IOT BASED NETWORK MANAGEMENT PORTAL FOR AUGMENTED REALITY APPLICATIONS

by

Chaithra Radhakrishna, B.E.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
December 2018

Committee Members:

George Koutitas, Chair

Semih Aslan

William Stapleton

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

DEDICATION

TO LORD GANESHA

To my parents, and family for their continuing support and patience; to all my friends for their significant advice and time throughout the completion of my thesis.

ACKNOWLEDGEMENTS

I would like to sincerely thank my thesis advisor, Dr. George Koutitas for his devoted motivation and supervision throughout my career at Texas State University. His guidance helped me complete my thesis successfully.

I would like to take this opportunity to thank X-reality team of Texas state university for their constant support and guidance throughout my thesis. Their suggestion and advice helped me understand the technology and gain more knowledge. I would like to thank members of the committee for their effort and time in reviewing this thesis.

I would like to extend my gratitude towards Prashanth Guha, Karthik Balasubramanya and everyone else who directly or indirectly helped and motivated me with my research.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS v
LIST OF TABLES ix
LIST OF FIGURES x
LIST OF ABBREVIATIONS xii
ABSTRACT xiii
I. INTRODUCTION 1
1.1 Overview 1
1.2 Wireless sensor networks 2
1.3 Augmented Reality in Education 2
1.4 Thesis Outline 3
II. LITERATURE SURVEY 5
2.1 Educational applications for IOT 6
2.2 Network activity with Augmented reality 7
III. ZIGBEE STANDARD 9
3.1 Introduction to Zigbee 9
3.2. Zigbee Architecture 10
3.3 Zigbee Network Characteristics 12
3.4. Zigbee device types 13
3.4.1 Zigbee Coordinator 13
3.4.2 Zigbee router 13
3.4.3 Zigbee End Device 13
3.5 Zigbee Topologies 14
3.5.1 Star Topology 14

3.5.2 Tree Topology 15
3.5.3 Mesh Topology 16

3.6 Zigbee Networking Protocols 16
3.6.1 AODV Routing 17
3.6.2 Many to one routing 18
3.6.3 Source routing 19

3.7 Applications of Zigbee 20
IV. HARDWARE AND SOFTWARE DESCRIPTION 22

4.1 Introduction 22

4.2 Arduino Uno 22

4.3 LM35 Temperature Sensor 23

4.4 Raspberry pi 3 24

4.5 Digi XBee S2D Module 25

4.6 XCTU Application 27

4.7 XBee S2D Operating Modes 27
4.7.1 Transmit mode (AT) 27
4.7.2 Application Programing Interface Operation (API) 28
V. NETWORK MANAGEMENT PORTAL 29

5.1 Routing protocol 29
5.1.1 Short range source routing based LQI protocol 29
5.1.2 Short range RSSI based routing protocol 36

5.2 Data Transmission between Network and Portal 37
5.2.1 Transmit data from network to portal 37
5.2.2 Transmit data from portal to network 40

5.3 Flask web-based Network Management Portal 41
5.3.1 Accessing data from SQL Database 41
5.3.2 Fetching temperature from sensors 41

5.3.3 Fetching RSSI from sensors 41
5.3.4 Choosing the desired Network Topology 42
5.3.5 Mesh Topology 42
5.3.6 Controlling Actuators 43

5.4 Handling Errors 44

5.5 Integration with Augmented Reality 44

VI. EXPERIMENTAL RESULTS 47

6.1 LQI with distance (Indoor Environment) 47

6.2 Faraday cage 48

6.3 LQI when there is obstruction 49

6.4 Teaching Network Topology and Routing to Students 50

VII. CONCLUSION AND FUTUREWORK 53

APPENDIX SECTION 55

REFERENCES 80

LIST OF TABLES

Table **Page**

1. Network Characteristics Comparison 12
2. Arduino Specifications..... 23
3. LM35 Specifications..... 23
4. Raspberry pi specifications..... 25
5. ZigBee range..... 25
6. Pin configuration of S2D Module..... 25
7. Management LQI Neighbor table request: (Cluster ID :0X0031)..... 30
8. Management LQI Response (Cluster ID 8031)..... 30

LIST OF FIGURES

Figure **Page**

1. Internet of Things 2
2. System Architecture 3
3. Zigbee Architecture 10
4. Star Topology 14
5. Tree Topology 15
6. Mesh Topology 16
7. RREQ Packet format 18
8. RREP Packet Format 18
9. Create Source route packet 20
10. Medical Applications 21
11. Traffic monitoring 21
12. Arduino Uno 22
13. Interface of LM35 Temperature sensor with Arduino Uno 24
14. Raspberry pi 3 Module 24
15. Arduino and XBee Interface 26
16. Flowchart for LQI based routing protocol 32
17. Flowchart for Data transfer from Network to Portal 37
18. Sensor network Set-up 38
19. MySQL database 39

20. Web management portal 39

21. Flow chart for data transfer from portal to network 40

22. Star command 42

23. Mesh command 43

24. LED Command 43

25. ngrok to host the data to internet from localhost 45

26. Hologram Script 46

27. Graph of LQI vs distance 47

28. Graph-Faraday cage experiment 48

29. LQI with obstruction(Wall) 49

30. User Interface on Holo-lens 51

31. Temperature and RSSI Visualization 51

32. Star Network Visualization 52

33. Mesh Network Visualization 52

LIST OF ABBREVIATIONS

Abbreviation	Description
XCTU	XBee Configuration and Test Utility
USB	Universal Serial Bus
RF	Radio Frequency
CSMA	Carrier sense multiple access
ZDO	Zigbee Device Object
AODV	Adhoc on demand distance vector
RREQ	Route request
RREP	Route Response
API	Application Program Interface
LQI	Link Quality Indicator
RSSI	Received signal strength Interference
WSN	Wireless sensor Network
IOT	Internet of Things

ABSTRACT

Internet of Things (IoT) along with the evolution of Wireless sensor networks and Augmented Reality technologies has opened many avenues in multiple real-world scenarios that need extensive human interaction. The potent combination of technologies seamlessly amalgamates real and digital world allowing us to improvise in the field of education. Zigbee wireless protocol (802.15.4) has been instrumental in achieving low cost continuous monitoring systems but the ability to demonstrate the network topology depends solely on the distance separating the various nodes. We improvise the existing protocol and thus facilitate the nodes to form mesh topology based on pre-defined parameters but predominantly based on Link Quality Indicator (LQI).

A network management portal is created which gives us the ability to control the type of desired network topology based on the defined protocol and interact with actuators. This further integrates with an Augmented Reality application thus providing us with an ability to control and visualize the topology through the AR interface. A comparative analysis of the node behavior under different external factors is done and conclusion is made regarding the sustainability of the proposed model for short range experimentation in educational applications

I. INTRODUCTION

1.1 Overview

Internet of Things (IoT) has been a trending topic in the recent past in communication and industrial engineering alike. Ever since the evolution of miniature devices, there has been an increase in the number of smart devices which have become part of our day to day lives. From toasters to activity trackers, there is an increasing demand for the data generated by the consumer for a myriad of use cases.

One of the key aspects for why IoT has become a rage is because of the ability of the devices to communicate with each other. The communication can be achieved in a few different ways but since most of the devices are small and are placed in remote locations, they mainly rely on wireless communication. Wireless mesh networks are a common mode of connection and in some rare cases star networks are utilized.

The essence of Internet of things (IoT) is to make life simple and exciting by interconnecting the world of a consumer. Sensors are the backbone of an IoT infrastructure facilitating the interaction between the user and the multiple devices that the user interfaces with such as laptops, smartphones, fitness trackers and robots. At any given point in time one of those devices will keep track of factors such as location and activity of a user to offer the user a personalized world. Effectively the success or failure of IoT depends on how user-centric the solution can get. It must address the challenges posed in real-life scenarios and work on ease of access and use. There is an increase in the amount of data a sensor can process and with an ever-increasing pace of data collection and machine learning, it is totally relevant for the sensors to match up in terms of what they can offer.

Figure 1. Internet of Things[1]

1.2 Wireless sensor networks

Wireless sensor networks (WSNs) are a field of study that has been there for a long time and we see practical applications in our lives daily and how the technology has seamlessly integrated itself into our world. Although the technology has a lot to offer, little work has been done to leverage the same. Sensors form an integral part of WSNs and capture a myriad of information continuously. The data that is collected is then used to design and introduce new features to the consumer. We are utilizing sensors, XBee, Arduino and Raspberry pi to create an IoT network of devices.

1.3 Augmented Reality in Education

Advancement of IoT has increased the expectation of getting real time information from all the smart devices. When the same is combined with Augmented reality the end users can be provided with information from real world as well as described in [2]. By using these technologies, the education sector stands to gain a lot. The learning can be

made easier and more involving for the students along with empowering the instructor with all the tools at their disposal. Augmented reality in education will basically provide more contextually aware and relevant coursework to the students. To render the multimedia content which is context aware, the physical devices through IoT interact with the AR environment.

