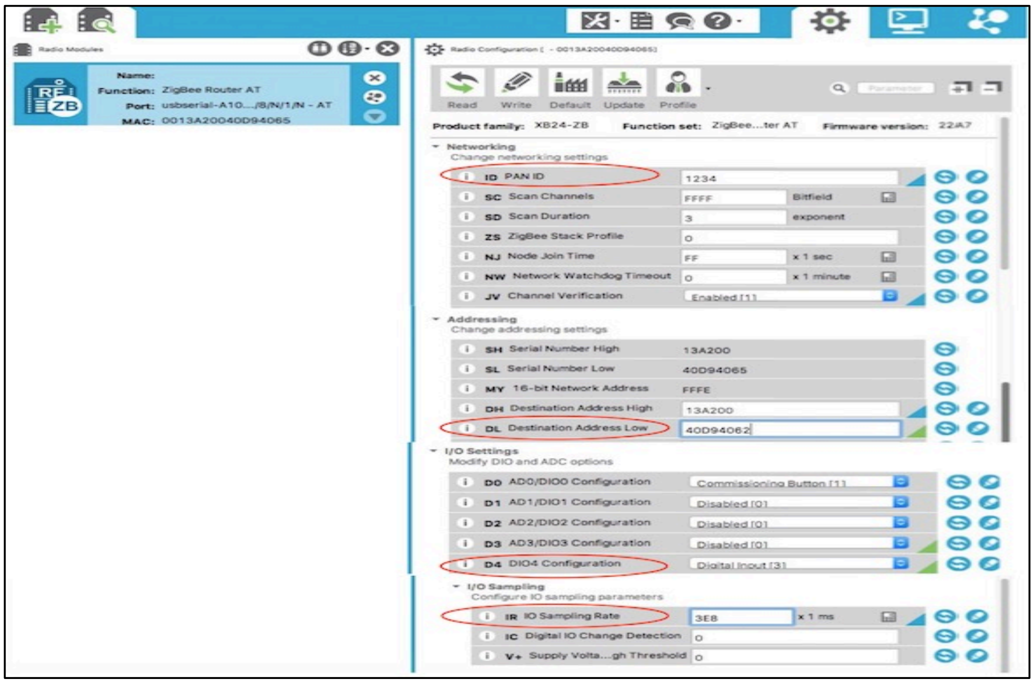


FIGURE 3.4 XBee Coordinator 64-bit serial number

Steps to configure XBee router node:

1. Set the mode of the XBee as ZigBee Router AT.
2. Set JV (Channel Verification) to 1 to ensure that the router will connect with coordinator on startup.
3. Set the PAN ID with 64-bit address (for example 1234).
4. Set DH (Destination High) and DL (Destination Low) with the serial high and low of the sink node.
5. Set D4 pin to 3 to read digital input.
6. Set IR-IO sampling rate with 20 mille Second (hex =14).
7. Click WRITE to save the changes.



Name:
Function: ZigBee Router AT
Port: usbserial-A10..../8/N/1/N - AT
MAC: 0013A20040D94065

Radio Configuration [- 0013A20040D94065]

Product family: XB24-ZB   Function set: ZigBee....ter AT   Firmware version: 22A7

Networking
Change networking settings

ID	PAN ID	1234	
SC	Scan Channels	FFFF	Bitfield
SD	Scan Duration	3	exponent
ZS	ZigBee Stack Profile	0	
NJ	Node Join Time	FF	x 1 sec
NW	Network Watchdog Timeout	0	x 1 minute
JV	Channel Verification	Enabled [1]	

Addressing
Change addressing settings

SH	Serial Number High	13A200
SL	Serial Number Low	40D94065
MY	16-bit Network Address	FFFE
DH	Destination Address High	13A200
DL	Destination Address Low	40D94062

I/O Settings
Modify DIO and ADC options

D0	AD0/DIO0 Configuration	Commissioning Button [1]
D1	AD1/DIO1 Configuration	Disabled [0]
D2	AD2/DIO2 Configuration	Disabled [0]
D3	AD3/DIO3 Configuration	Disabled [0]
D4	DIO4 Configuration	Digital Input [3]

I/O Sampling
Configure IO sampling parameters

IR	IO Sampling Rate	3E8	x 1 ms
IC	Digital IO Change Detection	0	
V+	Supply Volta....gh Threshold	0	

FIGURE 3.5 Configuration of XBee router using XCTU software

After configuring the router node connect a digital output circuit which consist of normal switch, led and 200-ohm resistor to give a digital output (High or low) to pin D3 in XBee router node.

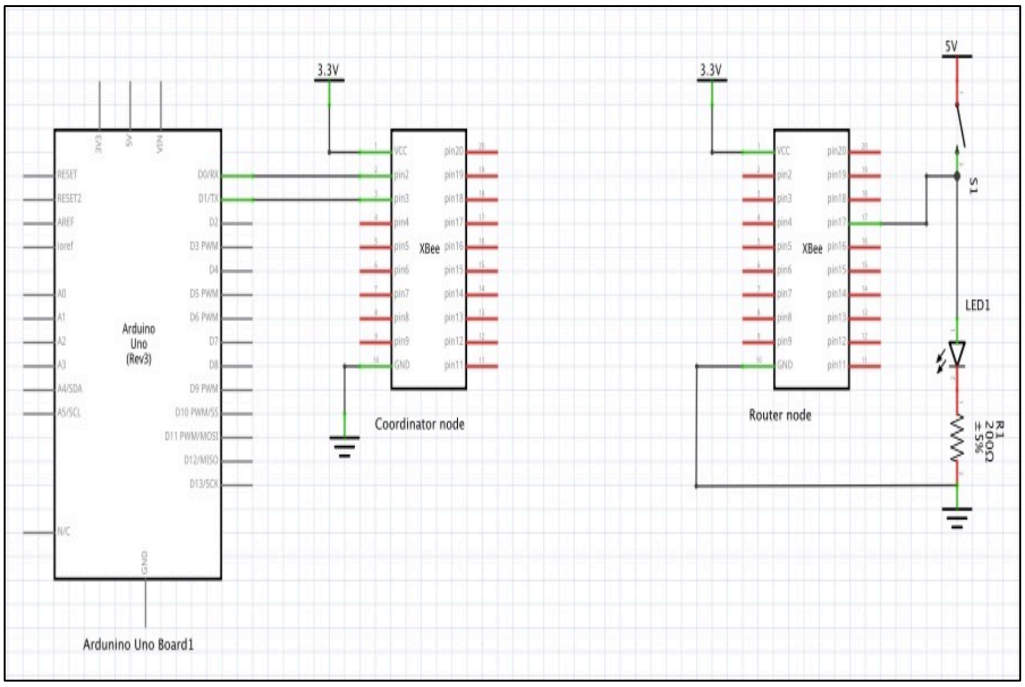


FIGURE 3.6 Point to point WSN (circuit diagram connection)

The point to point WSN circuit design successfully completed and now the circuit is ready to send and receive a digital values form the switch circuit that will give a digital output (High or Low).

3.5.2 Multi-Hop WSN Implementation with Temperature Sensor

The multi-hop WSN will be implemented based on three XBee (series 2) modules, one microcontroller, temperature sensor and one Arduino Uno board. The sensing data will be captured using the LM35 temperature sensor and it will be transmitted wirelessly through the XBee modules before it shows on the computer screen.

The design of multi-hop WSN will contain three nodes which are sensor node, router node and sink node (coordinator). The connection of each node will be different from one another but the basic structure will remain the same. All nodes will be powered by the 3.3V power supply form the Arduino Uno board.

In this implementation we will study the efficiency of sending and receiving of data and the performance of each item in the in the implementing of multi-hop WSN. Moreover, the designing issues such as code error correction and troubleshooting.

3.5.2.1 LM35 Temperature Sensor

The LM35 is a temperature sensor, whose output voltage is linearly proportional to the Celsius temperature. LM35 temperature sensor will be connected to pin 18 in router 2 node to continuously send the data to the sink node through router 1. The LM35 operates from 4 to 30 volt over a temperature range between -55℃ to 150℃ [9].