1.4 Thesis Outline

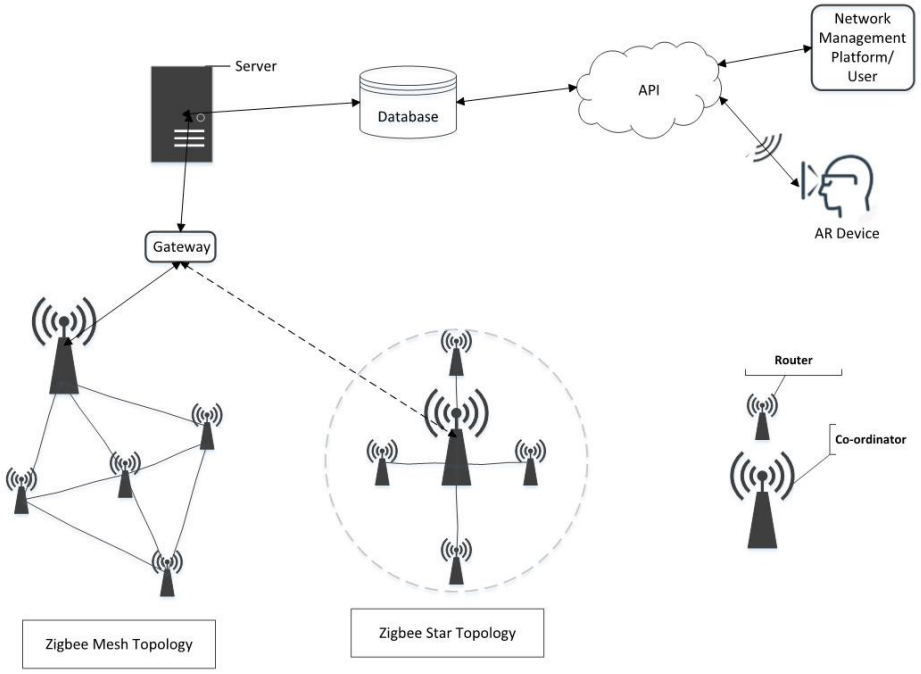


Figure 2. System Architecture

In this research we are going to integrate WSN into an IoT environment to build a platform for data access for practical applications using augmented reality. We create a new LQI based routing that allows us to visualize both star and mesh network when the routers are all placed at a close distance. The work here will be a backend architecture for any user interface to access the data collected by various sensors and gives the user to control the data seen by giving a choice of different options like the type of network

topology, sensor nodes the user wishes to access etc. System architecture is shown in Figure 2

In Chapter 2, literature review based on previous work related to IOT , Wireless sensor networks and Augmented Reality is explained. Chapter 3 is focused on Zigbee standards ,topologies and applications of ZigBee .In Chapter 4, the experimental setup and hardware used in explained in detail along with the interface of hardware's. In Chapter 5, Network routing protocol based on LQI and Interaction between portal and network along with integration with Augmented reality is explained. In Chapter 6, results and simulations are explained. Finally, Chapter 7 summarizes the overall research and future research possibilities are mentioned.

II. LITERATURE SURVEY

There have been several use-cases and needs to implement different routing techniques in a star or mesh topology for a sensor network using XBee nodes. Some of the research that align closely with our study are explored here to understand the body of work.

A comparative study of wireless protocols such as Bluetooth, Wi-Fi, Zigbee and UWB is made as shown in [3]. It lists the various advantages and disadvantages of each protocol. In conclusion, ZigBee finds a niche space in networks dealing with small payloads and scales well due it its minimal energy consumption thus providing longevity. Wireless sensor networks are characterized by small payloads as they are used to perform continuous monitoring and the computations are remote.

Khanh *et al*. explore the benefits of a LQDV routing protocol when tested under a real environment as opposed to a test setup. They propose the LQDV protocol and explain how it differs from the traditional AODV protocol in which it uses link metric instead of the minimum hop count. They also explain a contingency mechanism of waiting for multiple requests while determining the link metric and using it to form the routing table. The results from their experiment suggest that LQDV performs better than AODV in a real environment but could be challenged in the event of a heavy load which is explained in [4].

Payam Porkar *et al.* implement a routing protocol called NbZbr taking into account the number of neighbor nodes and their respective distances and is predominantly based on the theory that the ability to reach a destination is directly proportional to the number of neighboring nodes a particular node might have. They also indicate that the network with better connectivity tends to be energy efficient too. They calculate the cost of the network by considering parameters such as distance, number of nodes and estimate, a

parameter derived from hop count. They conclude their results with the argument that NbZbr is faster and more accurate than other algorithms and also prove to have a low implementation cost and is shown in [5].

Vachirapol et al do a comparative evaluation of ZigBee mesh networks using XBee modules. They base their evaluation on key parameters such as RSSI and packet delay. The experiments were performed in line of sight and non-line of sight cases and evaluated accordingly. They propose this study to be useful in employing mesh networks in building wireless sensor networks of mobile robots [6].

Authors explore a ranging technique based on LQI values in XBee sensor networks described in [7]. Link quality indicator is a metric which measures error values in modulation of successfully received incoming packets which pass the Cyclic redundancy check. Due to the external effects of environmental factors acting on the LQI metric, the authors propose and test a novel approach for LQI ranging. Their method can be divided into three data processing components namely pre-correction, error compensation and regression analysis. From their results they demonstrate that the new ranging mechanism has a high accuracy and can be successfully implemented for localized applications in a wireless sensor network [7].

2.1 Educational applications for IOT

Reza *et al*. Explore a key application using a Zigbee mesh network. They utilize a wireless sensor network to monitor the vitals of a patient and transmit it to a remote station thus achieving continuous monitoring through real-time measurement [8]. Various routing protocols are explored before homing in on ZigBee as it proves to be efficient and cost-effective. An optical heart rate sensor is used to receive heart beat signals which is sampled

and digitized by a slave node before being sent to the master node wirelessly. The received digitized value is transmitted to a PC to be displayed. Although the approach here is uni-dimensional, a lot of potential can be seen through which WSN in combination with ZigBee can be of great use in daily applications which utilize remote monitoring as explained in [8].

Dhiraj *et al*. implement a monitoring system to track various physiological parameters such as Blood Pressure, body temperature, blood oxygen level and fall detection [9]. The sensors report data every minute to a raspberry pi server which can be accessed from anywhere over an internet connection. The values are monitored for any abnormal spike which is communicated to the caretaker through SMS. Their work effectively describes how various sensors form a network to perform a single function with a seamless integration of Arduino and raspberry pi [9]

Pushkar and Sanghamitra as described in [10] propose a cost-effective irrigation system. Here sensors to measure water flow, temperature, soil moisture are used in tandem to produce optimum results. The Realtime values from sensors are compared against a standard set. This process helps the farmer to harvest a maximum quality yield. Data is collected by Arduino and linked to a website.

2.2 Network activity with Augmented reality

Kaufmann explores the advancement of augmented reality technology enabling innovations in learning tools for educational purposes[11]. The research provides greater insight into the immersive learning in virtual environments. The experiments are centered around an application designed for mathematics called Construct3D. The platform allows multiple users to share the same virtual space to create geometrical shapes in a 3D space.

He concludes that AR has much to offer and if it grows at the same rate could be an effective tool in education.

Haramaki and Nishino , in their work stress the importance of topology visualization during real-time network issues. They propose a system to visualize the network topology which assists a network administrator to take stock of a network issue quickly and efficiently and is shown in [12]. The system uses proximity to an access point and vision-based identification as key parameters to identify network equipment. The collected data is then parsed to a head mounted display to provide topology information.

III. ZIGBEE STANDARD

3.1 Introduction to Zigbee

Zigbee is based on 802.15.4 standards and got the formal consent from IEEE (Institute of Electrical and Electronics Engineer) in the year 2003 [13]. Zigbee supports multi-hop mesh networking topology, star and tree topology. Zigbee uses CSMA to improve the reliability in network [14]. This is because it looks for channel to check if it is clear and then begins to transmit data. The inbuilt protocol retries the transmission of packet up to 4 times and if it cannot reach any other node or destination then it informs the source node that the packet delivery was not successful [13].

ZigBee is widely used because of its low cost and self- healing capability in mesh network [13]. If a link in the network fails it automatically looks for other nodes and reforms the network. Zigbee is also reliable in terms of broadcasting the message to all the nodes together. It is available in international standard where the module comes as fully built computer with an MCU and RF module[14]. ZigBee uses AE6-128 for encryption where the packets cannot be understood by the nodes which does not have the encryption key and it can also not inject malicious packets into network. It has total of 16 channels in the 2.4GHz range and is also where each channel is separated by 5MHz [14].

The 2.4GHz in Zigbee is shared with Wi-Fi, Bluetooth and microwave ovens. XBee uses ZigBee protocol and provides a lot of wireless solutions to industries. It has different modes to operate so that the battery can be preserved. Chips on the module are integrated with microcontrollers and other radios [15].

3.2. Zigbee Architecture

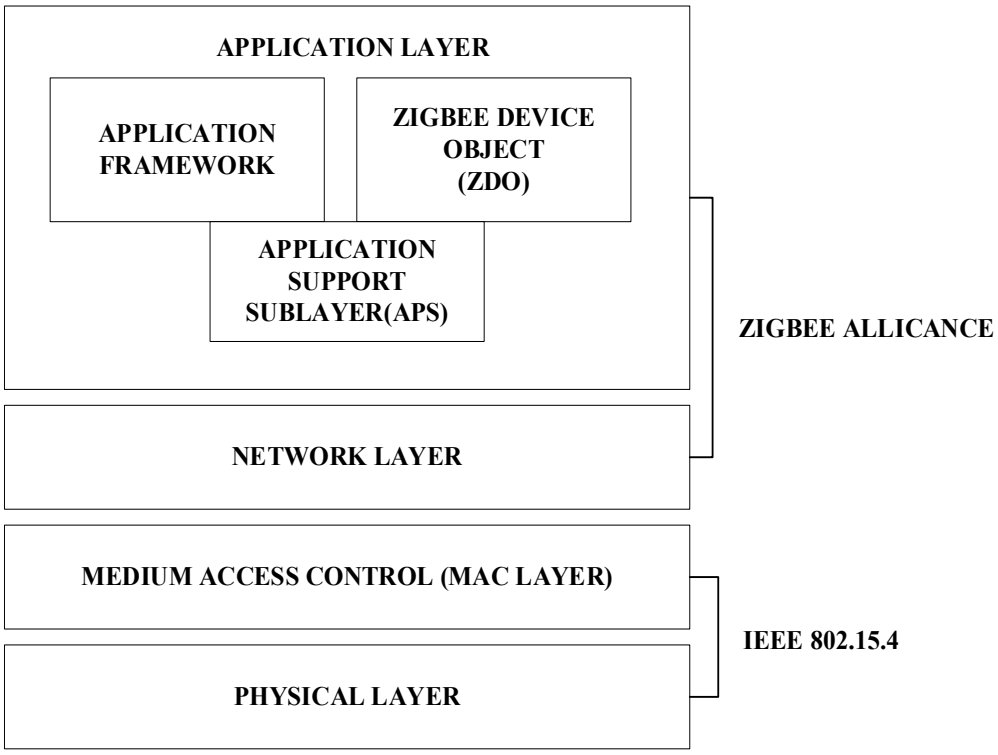


Figure 3. Zigbee Architecture

Zigbee architecture is divided into mainly two sections as shown in Figure 3:

- IEEE 802.15.4 which mainly consists of Physical (PHY) and MAC (Medium Access Control) layers covers the lower layers of the stack
- Zigbee alliance consists of ZDO (Zigbee Device Object), APS (Application support sublayer), Network layer and Security Management covers the upper layers of the stack.

Physical Layer: Physical layer is used for basic transmission and reception. It also performs modulation and demodulation for incoming and outgoing signals. It uses Direct Sequence spread spectrum (DSSS) for simple data transmission by which data is converted to 32bits and each bit is again modulated for Transmission[16]. Different countries choose different frequency band for the operation. For example, European Countries operate at a

frequency of 868MHz and 868.8MHz, North America and South America work on frequency band 915MHz and only 2.4Ghz frequency band is used to operate worldwide[16]. Physical layer is used to measure the Link Quality of the incoming signal and perform energy detection in the current working channel.

Medium Access Control Layer: MAC layer

MAC layer is located in between the physical layer and Network layer. It is used to handle point to point communications in a network, retries and acknowledgements are formed by the MAC Layer [17]. It is used to facilitate network start up and configuration of new devices.

Network Layer:

Routing and traversing multiple hops in a network are mainly supported by the network layer**.** It also facilitates addressing, identifying neighbors, Reception control and joining or leaving a network[13].

Zigbee Device Object (ZDO):

It provides device and service discovery facilities and provides network management facilities [13].

Application Layer (APS):

Addressing objects like profiles, clusters and end points are defined by the application layer [17]. It also responds to any binding requests from the end points and facilitates application services

3.3 Zigbee Network Characteristics

Zigbee is widely used in varied applications. Some of the important characteristics of Zigbee which makes it more popular compared to other wireless technologies like Bluetooth and Wi-Fi are [13]:

- Low data rate
- Low Battery Consumption (A single 9V battery can work for many years)
- Cost Effective.
- Zigbee can form its own network automatically.

Table 1. Network Characteristics Comparison[18]

Data	**Z-Wave**	**Bluetooth**	**ZigBee (802.15.4)**	**Thread (802.15.4)**
Operating Frequency	908.42MHz	2.4GHz	2.4GHz, 868MHz ,915MHz	2.4GHz
Data Rate	9600bits/sec 40Kbits/sec	1Mbps	20kbit/sec (868MHz band) 250kbit/sec (2.4GHz band)	250Kbps
Range	300ft (Outdoor) 80ft (Indoor)	330ft	80ft	100ft
Network Topology	Mesh	Mesh	Star, Tree, Mesh	Mesh
Modulation Techniques	Frequency shift keying (FSK)	Direct Sequence Spread	Direct Sequence Spread Spectrum Modulation (DSSS)	O-QPSK Modulation
Power Consumption	High	Low	Low power	Low power
Nodes	232	32,000	65,000	250-300
Applications	Home and Commercial Automation	Wireless communicatio n between	WSN’S	WSN’S

3.4. Zigbee device types

To build any wireless sensor network using ZigBee, the following three ZigBee devices play an important role. They are:

3.4.1 Zigbee Coordinator

Zigbee Coordinator plays an important role which is responsible for forming a network and managing the network [19]. Coordinator has the information of all the routers and end-devices connected in the network. Coordinator assigns as 16bit PAN ID for all the nodes in the network to identify the device. This PAN ID is same across all the devices in the same network[3]. Coordinator is also responsible to run assist in routing and security services and many other services and hence it to be powered all the time and cannot go to sleep/Idle mode. If the PAN ID of the coordinator is pre-configured to 0, it scans and picks any random available ID for the network. If the PAN ID of the router/ End-device is configured to 0, then it joins any available network

3.4.2 Zigbee router

Zigbee router can join existing network, form routing and can also have child nodes and other routers connected to it. Routers are responsible to forward information in a network if the nodes are far and cannot reach the destination. It can buffer wireless packets if the end devices are in sleep mode [19]. Routers must always be powered on

3.4.3 Zigbee End Device

Zigbee end devices /child nodes can join any existing network[19] ,They can transmit and receiving information but cannot relay messages in a network. It does not have the ability to all other devices to join the network. When there are no data packets to be

sent, they enter sleep mode thus reducing power. It needs to always connect to parent device such as router or coordinator to join the network [13].

3.5 Zigbee Topologies

Zigbee supports three topologies mainly Star, Tree and Mesh Topology

3.5.1 Star Topology

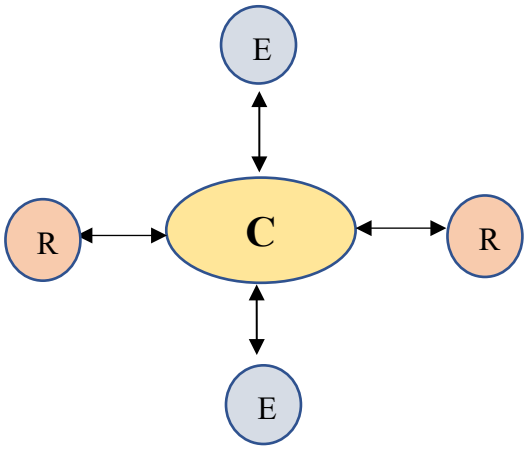


Figure 4. Star Topology

In a star topology, all the routers and end devices are connected to coordinator directly as s. Any node that needs to send a packet to any other node should go through coordinator only. For a network to form a star, all the routers and end devices must be in the same range of the coordinator as described in [19]. In a star topology the routing of messages is not broadcasted, hence any node that is not in the communication range will not receive any packets. Since the coordinator is always responsible and acts as a bridge between the source and destination although the routers are in the same range, this increases network latency and redundancy in the network [3]. Failure of any node in the network is easy to re-configure without disturbing other nodes in the network

3.5.2 Tree Topology

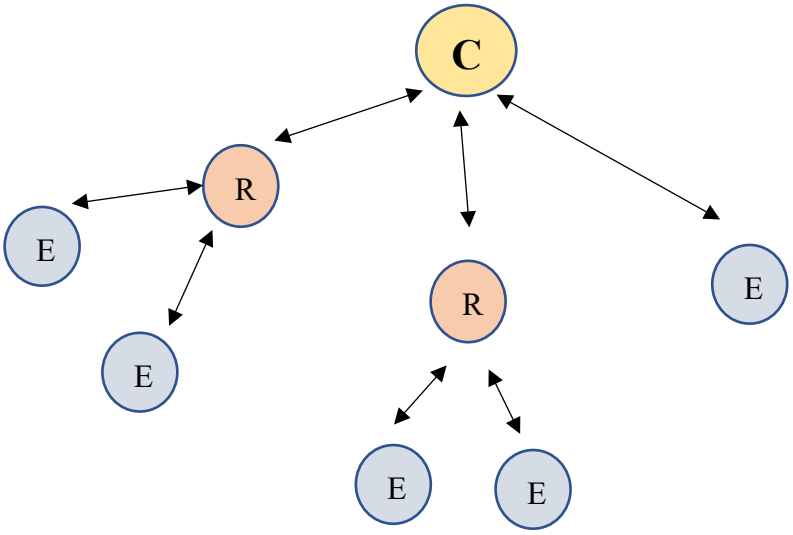


Figure 5. Tree Topology

Tree network consists of coordinator, routers and end-devices as shown in Figure 5. Coordinator is the main hub and is responsible for the forming of network [13]. It can either have routers or end devices connected to it. Routers connected can have multiple child nodes connected it whereas the end devices which act as child nodes cannot have any other child nodes connected because it cannot relay messages. Routers and coordinators can be used as intermediate networks for carrying the data from source to destination in any network. The main disadvantage of a tree network is when a router fails, all the child nodes/end devices are also disconnected from the network and it doesn't have the ability to connect to any other routers in the network [19].