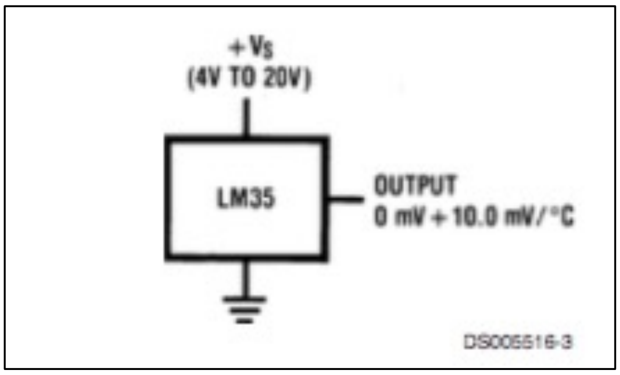
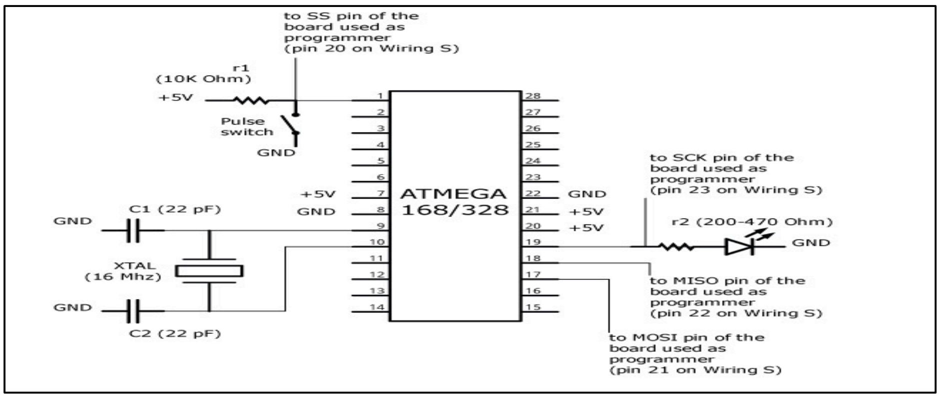


FIGURE 3.7 LM35 basic temperature sensor circuit [9]

3.5.2.2 Microcontroller

Arduino bootloader programming chip (Atmega 328P) performs a task of extracting the packet of data received at the router node, processing it and resend it again to the pin 18 of router node using a low pass filter circuit to filtered high frequency signals.



to SS pin of the board used as programmer (pin 20 on Wiring S)
r1 (10K Ohm)
+5V
Pulse switch
GND
ATMEGA 168/328
+5V
GND
C1 (22 pF)
GND
XTAL (16 Mhz)
GND
C2 (22 pF)
GND
+5V
+5V
to SCK pin of the board used as programmer (pin 23 on Wiring S)
r2 (200-470 Ohm)
GND
to MISO pin of the board used as programmer (pin 22 on Wiring S)
to MOSI pin of the board used as programmer (pin 21 on Wiring S)

FIGURE 3.8 Wiring of Atmega 328P to the board [10]

3.5.2.3 Low Pass Filter

Low pass filter consists of a resistor and a capacitor, this kind of filters allow low-frequency signal to pass through it while blocking the high-frequency signals. As we can see from the figure below a resistor is placed in series with the signal source while the capacitor is placed in parallel with signal source. Filtering process happens because of the reactive properties of a capacitor as it offers a low impedance for the high - frequency singles and high impedance for the low-frequency signals thus the high-frequency signal will take the capacitor path and low-frequency signals will take the output path [11].

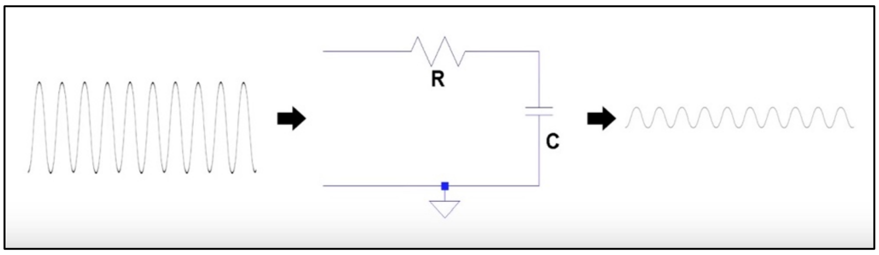


FIGURE 3.9 RC low pass filter circuit [12]

3.5.2.4 Flow of the Data

Figure below specify the flow of the data that will be sensed using LM35 temperature sensor until it reach to the destination.

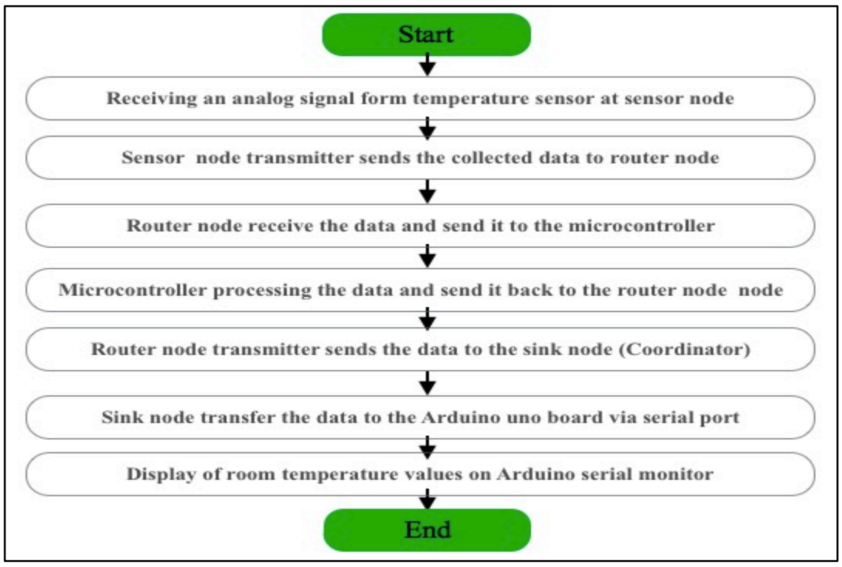

Start
Receiving an analog signal form temperature sensor at sensor node
Sensor node transmitter sends the collected data to router node
Router node receive the data and send it to the microcontroller
Microcontroller processing the data and send it back to the router node node
Router node transmitter sends the data to the sink node (Coordinator)
Sink node transfer the data to the Arduino uno board via serial port
Display of room temperature values on Arduino serial monitor
End

FIGURE 3.10 Flow of the data in multi-hop WSN

3.5.2.5 XBee Modules Setup

For this implementation of a multi-hop WSN ZigBee protocol will be used in the formation of the network. Building a multi-hop WSN using ZigBee protocol is easier as ZigBee is self-healing and self-organizing protocol. The coordinator is responsible for the formation of the network as it will be establishing the connection with other nodes that has the same PAN ID.

The setting for each XBee module in the network:

Sink node (coordinator):

- Set the mode of the XBee as coordinator API mode.
- Set the PAN ID with 64-bit address (for example 1234).
- Set IR-IO sampling rate with 100 milliseconds (hex =3E8).
- Click WRITE to save changes.

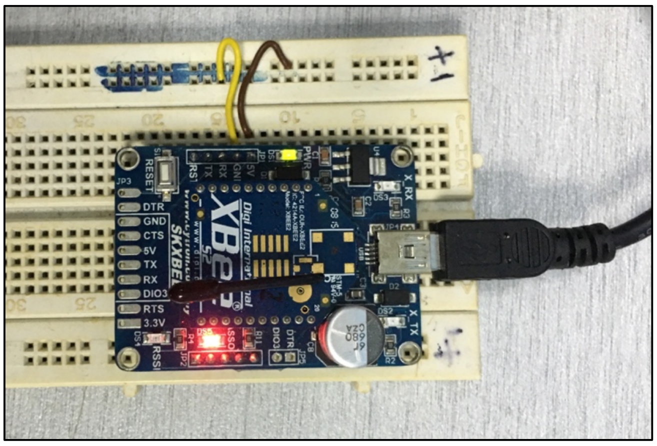


FIGURE 3.11 Sink node (coordinator)

Router node:

- Set the mode of the XBee as ZigBee Router API.
- Set JV (Channel Verification) to 1 to ensure that the router will connect with coordinator on startup.
- Set the PAN ID with 64-bit address (for example 1234).
- Set DH (Destination High) and DL (Destination Low) with the serial high and low of the coordinator.
- Set D3 pin to 2 (ADC).
- Set IR-IO sampling rate with 100 mille Second (hex =3E8).
- Click WRITE to save the changes.

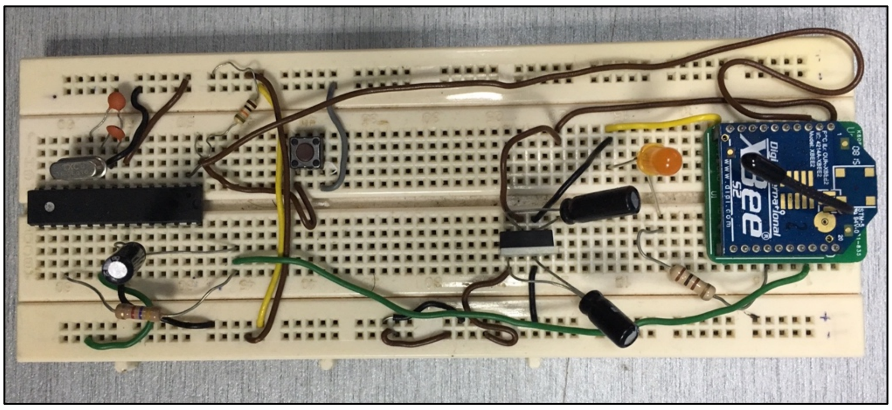


FIGURE 3.12 Router node connected with Atmega 328-P Microcontroller

Sensor node:

- Set the mode of the XBee as ZigBee Router AT.
- Set JV (Channel Verification) to 1 to ensure that the router will connect with coordinator on startup.
- Set the PAN ID with 64-bit address (for example 1234).
- Set DH (Destination High) and DL (Destination Low) with the serial high and low of the router node.
- Set D3 pin to 2 (ADC).
- Set IR-IO sampling rate with 100 mille Second (hex =3E8).
- Click WRITE to save the changes.

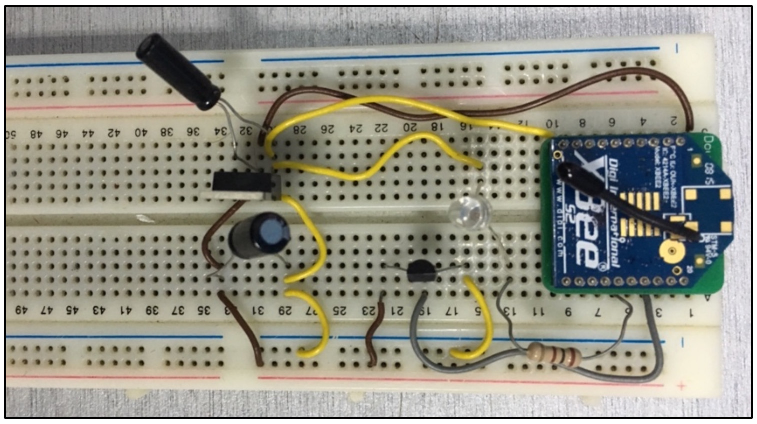


FIGURE 3.13 Sensor node connected with LM35 temperature sensor

After the configuration of each XBee module the multi-hop WSN is ready to work and the temperature values will be sent as packets of data over the multi-hop WSN.

Once the circuit connection is done we connect the sink node (Coordinator) to the PC and run XCTU software to scan the network and shows the connected nodes.

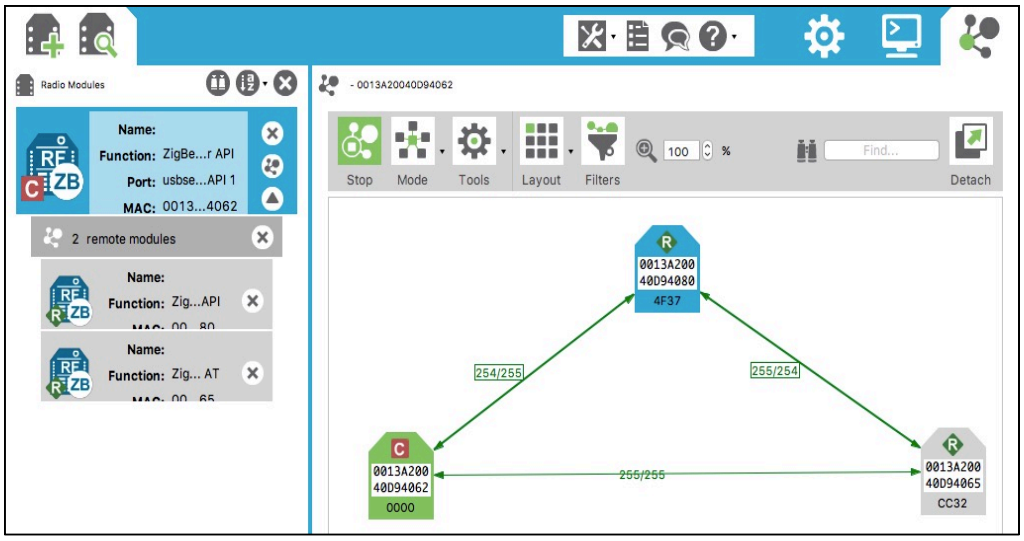


FIGURE 3.14 Multi-hop WSN (XCTU software)

In the figure below we see that the multi-hop WSN consist of sensor node, router node and one Coordinator, the data sending will follow the path from the sensor node to the router node until it reaches the destination which is sink node (Coordinator).

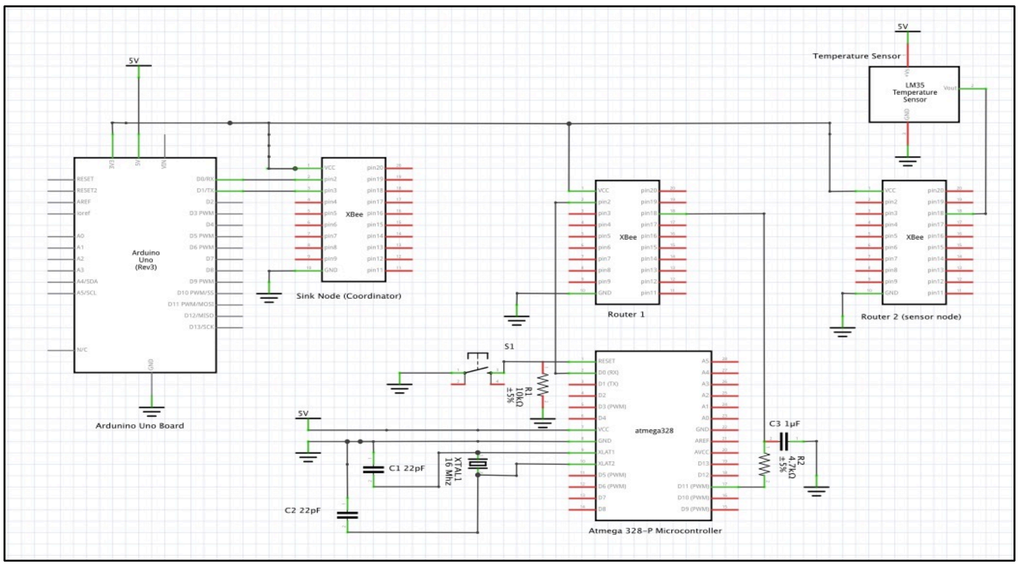


FIGURE 3.15 Multi-hop WSN circuit connection

3.5.3 Integration of Differential Pressure Transmitter with the Multi-Hop WSN

After the implementation of multi-hop wireless sensor network with the LM35 temperature sensor now we will move to the integration of the differential pressure transmitter with the sensor node to send the pressure data to the sink node.

In this section we will connect the signal wires from the differential pressure transmitter to the sensor node and leave all the other settings and parameters of the XBee modules as the same as we do for the temperature sensor.

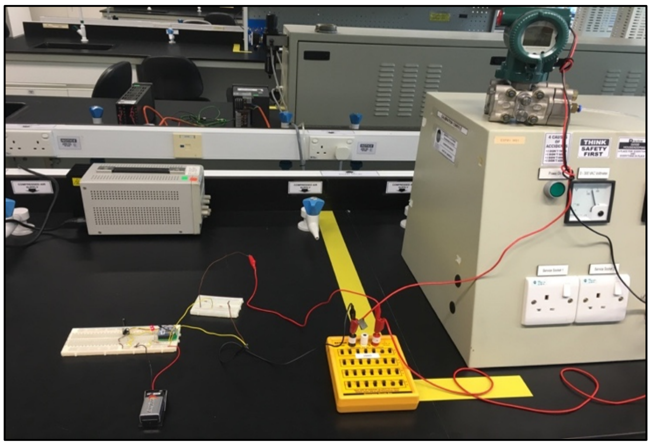


FIGURE 3.16 Integration of differential pressure transmitter with sensor node

The differential pressure transmitter needs 24 DC volt power supply to work and by ground both differential pressure transmitter and XBee sensor node the circuit will be ready to transmit pressure data wirelessly by connect the signal wire to pin 17 in XBee module.

3.5.4 Designing a Reporting System using Processing Software

Processing is an open source software development environment designed for novice programmers and geared toward visual displays. It used to design a graphical user interface that will be more easy to monitor and control variables instead of normal display.

In this project we have used processing software to design a graphical user interface for both temperature and pressure values based on some mathematical equations that will convert the digital values received from the XBee modules into normal values whether for temperature or pressure values.

In the outlining of the code we utilized two units to show the values so that the code will be more flexible and the reading of the data will be easier also to increase the accuracy and detection of errors if exist.