3.5.3 Mesh Topology

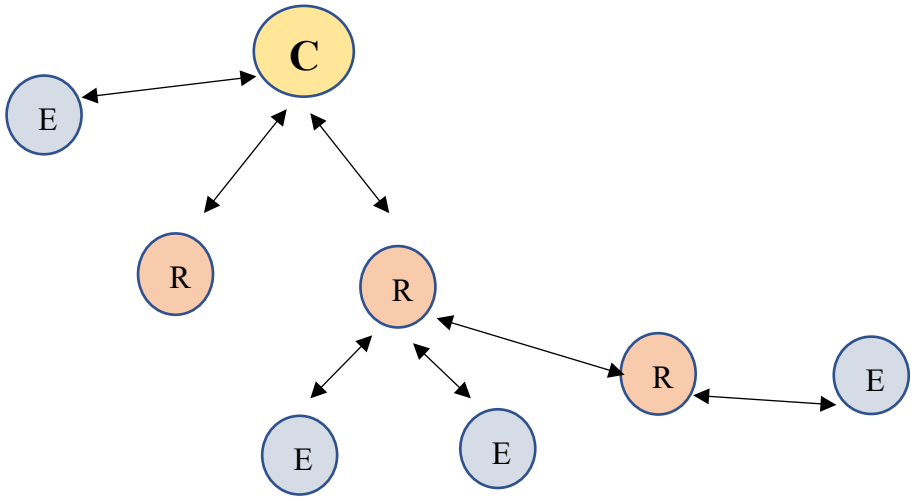


Figure 6. Mesh Topology

Mesh network consists of routers, coordinator and end devices as shown in Figure 6. The mesh topology of ZigBee is the most flexible due to its self-healing capacity. Self-healing is when a router or end device in the network fails it can automatically find other paths and route the packet form source to destination [13]. It is also referred to as multi-hop, since it can traverse to multiple routes to reach the destination. Zigbee mesh has some of the complex routing protocols for self-healing capacity. They are AODV routing, Many to one routing and source routing. Adding or removing a device from the network is very easy which will not disturb the normal functioning of the network.

3.6 Zigbee Networking Protocols

Performance and functioning of any wireless network can be determined by the routing protocol. The path that the network takes to transfer data from one node to another through single or multi-hops is known as routing. The end devices do not have the ability to follow any routing protocol but then just send a unicast transmission to the parent and the parent

makes the decision about the route it needs to take to reach destination. The parent node make decision about the routing path to reach destination[13].

3.6.1 AODV Routing

In AODV routing each node uses routing table entry to store the information of next hop and the destination[13]. Routers and coordinator establish routes using a process called route discovery. The parameters that routers look to determine the next hop are combination of link quality, range, neighboring nodes and end devices. Each parent node will have a routing table record. This routing protocol is used in cases where it does not have more than 40 destinations. [19]

a. Source node sends a route request packet broadcast to all neighboring nodes
b. RREQ packet will have information of source mac address, destination mac address and path cost field (to measure link quality). A sample RREQ packet format is shown in Figure 7
c. All the nodes that receive this will update the path cost field
d. The will create an entry in route discovery table.
e. If any intermediate router other than destination receives this packet, it will forward the packet to destination
f. On receiving the packet, the destination node compares it with the path cost field of the previously received.
g. The destination node sends the route reply packet to the source node. Sample format is shown in
h. Figure 8.
i. For every transmission from one node, route request packet is sent.

If there are large networks and if a route request packet is sent for every packet, then there might be significant Network delays [13].

Source address	Requisition ID	Destination address	Destination Sequence	Source Sequence	Hop count

Figure 7. RREQ Packet format [13]

Source address is the 64-bit mac address of the sending node.

Requisition ID refers to the serial number of the packet, which eventually discards if any duplicates are found

Destination address is the 64-bit mac address of the receiving node

Destination sequence refers to the recent packet from destination which is seen by source

Source sequence is incremented by 1 whenever a packet is sent

Hop count refers to the no of hops the source packet took to reach the destination

RREP is a unicast transmission which traverse in the reverse path

Source address	Destination address	Destination sequence	Hop count	Life time

Figure 8. RREP Packet Format [13]

Destination sequence refers to the duration the route is valid

3.6.2 Many to one routing[2]

Many to one routing can be used along with AODV in a network which has large number of nodes typically more than 40 [19].

a. Instead of all nodes performing the route discovery, the coordinator sends a single many to one broadcast transmissions are sent to all routers and forms reverse route on all devices.

b. The destination address is set to the address of the coordinator.

c. Each node receiving this packet creates a reverse routing entry in the many to one request to create a path back to coordinator.

d. When a node needs to transmit to coordinator and find the many to one request packet, it will transmit the packet ignoring route discovery.

e. We need to send this many to one routing packet at a certain interval to update the reverse routes. To enable many to one routing we need to set the AR (Many to one broadcast time) command to any value other than 0xff.

3.6.3 Source routing

Maximum number of routing table entries in a ZigBee network is 40. To use source routing the device should be updates with API firmware [13]. Source routing packet format is shown in Figure 9

a. Source node sends a route record transmission called Create Source Packet(0X21) to route the packets to the destination.

b. As this packet traverse to multiple hops, the 16bit network address of the hop is appended to RF payload in the packet.

c. The packet will have information of source mac address, destination mac address and 16bit network address of the hops packet traversed.

d. Source node also transmits a Transmit Request packet(0x10) or Explicit Transmit Request packet(0x11) to send the data to destination node

e. The destination node receives this data packet as Receive packet or Explicit RX indicator

f. Destination node also sends a routing information reply or acknowledgment to the source of the route it traversed

g. If any link in the source route is broken then, devices that used that respective route will also be broken.

h. Source routing should be used along with many to one routing, so that periodic many to one request will be sent to refresh the route.

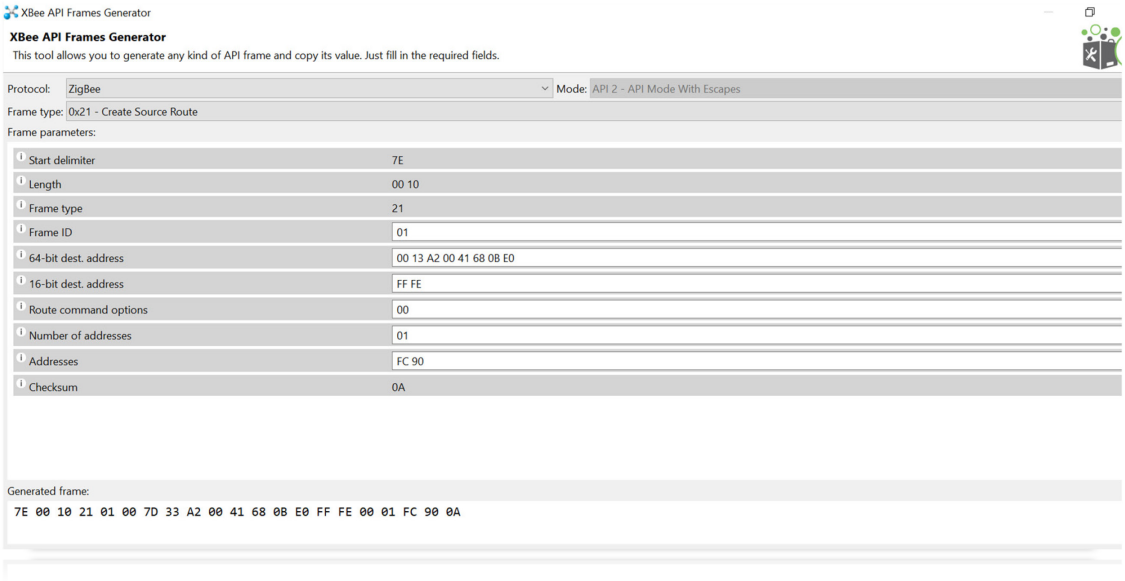


Figure 9. Create Source route packet

3.7 Applications of Zigbee

Some of the applications of ZigBee include:

- Home Automation-It can be used to control some of the appliances at home like heater , cooler and security surveillance [20].
- Environment control-It can be used in forest , climate and habitat monitoring[21]
- Medical applications - It can be used to monitor patients' vitals remotely. An example is shown in Figure 10

- Industrial automation[20] – It is efficiently used in manufacturing industries for the control of equipment's.
- Traffic control – ZigBee can act as good collision detection device as show in Figure 11 when two devices on different vehicles come in range within each other, signals can be sent out of what the driver ahead or behind plans to do.