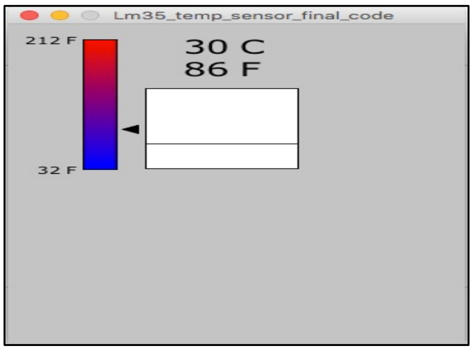


FIGURE 3.17 Graphical user interface using processing software (Temperature)

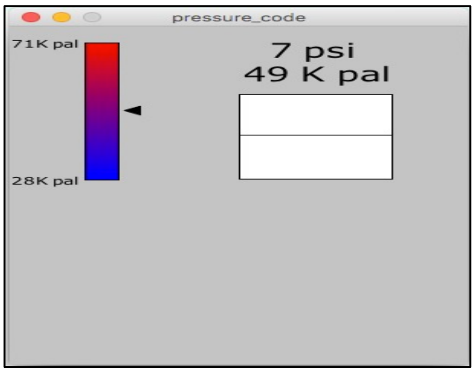
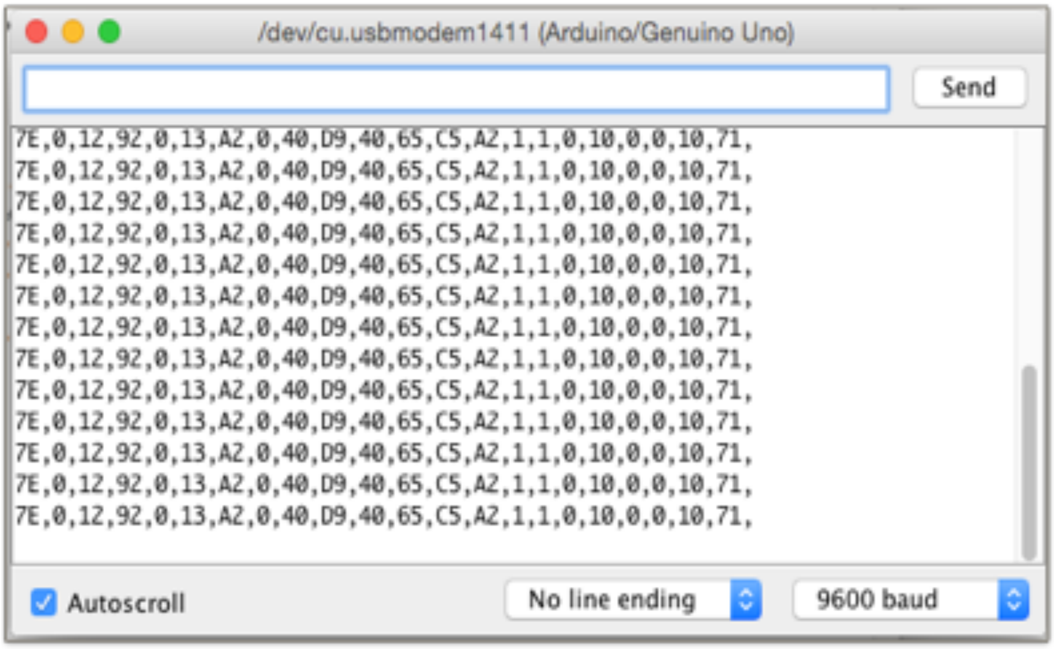


FIGURE 3.18 Graphical user interface using processing software (Pressure)

CHAPTER 4
RESULT & DISCUSSION

4.1 Sending and Receiving of Data (Point to Point WSN)

After making the connection of point to point WSN, the sending of data from the router node will be high or low. In the sink node the data will be received and transferred to Arduino Uno board to make the necessary processing for the data by the developing code to show it on the computer screen.



```
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,10,0,0,10,71,
```

FIGURE 4.1 Data receiving structure

A brief description of the packet structure received is given below:

IO data Sample RX Indicator (API 1)

7E 00 12 92 00 13 A2 00 40 D9 40 65 C5 A2 01 01 00 10 00 00 10 71

Start delimiter: 7E, this indicates the beginning of data frame.

Length: 00 12 (18) , this indicates the number of bytes except the beginning byte and checksum byte.

Frame type: 92.

64-bit router address: 00 13 A2 00 40 D9 40 65

16-bit source address: C5 A2, network address

Receive options: 01, (01= Packet acknowledged, 02= Broadcast packet).

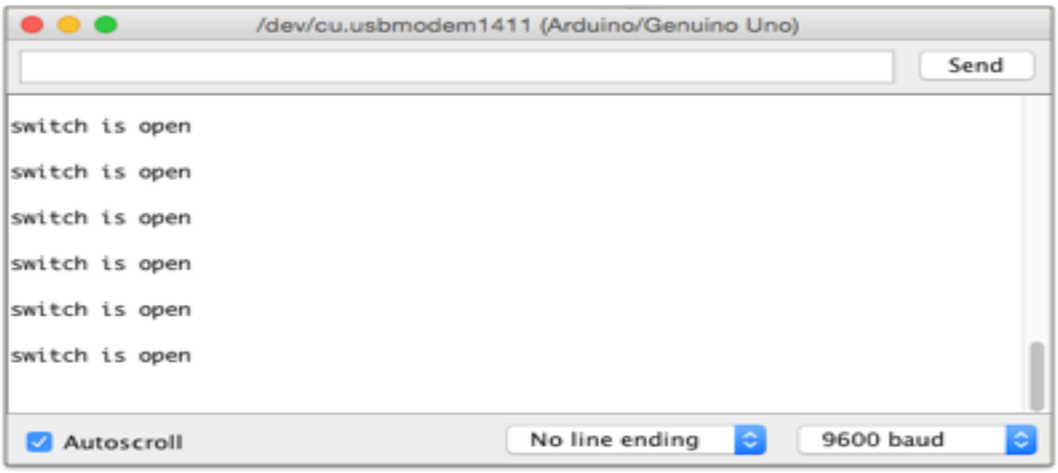
Number of samples: 01, always set to 01 due to XBee limitation.

Digital channel mask: 00 10, indicates which pin is configured to send data in this case it is pin 4 configured as digital input pin.

Analog channel mask: 00

DIO4/AD4 digital value: High

Checksum: 71, this byte basically comes at the receiving end to check and see if there was a transmission error.



```
switch is open
switch is open
switch is open
switch is open
switch is open
switch is open
```

FIGURE 4.2 Receiving of digital input data

4.2 Sending and Receiving of Data (Multi-Hop WSN) Temperature Values

The LM35 temperature sensor will start sensing and the temperature values will be sent to sensor node. By measuring the output voltage from the Lm35 temperature sensor using the Multimeter equal to 0.333 volt, thus the temperature value equals to (0.333/0.01) = 33.3 °C.

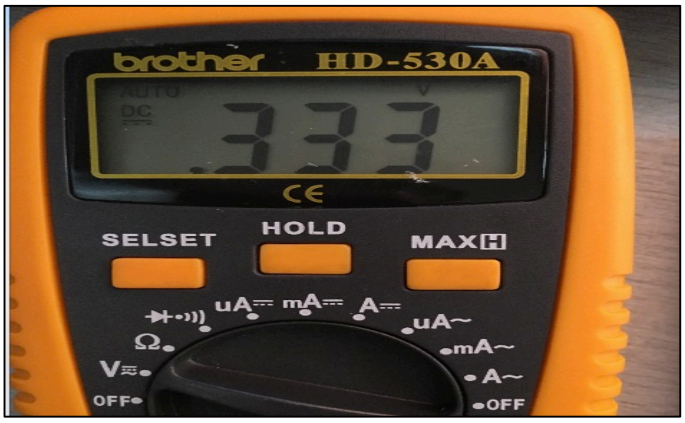
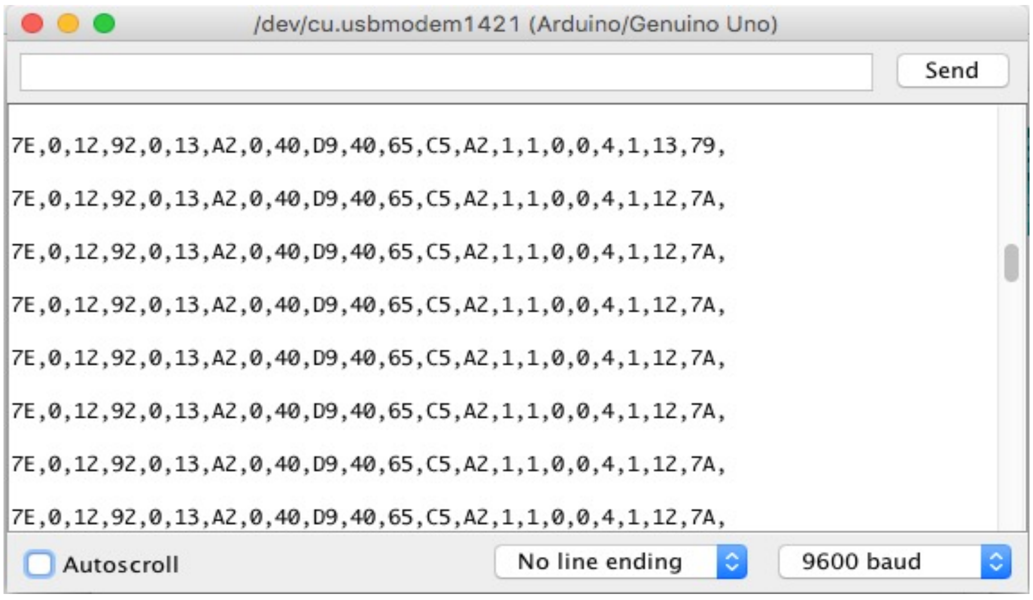


FIGURE 4.3 Measured voltage output from LM35 temperature sensor

Now we will send the temperature values in a multi-hop WSN using XBee modules and after the data reaches the destination we will compare it with the measured values.