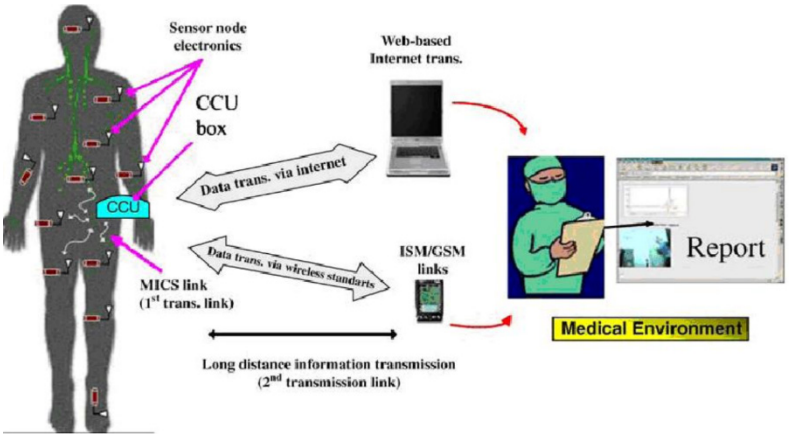


Figure 10. Medical Applications [22]

Figure 11.Traffic monitoring[23]

IV. HARDWARE AND SOFTWARE DESCRIPTION

4.1 Introduction

For this research we are using Arduino Uno, Digi pro S2D XBee Module, Raspberry pi, LM-35 Temperature sensor, XCTU software to configure XBee to test the working of modules, and PyCharm to run the server program. The specifications of the hardware and software are mentioned below:

4.2 Arduino Uno

The Arduino Uno board is a portable microcontroller based on Atmega 328. It can be programmed with Arduino software and can be used with any computer when plugged in with Universal serial bus (USB) cable[24]. The open source nature and ease of use makes it a high demand device. Arduino Uno uses Atmega16u2 programmed as a USB to serial driver chip compared to other boards which use FTD1 USB to serial driver[24]. Transmit (TX) and Receive pins (RX) is used for serial data transmission and reception. It uses USB COM driver's firmware and does not require external drivers[24]. The hardware specifications for Arduino is shown in Table 2.

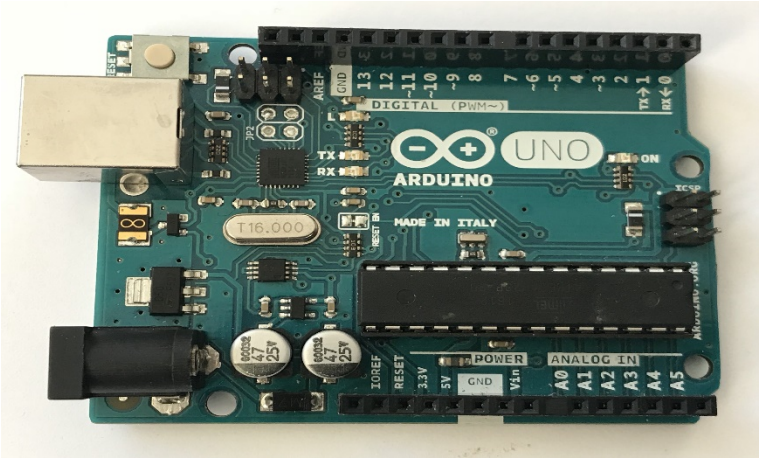


Figure 12. Arduino Uno

Table 2. Arduino Specifications [24]

Board	Arduino Uno
Processor	ATmegs328
Operating Voltage	5V
CPU Speed	16MHz
Analog Input/output	6/0
Digital Input/output	14/6
SSRAM [kb]	2
Flash [kb]	32
USB	Regular
UART	1

4.3 LM35 Temperature Sensor

LM 35 is a precision temperature scale and which has temperature value almost linearly proportional to the centigrade value. It converts the temperature to proportional output voltage [25].Due to high output voltage generated by the sensor, it does not require the output voltage to be amplified. It has a scale factor 0.1V/deg Celsius [25].The sensor has two inputs and one output. The output of the sensor is connected to the analog input of Arduino. One input is connected to +5V of Arduino Uno and another input to ground with specifications shown in Table 3.

Table 3. LM35 Specifications[25]

Output Voltage	-55℃ (−1V) to + 150℃(6V)
Supply voltage	+5V
Precision	±1℃
Sensitivity	10mV/℃

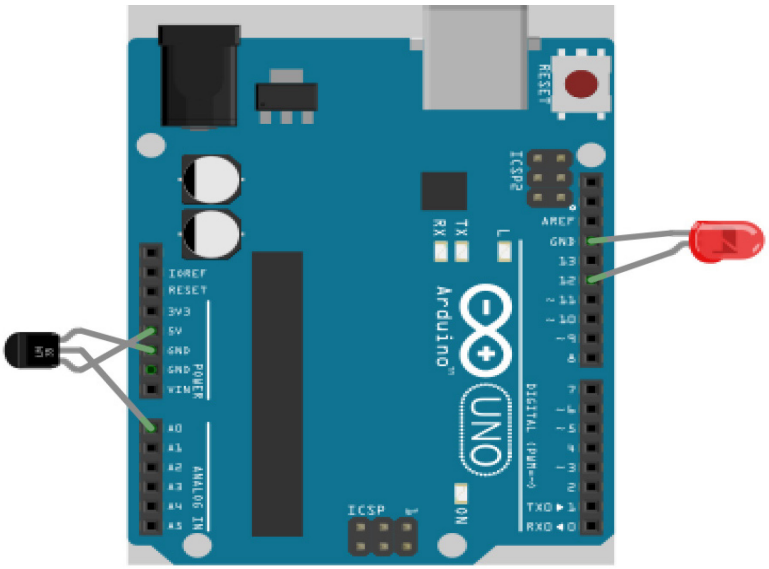


Figure 13. Interface of LM35 Temperature sensor with Arduino Uno

4.4 Raspberry pi 3

Raspberry pi is a credit card sized computer which is based on Linux operating system and can run applications like a normal PC. This low cost enough power, memory and a storage to gather all the data in a single SD card [26]. Raspberry pi is used as a gateway between IOT network and database. It is programmed in python in Raspberry pi which in turn acts as a server.

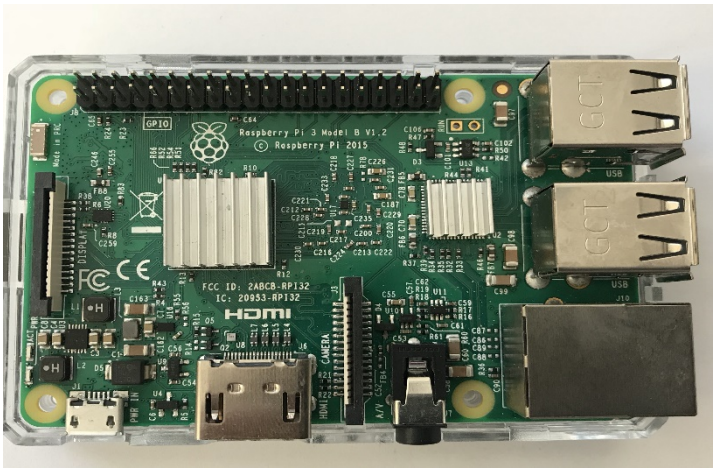


Figure 14. Raspberry pi 3 Module

Table 4. Raspberry pi specifications[26]

CPU	4x ARM Cortex- A53, 1.2GHz
RAM	900MHz – 1GB LPDDR2
STORAGE	Micro SD
GPIO	40 pin headers
SoC	Broadcom BCM2837
Networking	2.4GHz 802.11 wireless, 10/100 Ethernet

4.5 Digi XBee S2D Module

The XBee shield is interfaced with Arduino Uno for wireless transmission of information using the ZigBee protocol. It is loaded with ZigBee firmware which is used to support low cost sensor networks. It does not require too much power and can rely on a single 9V battery for many days as described in [13]. This module can be used in a command mode or as a USB/Serial replacement. Key advantage of Zigbee is supports mesh networks. The range can vary based on external interferences. The range is predominantly dependent on transmitting power [13]. In this research we are focusing on e1liminating the range and form mesh for educational purposes to be able to visualize real-time data.

Table 5. ZigBee range [19]

Indoor	Up to 60 m (200ft)
Outdoor (RF Line of sight range)	Up TO 1200m (4000ft)

Table 6. Pin configuration of S2D Module [19]

Pin	Name	Description
1	GND	Ground
2	VCC	Power supply

Table 6 . Continued		
3	DOUT/DIO3	Data out
4	DIN/DIO14	Data in
5	DIO12	GPIO
6	RESET	Reset
7	RSSI/PWM	Receive signal strength
8	PWM1	Pulse width modulator
9	Reserved	No connection

Arduino Uno is connected to XBee shield to transmit the information from wirelessly between routers. Program to design protocol and form routing is embedded inside Arduino IDE software. The TX pin of Arduino is connected to DIN of XBee and RX pin is connected to DOUT of XBee. Power supply of 3.3V is given from Arduino to XBee. The connection is as shown in Figure 15.