4.2.1 Data Received at Router Node

Figure below shows the packets of data received at Router node. The packets of data will be shown on Arduino Uno serial monitor after uploading code 2 in the appendixes to the Arduino Uno board.



```
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,13,79,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,12,7A,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,12,7A,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,12,7A,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,12,7A,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,12,7A,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,12,7A,
7E,0,12,92,0,13,A2,0,40,D9,40,65,C5,A2,1,1,0,0,4,1,12,7A,
```

FIGURE 4.4 Structure of data received at router node

A brief description of the packet structure received at router node:
IO data Sample RX Indicator (API 1)

7E 00 12 92 00 13 A2 00 40 D9 40 65 C5 A2 01 01 00 00 04 00 E9 71

Start delimiter: 7E, this indicates the beginning of data frame.
Length: 00 12 (18), this indicates the number of bytes except the beginning byte and checksum byte.
Frame type: 92.
64-bit router address: 00 13 A2 00 40 D9 40 65
16-bit source address: C5 A2, network address
Receive options: 01, (01= Packet acknowledged, 02= Broadcast packet).
Number of samples: 01, always set to 01 due to XBee limitation.

Digital channel mask: 00 00.
Analog channel mask: 00
DIO3/AD3 analog value: High, this indicates which pin is configured as ADC in this case it is pin 18.
Analog Sample data: 01 12, this indicates the temperature value in hexadecimal.

Checksum: 7A, this byte basically comes at the receiving end to check and see if there was a transmission error.

After the data received we convert the temperature value from hexadecimal to decimal, thus:
01 12 hexadecimal equal to:
$(0*16^3 + 1*16^2 + 1*16^1 + 2*16^0) = 274$ (Decimal).

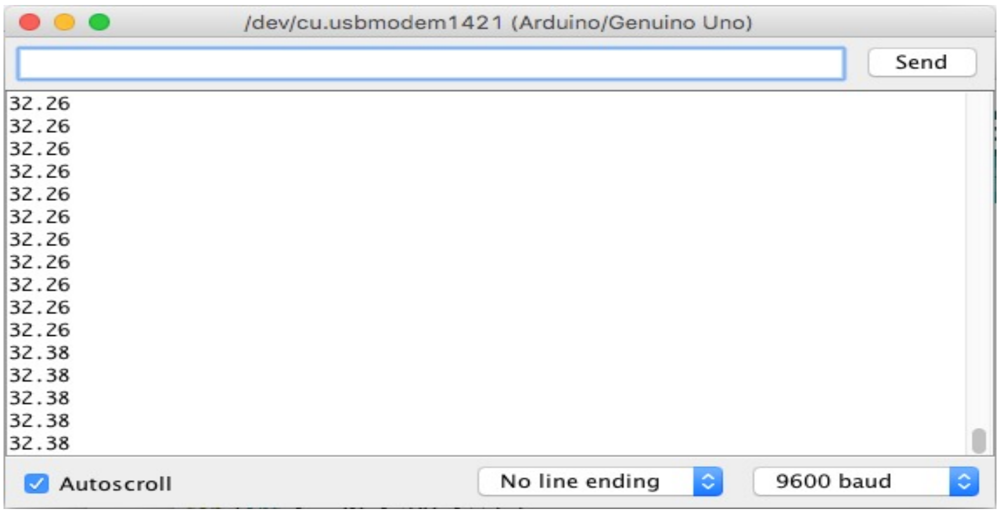
Since the voltage reference for the XBee modules (series 2) is 1.2 volt, and the analog sample values are ranged between 0 and 1023 (0 = 0 volt, 1023 = 1.2 volt).
To convert the 233decimal value to volt we divide it by 852.5, thus the equation becomes: (274/852.5) = 0.321 volt.

To convert the decimal value to temperature value we need to upload the code 3 in the appendixes to the microcontroller using Arduino Uno board, the code will read the packets of the data and it will discard unnecessary data until it reaches to the analog sample value. Which is the important value in the packet that we need to processing it using the microcontroller in order to see the temperature value.

The microcontroller will convert the analog sample value from hexadecimal to decimal value and after that it will convert the decimal value to temperature value using this equation:

```
temp_value_router_node = (analog_sample_value *0.117302053);
Serial.println(temp_value_router_node);
```

temp_value_router_node = (274*0.117302053) = 32.14 $^{\circ}$C.



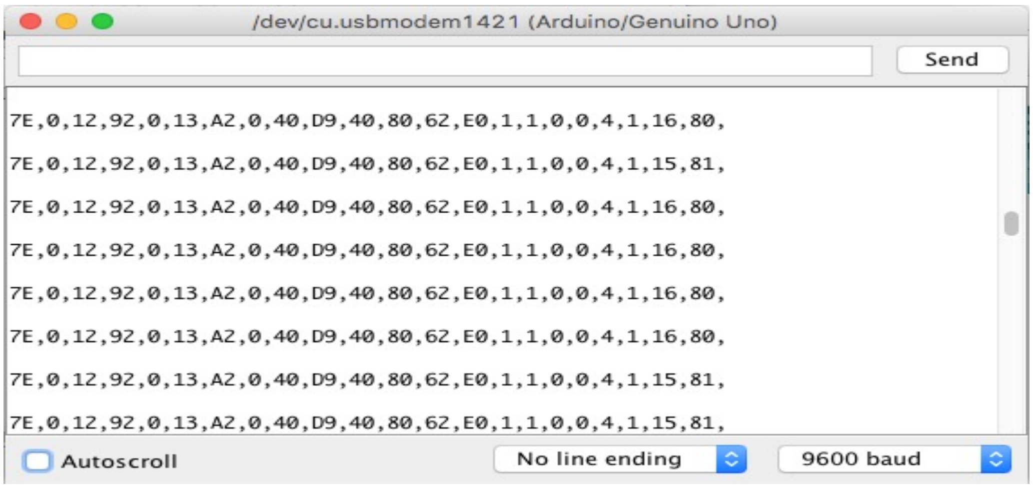
```
32.26
32.26
32.26
32.26
32.26
32.26
32.26
32.26
32.26
32.26
32.26
32.38
32.38
32.38
32.38
32.38
```

FIGURE 4.5 Display of temperature values received at router node

The microcontroller will be processing the data received at router node and it will send it back to pin 18 of router node to transmit the data again to the sink node (coordinator).

4.2.2 Data Received at Sink Node (Coordinator)

Figure below shows the packets of data received at sink node (Coordinator) from router 1. The packets of data will be shown on Arduino Uno serial monitor after uploading code 2 in appendixes to the Arduino Uno board.



```
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,16,80,
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,15,81,
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,16,80,
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,16,80,
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,16,80,
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,16,80,
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,15,81,
7E,0,12,92,0,13,A2,0,40,D9,40,80,62,E0,1,1,0,0,4,1,15,81,
```

FIGURE 4.6 Structure of data received at sink node (Coordinator)

A brief description of the packet structure received at sink node (Coordinator):
IO data Sample RX Indicator (API 1)

7E 00 12 92 00 13 A2 00 40 D9 40 65 C5 A2 01 01 00 00 04 00 E9 71

Start delimiter: 7E, this indicates the beginning of data frame.
Length: 00 12 (18), this indicates the number of bytes except the beginning byte and checksum byte.
Frame type: 92.
64-bit router address: 00 13 A2 00 40 D9 40 80
16-bit source address: C5 A2, network address
Receive options: 01, (01= Packet acknowledged, 02= Broadcast packet).
Number of samples: 01, always set to 01 due to XBee limitation.
Digital channel mask: 00 00.
Analog channel mask: 00
DIO3/AD3 analog value: High, this indicates which pin is configured as ADC in this case it is pin 18.

Analog Sample data: 01 15, this indicates the temperature value in hexadecimal.
Checksum: 81, this byte basically comes at the receiving end to check and see if there was a transmission error.

After the data received we convert the temperature value from hexadecimal to decimal, thus:
01 15 hexadecimals equal to:
$(0*16^3 + 1*16^2 + 1*16^1 + 5*16^0) = 277$ (Decimal).

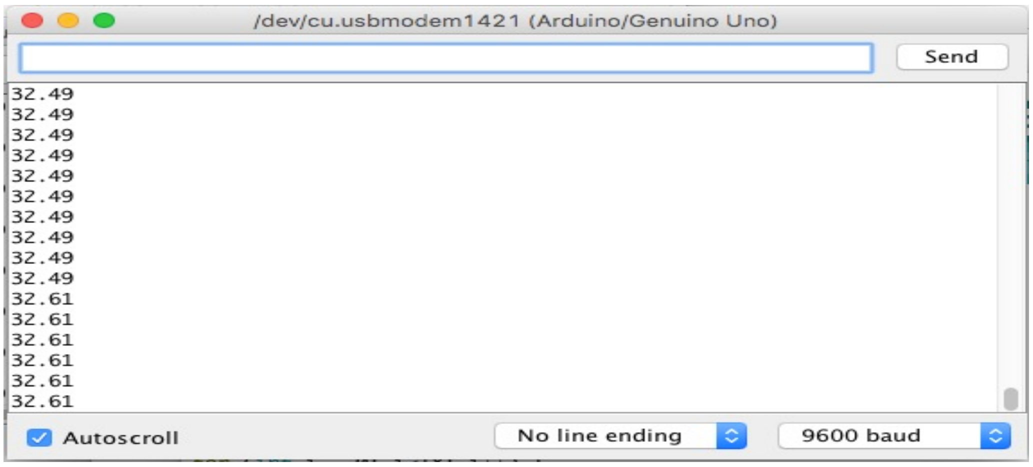
Since the voltage reference for the XBee modules (series 2) is 1.2 volt, and the analog sample values are ranged between 0 and 1023 (0 = 0 volt, 1023 = 1.2 volt).
To convert the 233decimal value to volt we divide it by 852.5, thus the equation becomes: (277/852.5) = 0.324 volt.

To convert the decimal value to temperature value we need to upload the code 4 in the appendixes to the microcontroller using Arduino Uno board, the code will read the packets of the data and it will discard unnecessary data until it reaches to the analog sample value. Which is the important value in the packet that we need to processing it using the microcontroller in order to see the temperature value.

The microcontroller will convert the analog sample value from hexadecimal to decimal value and after that it will convert the decimal value to temperature value using this equation:

```
temp_value_coordinator = (analog_sample_value *0.117302053);
Serial.println(temp_value_coordinator);
```

temp_value_coordinator = (277*0.117302053) = 32.49 $^{\circ}$C.



```
32.49
32.49
32.49
32.49
32.49
32.49
32.49
32.49
32.49
32.49
32.61
32.61
32.61
32.61
32.61
32.61
```

FIGURE 4.7 Display of temperature values received at sink node (Coordinator)

For the pressure values it will take the same procedure for sending and receiving of data, in the pressure we need to change the mathematical equation that used in the code for the calculation of the pressure values.

4.3 Difference Between the Measured and Received Data (Temperature Values)

After data is received now we will compare the result of sending and receiving of temperature values

Measured temperature from LM35 temperature sensor = 33.3 °C.

Received temperature at router node = 32.14 °C.

Received temperature at sink node (coordinator) = 32.49 °C.

Percentage of error equal the difference between the measured value and the exact value.

Percentage of error $= \frac{Approximate\ value - exact\ value}{Exact\ value}$ X 100% (5)

Thus the percentage of error for each stage is:

Router node:

Percentage of error $= \frac{33.3-32.14}{32.14} \times 100\% = 3.61\%$

Sink node (Coordinator):

Percentage of error $= \left|\frac{32.14-32.49}{32.49}\right| \times 100\% = 1.07\%$

The overall percentage of error between the measured value and the received value at the sink node (Coordinator):

Percentage of error = $\frac{33.3-32.49}{32.49} \times 100\% = 2.49\%$

After the calculation of percentage of error in each stage (router node and sink node), we can see that there is a quiet difference in the temperature value between the measured value and the received value at router node this error may be occurring because of the delay in the sending of data packets.

Because of the using of low pass filter circuit in router node that will clear the signal and eliminate the high frequency signals from passing and allow only the low frequency signals to pass through the output. We can analyze the small difference in the percentage of error between the temperature value received at router node and the temperature value received at the sink node (Coordinator).

The overall percentage of error equal to 2.49% and the sending and receiving of data is acceptable with a difference between the measured and received temperature value equal to 0.81 °C. Same procedure will be followed for the calculation of the error in the pressure values.

4.4 Average Delay Time Between the Received Packets

The delay time specifics how long it will take for a packet of data to travel across the multi-hop wireless sensor network. In this section we will discuss the average delay time taken for the packets to be received at sink node and this measurement will be conducted using the XCTU software as it gives the times received for each packet.

In the configuration of the XBee modules we specify the sampling time in both sensor and router nodes. We have configured the XBee modules with a sampling times of (50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600) ms respectively and we have calculated the average delay time for each sampling time.

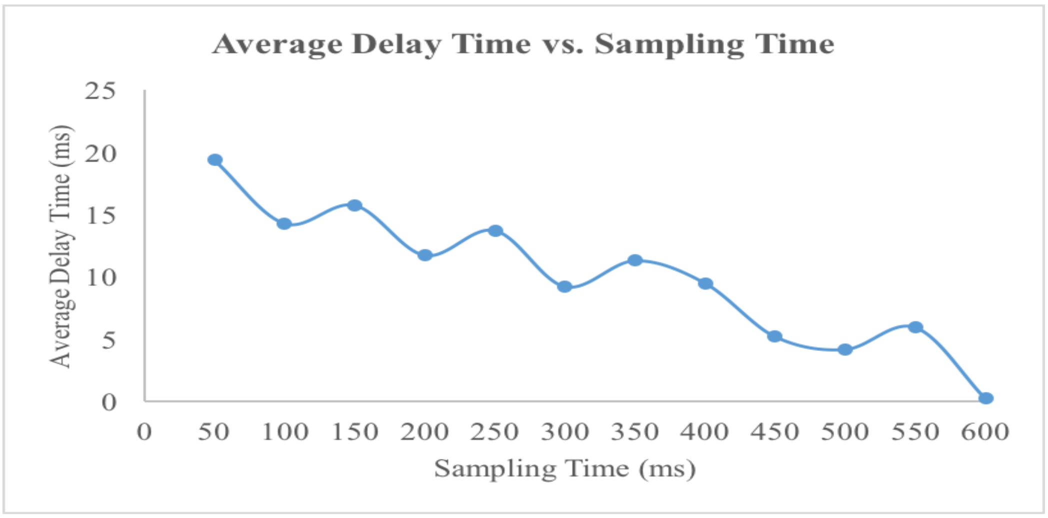


FIGURE 4.8 Average delay time vs. sampling time

From the figure above we can see the average delay time for each sampling time starting with 50 ms with almost 20ms average delay time. The average delay time is decreased by increasing of the sampling time until it reaches to 0.196 ms for the 600 ms sampling time. We can conclude that by increasing the sampling time the average delay time will be decreased.

CHAPTER 5
CONCLUSION & RCOMMENDATION

5.1 Conclusion

The designing of multi-hop WSN successfully completed with the integration of both temperature sensor and differential pressure transmitter. The sending and receiving of data is working perfectly with a small delay time and a small percentage of error between the measured and the received data. The multi-hop wireless sensor network can can be used in monitoring and controlling systems in different places such as hospitals, companies, industries and public places, by using the XBee modules to form the multi-hop wireless sensor network it becomes easier to control a lot of variables at the same time with a small delay time and high level of security.

5.2 Recommendation

It is recommended to study the power consumption by the XBee modules in a large wireless sensor network and also the distance covered by the XBee modules to make this project more valuable and improve it to the advanced level.

REFERENCES

[1] N. Patel, H. Kathiriya and A. Bavara, “Wireless Sensor Network Using ZigBee”, *IJRET*, vol. 2, pp. 1038- 1042, June. 6, 2013.

[2] E. Yaccoub and A. Abu-Dayya. “ Multihop Routing for Energy Efficiency in Wireless Sensor Networks” in *Wireless Sensor Networks - Technology and Protocols,* M.A. Matin, InTech, 2012, pp. 165-188.

[3] Akyildiz LF., Su W, Sankarasubramaniam Y., and Cayirci E, "Wireless Sensor Networks: A Survey," Communication Magazine, vol. 40, no. 8, pp. 102-114, August 2002.

[4] Wikipedia. “Wireless sensor network”, Wikipedia.org. [Online]. Available https://en.wikipedia.org/wiki/Wireless_sensor_network [Last Modified: 8 Dec 2015, 09:21].

[5] R. A. Rashid, M. A. Sarijari and F. Mohamed. *Applications and System Design for Wireless Sensor Network.* Malaysia: Penerbit Universiti Teknologi Malaysia, 2008.

[6] Ferdinando, Fidelix, Francis, Coutinho, Samuel; Rocha, Monica L.“ZigBee For Building Control Wireless Sensor Networks”. Microwave and Optoelectronics Conference, 2007. IMOC 2007. SBMO/IEEE MTT-S International Oct. 29 2007-Nov. 1 2007 Page(s):511 – 515.

[7] Kheireddine, M. and Abdellatif, “Analysis of Hops Length in Wireless Sensor Networks, *SRC*, pp. 109-117, June. 17, 2014.

[8] S. C. Mukhopadhyay. *Intelligent Sensing, Instrumentation and Measurements*. Palmerston North, NZ: Springer, 2013.

[9] National Semiconductor, “Precision Centigrade Temperature Sensors,” LM35 datasheet, Nov. 2000 [Revised Mar. 2016].

[10] Wiring. "Burning a bootloader into a brand new DIP atmega168/328part to use it with Wiring," wiring.org.co. [Online]. Available: http://wiring.org.co/learning/tutorials/atmegaDIPbootloader/ [Accessed: Mar. 6, 2016].

[11] Learningaboutelectronics.“LowPassFilterCalculator,”learningaboutelectronics.com.[Online].Available:http://www.learningaboutelectronics.com/Articles/Low-pass-filter-calculator.php#answer1 [Accessed: Mar. 6, 2016].