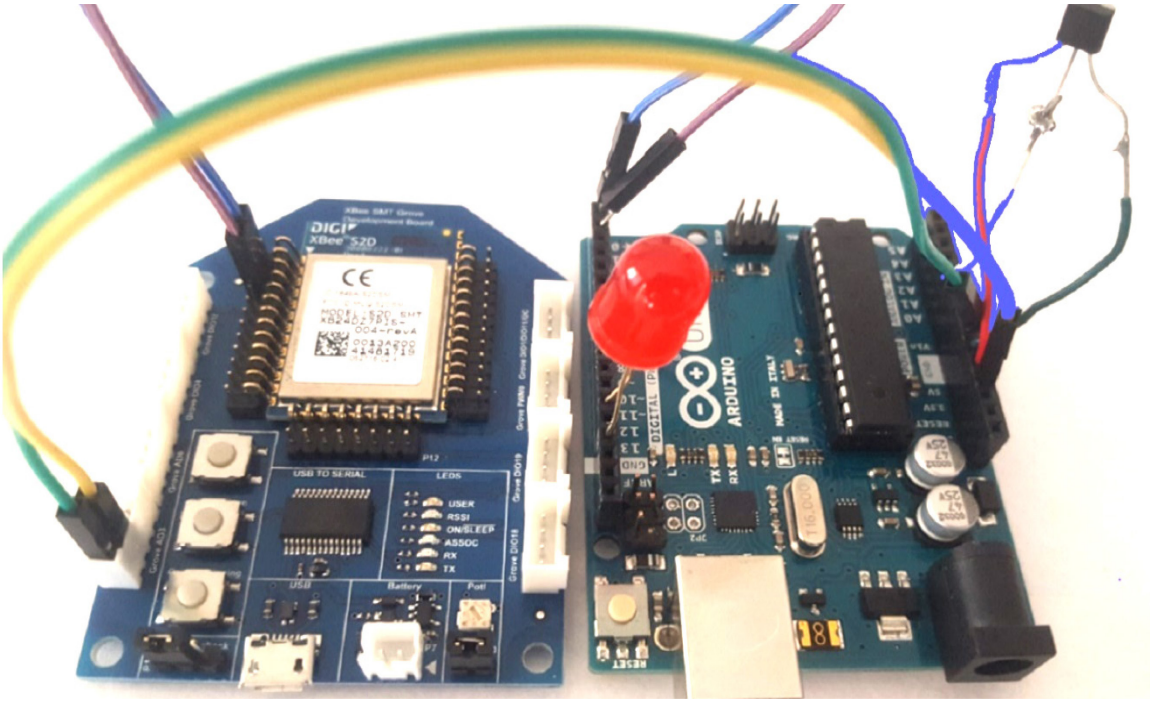


Figure 15. Arduino and XBee Interface

4.6 XCTU Application

XCTU stands for XBee configuration and test utility [19] which provides a graphical interface to user to interact with the radio modules. It has built in tools and firmware which helps to test inbuilt features. It also helps user to read and explore firmware in the device.

This is an easy way to check if the module is working properly and upgrade the latest firmware from the software. If there is any problem with the module, it can be fixed using XBee recovery from the software

4.7 XBee S2D Operating Modes

XBee series2 S2D models mainly operate in two modes such as Transmit (AT mode), Application program Interface (API)

4.7.1 Transmit mode (AT)

Transparent (AT)mode is the simplest point to point wireless communication. Destination node receiving this data is sent serially without making any changes.

In transparent mode (AT), the transmitting node radio sends data to the destination node by setting the DH and DL pin to the address of the destination. Transparent mode is used in a simple communication between two nodes. When several XBee needs to communicate at the same time , the command mode would be needed to activate every time a new destination address needs to be entered. When a device is in AT mode , and it receives multiple messages from many routers, the source of the device. The source needs to send extra information of all the routers in the packet information[12]. When a module is working in AT mode it uses Transmit Request packet to send the data to remote node.

Command mode is the mode entered to modify the RF parameters.

4.7.2 Application Programing Interface Operation (API)

API mode can be used when messages need to be transmitted to multiple devices. In API mode instead of changing addresses in the command mode , the destination address can be changed in the payload packet [19] . The sender will retry sending the packet for up to 4 times if there is no acknowledgment received. API mode uses Explicit transmit request frame to send the information from one node to another[13][19].

There are two modes under API :

API =1 mode without escape characters = It depends on start delimiter and length of the bytes to differentiate between frames. The frame structure consists of Start delimiter(0X7E), Length-(1-2) (most and least significant byte) , Frame data(4-n) – API frame structure and checksum 1 byte as described in[19][13]

API =2 mode with escape characters = It is used to get reliable output noisy conditions [13] .When the output has 0X7D to escape the characters in a frame and XoRed with 0X20 as described in[19][13].

V. NETWORK MANAGEMENT PORTAL

5.1 Routing protocol

In our experiment we try to build XBee mesh routing protocol for single hop and multi-hop networks when the nodes are placed very close to each other. Each node consists of a single Arduino Uno with an TX/RX pin and an XBee controller. There are total of 5 routers and a single coordinator to visualize star and mesh networks. The communication between Arduino and XBee are established using the serial terminals. The setup of XBee modules can be implemented using the XCTU software. XCTU software also allows us to monitor transmission and reception of data. In this chapter we focus mainly on source routing based LQI and RSSI protocol and the interaction between sensor network and network management portal

5.1.1 Short range source routing based LQI protocol (Mesh Network)

LQI is metric which measures error values in modulation of successfully received incoming packets. It is calculated by the physical layer and gives information of the link quality to all the above layers each time when a data is received. LQI is measured in the range of 0-255 where 255 represents the highest quality of link with reduced packet loss rate and 0 being the least[27]. It can be measured by broadcasting an Explicit ZDO command to all the neighboring nodes and the response is stored in a table entry for all the nodes. To measure the link quality in ZigBee, ZDO commands use explicit transmit command frame (0x11)[27] . Any byte commands in the ZDO request must use little endian byte order so that the remote node is able to receive the right command in the desired format [13]. For the remote node to receive ZDO request, API output(A0) must be set to true. The ZDO packet used to send the LQI packet request and receive the response is

shown in Table 7 and Table 8:

Table 7. Management LQI Neighbor table request: (Cluster ID :0X0031)[28]

Start delimiter	7E
Length	001A
Frame type	11
Frame ID	01
64-bit destination address	0013a20041680ca1
16-bit destination address	FFFE
Source endpoint	00
Destination end-point	00
Cluster id	0031
Profile ID	0000

Table 8. Management LQI Response (Cluster ID 8031)[28]

Start delimiter	7E
Length	0017
Frame type (Explicit RX	91
64-bit destination address	0013a20041680ca1
16-bit destination address	FFFE
Neighboring device	Coordinator
Receiver on when idle	0x0 – off
Relationship	0X0 – Neighbor is parent
Cluster id	8031
Reserved	0
Depth	00 indicated neighbor is
LQI	LQI value

- ***Start Delimiter –*** *Start Delimiter is used to identify the start of a data packet that differentiates from other packets sent*
- ***Length -*** *Length defines the total length of the packet excluding checksum*
- ***Frame type*** – Refers to the packet request used to send the packet. For a zdo packet to be sent we need to use the explicit transmit request(0X91) [28]
- ***64-bit destination address*** – For a unicast transmission 64-bit address of the destination is set , and for broadcast transmissions it is set to 0x000000000000FFFF
- ***16-bit address*** – It is said to FFFE if the address is unknown or for broadcast transmissions.[28]
- ***Source and destination endpoint*** *– Set to 0 for ZDO endpoint[28]*
- ***Cluster id*** – Is set to the respective ZDO command sent(here it is 0031 and 8031)
- ***Neighboring device*** – 0x0 – ZigBee coordinator, 0x1 – ZigBee router ,0x2 – ZigBee end device ,0x3 – Unknown[28]
- ***Relationship –*** *Relationship defines if the node is a parent device or a child device[28**]***
- ***Receiver*** –set to 0
- ***Receiver on when idle*** – defines if the node is accepting incoming connections[28].
- ***Depth*** – 00 indicates the device is ZigBee coordinator [28]
- ***LQI*** – Estimated link quality of neighboring device[28]

The flowchart describing on how the data is fetched and transmitted to intermediate routers or destination is as shown in Figure 16 .