[12] YouTube. “Passive RC low pass filter tutorial!”. *Youtube.com.* [Online]. Available https://www.youtube.com/watch?v=OBM5T5_kgdI [Accessed: 6 March 2016].

[13] Vieira, M., Coelho, C., da Silva, D. & da Mata, J. [2003]. Survey on wireless sensor network devices, Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation, pp. 537–544.

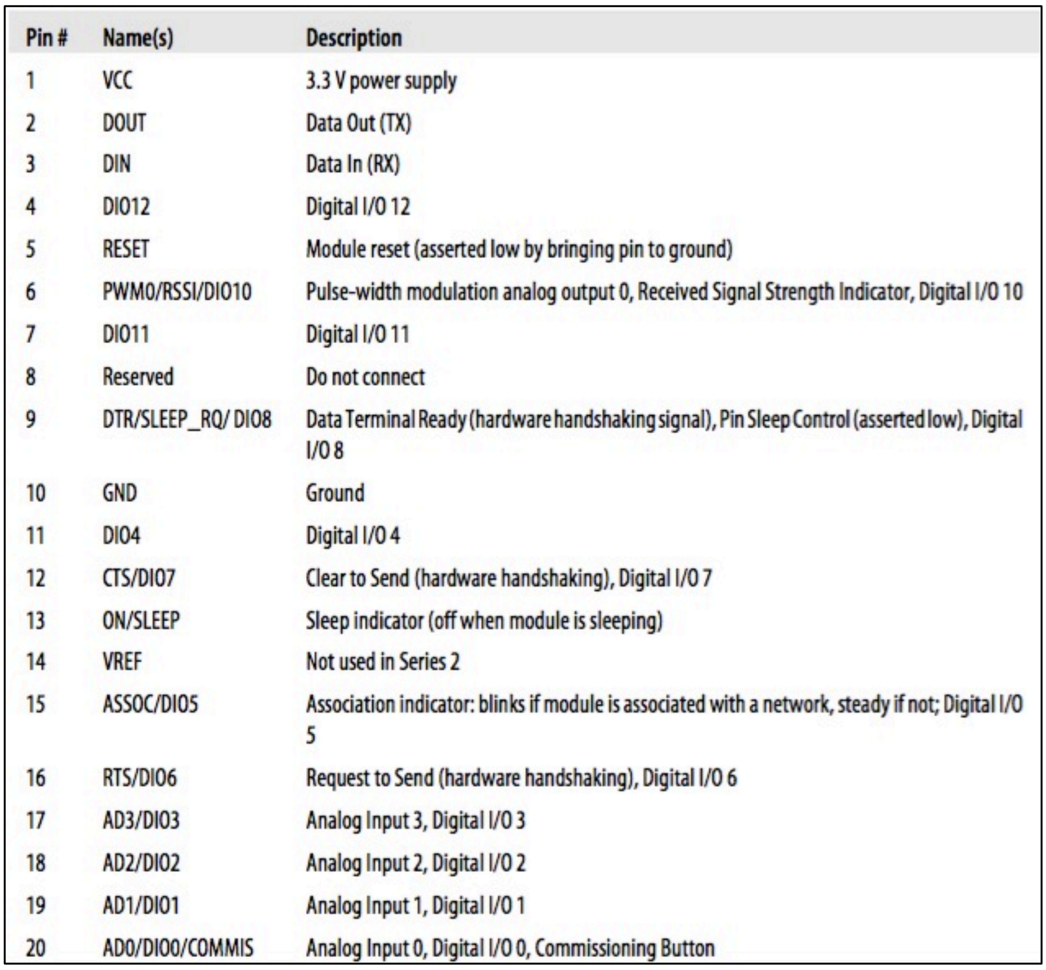
[14] P. R. Lakhe, “Wireless Sensor Network Using ZigBee’’, *IJERA*, pp. 292- 301, March. 30, 2012.

[15] M. Yuksel and E. Erkip, “Multiple-antenna cooperative wireless systems: A diversity-multiplexing tradeoff perspective,” IEEE Trans. Inf. Theory,vol. 53, pp. 3371–3393, Oct. 2007.

[16] Alawieh, B., Zhang, Y., Assi, C. & Mouftah, H. [2009]. Improving spatial reuse in multihop wireless networks-a survey, IEEE Communications Surveys and Tutorials 11(3): 71–91.

APPENDICES

Appendix I XBee series 2 pin description



Pin #	Name(s)	Description
1	VCC	3.3 V power supply
2	DOUT	Data Out (TX)
3	DIN	Data In (RX)
4	DIO12	Digital I/O 12
5	RESET	Module reset (asserted low by bringing pin to ground)
6	PWM0/RSSI/DIO10	Pulse-width modulation analog output 0, Received Signal Strength Indicator, Digital I/O 10
7	DIO11	Digital I/O 11
8	Reserved	Do not connect
9	DTR/SLEEP_RQ/ DIO8	Data Terminal Ready (hardware handshaking signal), Pin Sleep Control (asserted low), Digital I/O 8
10	GND	Ground
11	DIO4	Digital I/O 4
12	CTS/DIO7	Clear to Send (hardware handshaking), Digital I/O 7
13	ON/SLEEP	Sleep indicator (off when module is sleeping)
14	VREF	Not used in Series 2
15	ASSOC/DIO5	Association indicator: blinks if module is associated with a network, steady if not; Digital I/O 5
16	RTS/DIO6	Request to Send (hardware handshaking), Digital I/O 6
17	AD3/DIO3	Analog Input 3, Digital I/O 3
18	AD2/DIO2	Analog Input 2, Digital I/O 2
19	AD1/DIO1	Analog Input 1, Digital I/O 1
20	AD0/DIO0/COMMIS	Analog Input 0, Digital I/O 0, Commissioning Button

Appendix II Arduino Uno code 1 to receive digital data from XBee:

```
int switch =0;
void setup() {
  Serial.begin (9600);
}
void loop() {
 if (Serial.available () > 21) {
     if (Serial.read() ==0x7E) {
       for (int i=0; i<19; i++) {
         byte discard = Serial.read();

       }
       switch = Serial.read();
       Serial.print("Switch is:");
       if (switch ==0) {
         Serial.println ("closed");
       }
       else if (switch ==16) {
         Serial.println ("open");
       }
     }
 }
}
```

Appendix III Arduino Uno Code 2 to show the structure of data:

```
void setup() {
  Serial.begin (9600);
}
void loop() {
if (Serial.available()>21){
  for (int i=0;i<22;i++){
    Serial.print(Serial.read(),HEX);
    Serial.print(",");
  }
  Serial.println('\n');
}
}
```

Appendix IV External microcontroller code 3 to process the data at Router node:

```
int LED = 11;
int analog_sample_value;
float temp_value_router_node;
void setup() {
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
}
void loop() {
if (Serial.available() >= 21) {
if (Serial.read() == 0x7E) {
for (int i = 0; i<18; i++) {
byte discard = Serial.read(); }
int analogHigh = Serial.read();
int analogLow = Serial.read();
analog_sample_value = analogLow + (analogHigh * 256);
}
}
temp_value_router_node = (analog_sample_value *0.117302053);
analogWrite(LED, analog_sample_value/16);
Serial.println(temp_value_router_node);
}
```

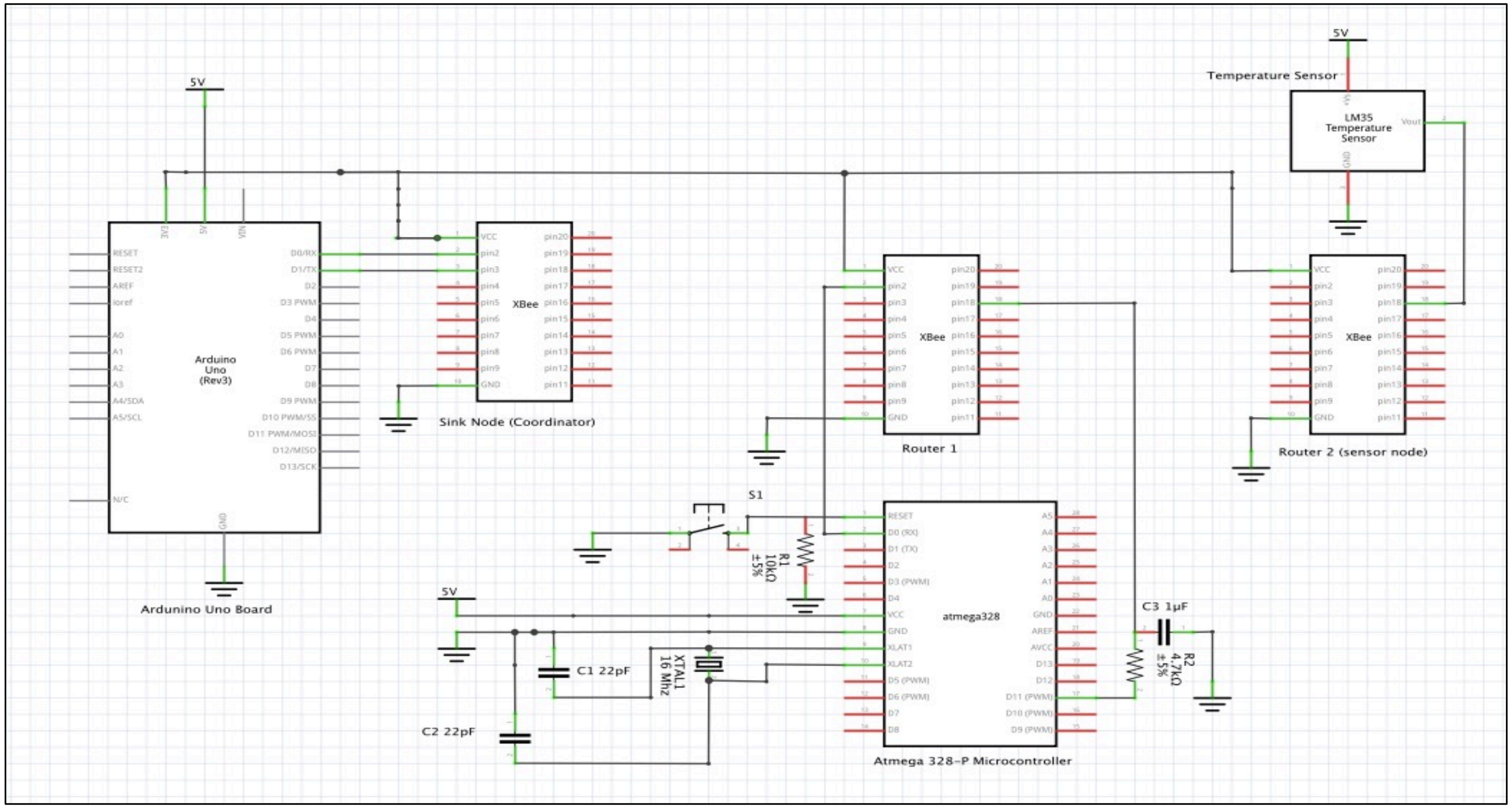
Appendix V Arduino Uno code 4 to display temperature values on serial monitor:

```
int analog_sample_value;
float temp_value_coordinator;
float pressure;
void setup() {
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
}
void loop() {
if (Serial.available() >= 21) {
if (Serial.read() == 0x7E) {
for (int i = 0; i<18; i++) {
byte discard = Serial.read(); }
int analogHigh = Serial.read();
int analogLow = Serial.read();
analog_sample_value = analogLow + (analogHigh * 256);
}
}
// we need to change the equation of the code in order to display pressure values
//pressure=(0.006134*(analog_sample_value-144)) +5;
//Serial.println"Pressure Value";
//Serial.println(pressure);
//equation to display temperature values
temp_value_coordinator = (analog_sample_value *0.117302053);
Serial.println"Room Temperature";
Serial.println(temp_value_coordinator);
}
```

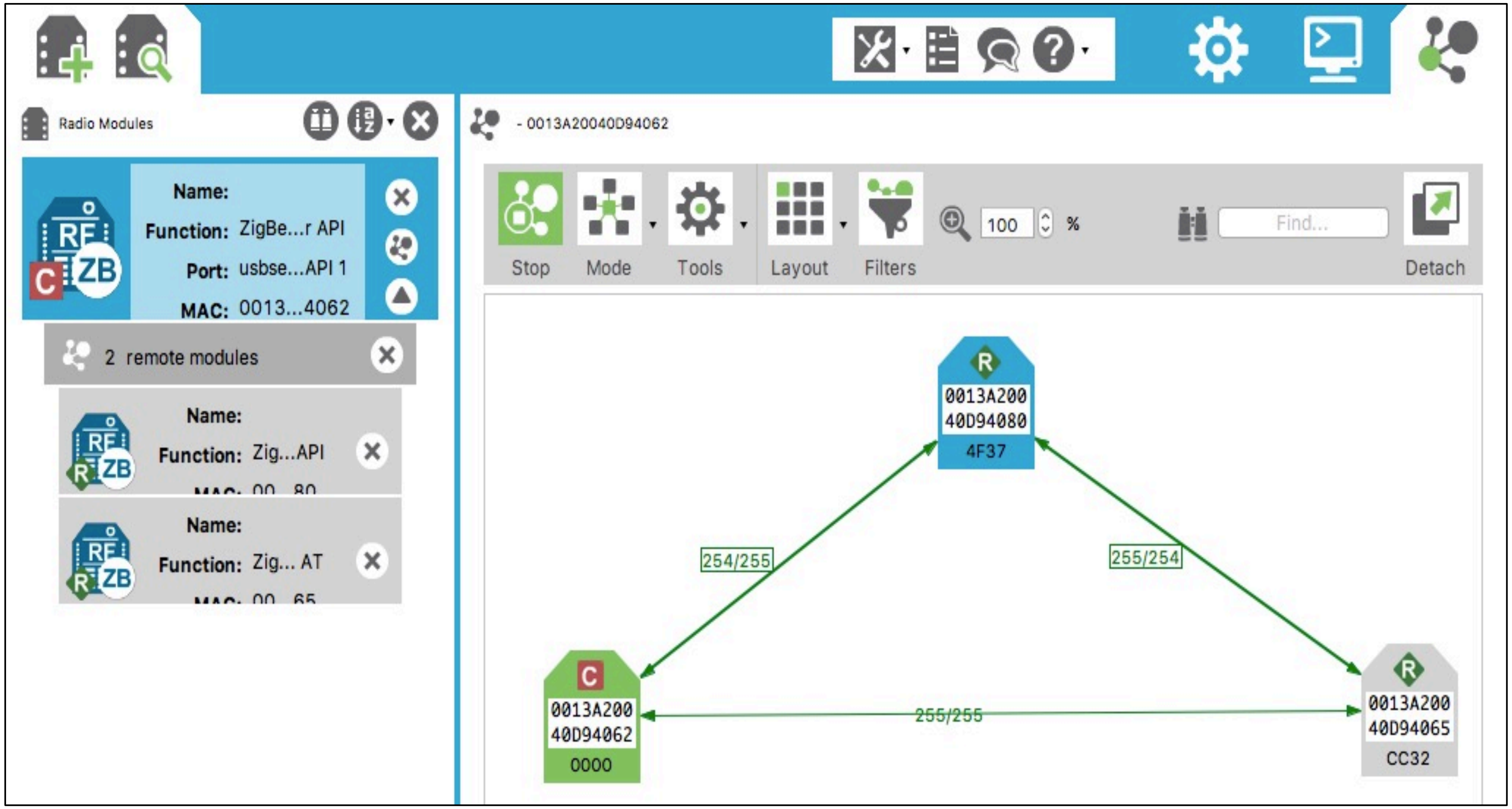
Appendix VI Processing software code:

```
import processing.serial.*;
int lf = 10;
String myString = null;
Serial myPort;
float num;
int yDist;
float pressureP;
PFont font12;
PFont font24;
float[] pressureHistory = new float[100];
void setup() {
  printArray(Serial.list());
  myPort = new Serial(this, Serial.list()[4], 9600);
  myPort.clear();
  size(300, 400);
  font12 = loadFont("Verdana-12.vlw");
  font24 = loadFont("Verdana-24.vlw");
  for(int index = 0; index<100; index++)
     pressureHistory[index] = 0;
}
void draw() {
  while (myPort.available() > 0) {
    myString = myPort.readStringUntil(lf);
    if (myString != null) {
    print(myString);  // Prints String
    num=float(myString);
    println(num);
    }
  }
  myPort.clear();
    background(123);
    colorMode(RGB, 160);
    stroke (0);
    rect (49,19,22,162);
    for (int colorIndex = 0; colorIndex <= 160; colorIndex++)
    {
      stroke(160 - colorIndex, 0, colorIndex);
      line(50, colorIndex + 20, 70, colorIndex + 20);
    }
     stroke(0);
    fill(255,255,255);
    rect(150,80,100,100);
    for (int index = 0; index<100; index++)
    {
    if(index == 99)
    pressureHistory[index] = num*10;
else
   pressureHistory[index] = pressureHistory[index + 1];
   point(150 + index,200 - (pressureHistory[index]));
}
fill(0,0,0);
textFont(font12);
textAlign(RIGHT);
text("71K pal", 45, 25);
text("28K pal", 45, 187);
yDist = int(250 - (250 * (num*0.01*9.5)));
stroke(0);
triangle(75, yDist + 20, 85, yDist + 15, 85, yDist + 25);
//write the pressure values in psi and pascal
fill(0,0,0);
textFont(font24);
textAlign(LEFT);
text(str(int(num)) + " psi", 170, 37);
pressureP = (num*6.89);
text(str(int(pressureP)) + " K pal", 153, 65);
}
```

Appendix VII Multi-Hop WSN circuit connection



Appendix VIII Formation of the Multi-Hop WSN (XCTU software)



Appendix IX Multi-Hop WSN prototype