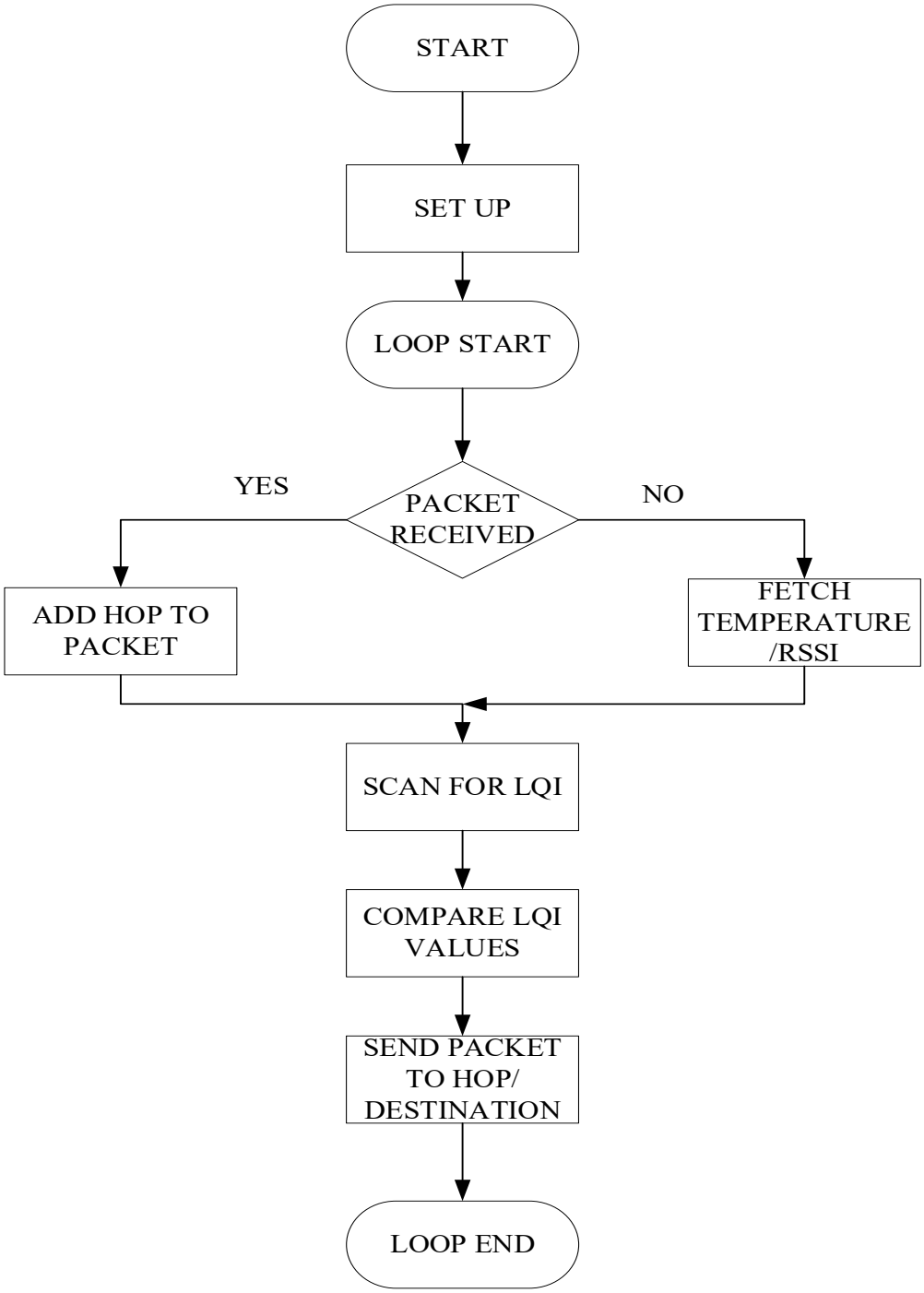


Figure 16. Flowchart for LQI based routing protocol

Creation of packet and protocol design is embedded inside the Arduino Uno and is in serial with XBee routers and is wirelessly transmitted to other routers using Custom LQI based source routing protocol

1. **Set up includes:**

- BAUD RATE= 9600- default Baud rate used by XBee modules which initializes the serial communication with XBee
- Pin configurations to read the output of temperature sensor and LED. Temperature is read at Analog pin A0 and LED at pin 12 of Arduino.
- Destination/remote address is set to the 64bit address of the coordinator.
- Fetch Mac address and Network address- Mac address and network address are fetched as soon as the serial connection is established to identify the router connected. Each router will have a unique 64bit MAC address and 16-bit network address. MAC and Network address is fetched using AT commands ATSH, ATSL, ATMY.

  ATSH: ATSL is 64bit high and low mac address of the node

  ATMY is16-bit Network address

Transmission and reception of data will not begin unless the setup is completed.

2. **Loop**

- Fetch Temperature from the sensor
- Fetch RSSI (Received signal strength Interference) using the ATDB command. RSSI is the signal strength of the last received packet.
- Scans network for LQI from the neighboring nodes with timeout-1sec

- Creates a payload packet which includes sensor information such as temperature, RSSI and LQI values.

3. **Create ZDO request packet:**

- Zigbee explicit Transmit request is used to create the ZDO request packet to all the neighboring nodes.
- Every node sends a response back to the sender every time the request packet is received with the LQI values appended

4. **Handle ZDO response:**

- The LQI response is handled by ZD0 response packet where the last value defines the link quality with the respective router, the ZDO response packet is handled which contains information of all LQI of all the neighboring routers in the network.
- Response is received and stored in the device table for all the neighboring devices.

5. **Compare LQI values:**

- Each router will have a table of all the LQI values with neighboring nodes in the network, where first five values represent the LQI with the neighboring routers and the last value represent the LQI with the coordinator.
- Each node compares the LQI values in the list and determines the next available hop.

For example, if R1 has LQI list in the following format, it will compare the LQI values of all the routers except the coordinator

LQI [R1] = [0, 254, 253, 253, 251, 252), In this case R1 send the packet to R2

6. **Receive packet:**

- Source node forwards the data packet to the next hop based on the LQI.
- On receiving the packet, the receiving hop node discards the LQI values of the source node but then LQI values of all the other nodes in network except the source node is compared and determines the next hop until it reaches coordinator.
- Since there are continuous fluctuations in the LQI values, the next hop is determined by comparing the LQI values and sets the max LQI value to be max and forwards the packet only if the new hop is greater than (max+15) in comparison with the previous hop

7. **Add Hop Packet**.

- The intermediate node other than the coordinator receiving the packet from the source node, appends its 16 -bit network address to the RF data packet to , so that when the destination node receives the packet it will have information of all the hops the packet traversed.

8. **Packets received by destination node**

- A python program is written in the server that is Raspberry pi to handle and read all the packet received by the coordinator. The coordinator is the central hub and has all the information of data packets from all the routers. The data decoded from the coordinator is appended to MySQL database

5.1.2 Short range RSSI based routing protocol

RSSI in ZigBee is the received signal strength of the last packet which is dependent on the power present in the signal[13]. RSSI is measured using ATDB command and can also be measured by reading the output at pin 7. The source sends a packet to another node and the receiving node responds with ACK, ZigBee used the ACK packet to determine the signal strength of the last packet. If a packet has traversed multiple hops, then DB value has no information about previous hops but just the packet received.

RSSI is measured in dBm and usually represented in negative values. If RSSI has a greater negative value, the signal is weaker and lesser negative value indicates stronger signal. The protocol differs from existing ZigBee protocol because it considers just the RSSI for determining the distance and does not consider range into account.

- Source node sends a packet to all the neighboring nodes.
- Neighboring nodes respond with the ACK packet on receiving the data packet.
- Source nodes calculates the RSSI with that node after receiving the packet
- RSSI values of all the nodes are compared, with the lower negative value chosen to be next hop.

Since RSSI values are continuously fluctuating and mainly dependent on power in the Antenna [13], it would reflect high power based on interference and noise. Based on this analysis, it was found that LQI is better estimation for distance compared to RSSI. Since LQI changes significantly only with distance , an experiment was conducted to use faraday cage to reduce the signal strength when the routers are close to each other

5.2 Data Transmission between Network and Portal

5.2.1 Transmit data from network to portal

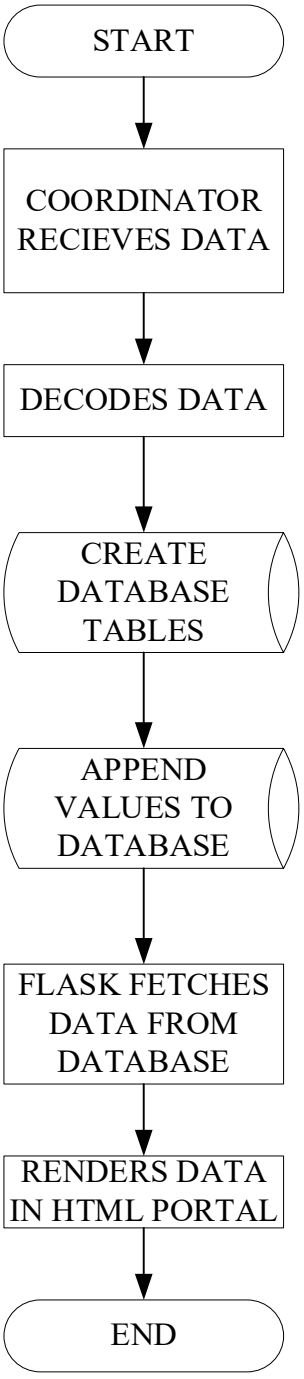


Figure 17. Flowchart for Data transfer from Network to Portal

1. **Coordinator receives data**

   - A server program is written in python to decode the received frame at the respective PORT with Baud rate set to 9600

2. **Decode received frame**

   The received RF data is in the format

   Rf data format = Temperature/RSSI/LQI values

   Coordinator identifies the source mac address to determine the data from respective routers before updating the database

   Once the data packets are received, router status is set to1 if the coordinator receives packet from that mac address otherwise the router status is set to 0

3. **Creates SQL tables** for each sensor and appends the packet information received from the sensors

4. **Flask web-management portal** that hosts the information from data base to portal

Figure 18 shows the set-up of routers and Figure 19 and Figure 20 shows the sample data updated in database which in turn is hosted on website.

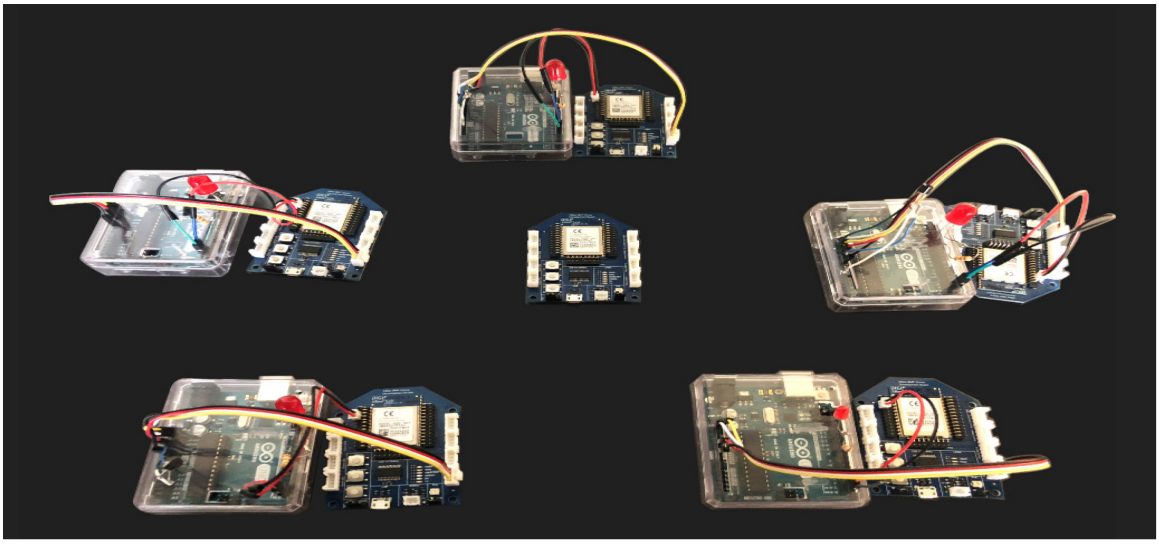
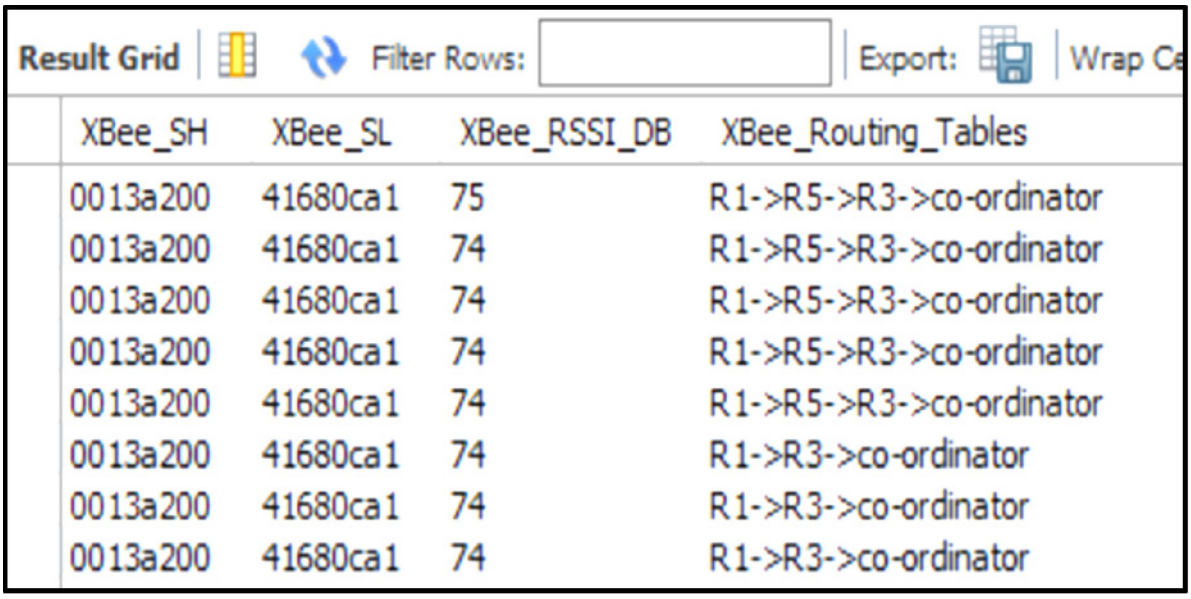


Figure 18. Sensor network Set-up



Result Grid | Filter Rows: | Export: | Wrap Ce

XBee_SH	XBee_SL	XBee_RSSI_DB	XBee_Routing_Tables
0013a200	41680ca1	75	R1->R5->R3->co-ordinator
0013a200	41680ca1	74	R1->R5->R3->co-ordinator
0013a200	41680ca1	74	R1->R5->R3->co-ordinator
0013a200	41680ca1	74	R1->R5->R3->co-ordinator
0013a200	41680ca1	74	R1->R5->R3->co-ordinator
0013a200	41680ca1	74	R1->R3->co-ordinator
0013a200	41680ca1	74	R1->R3->co-ordinator
0013a200	41680ca1	74	R1->R3->co-ordinator

Figure 19. MySQL database

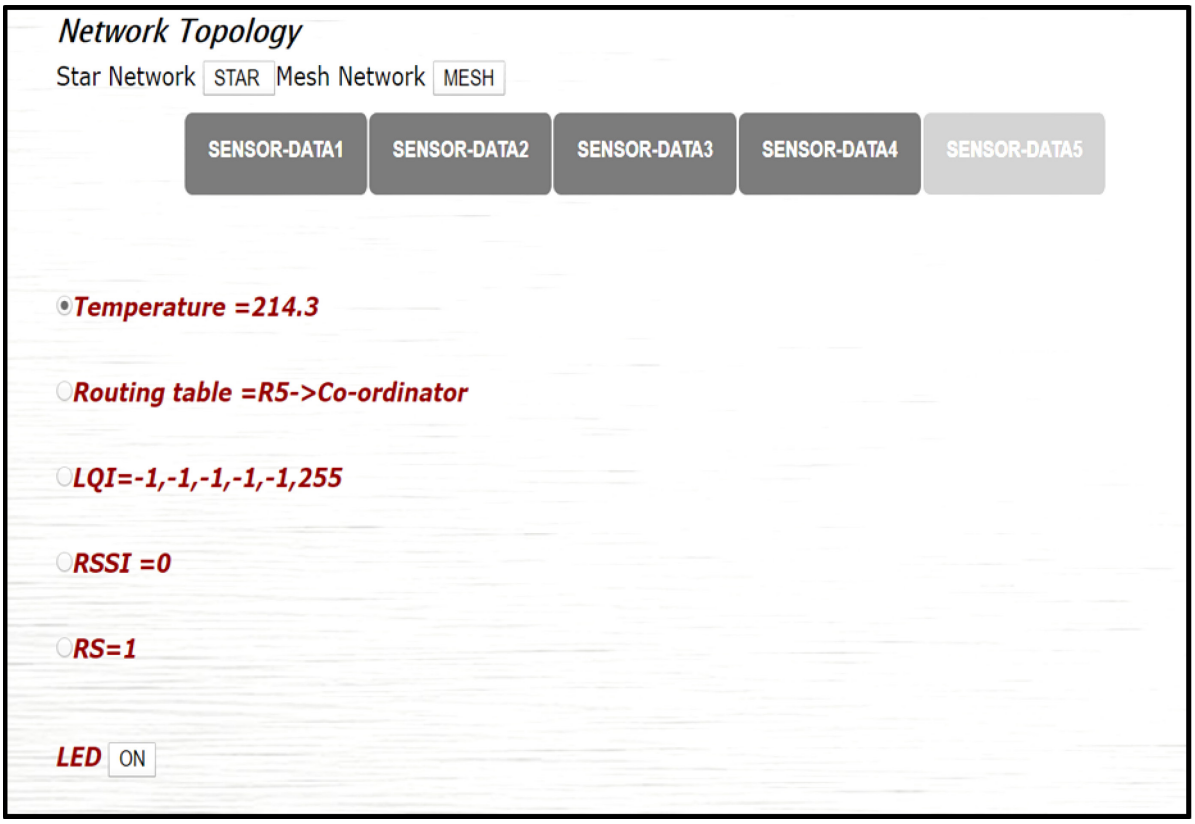


Figure 20. Web management portal

5.2.2 Transmit data from portal to network

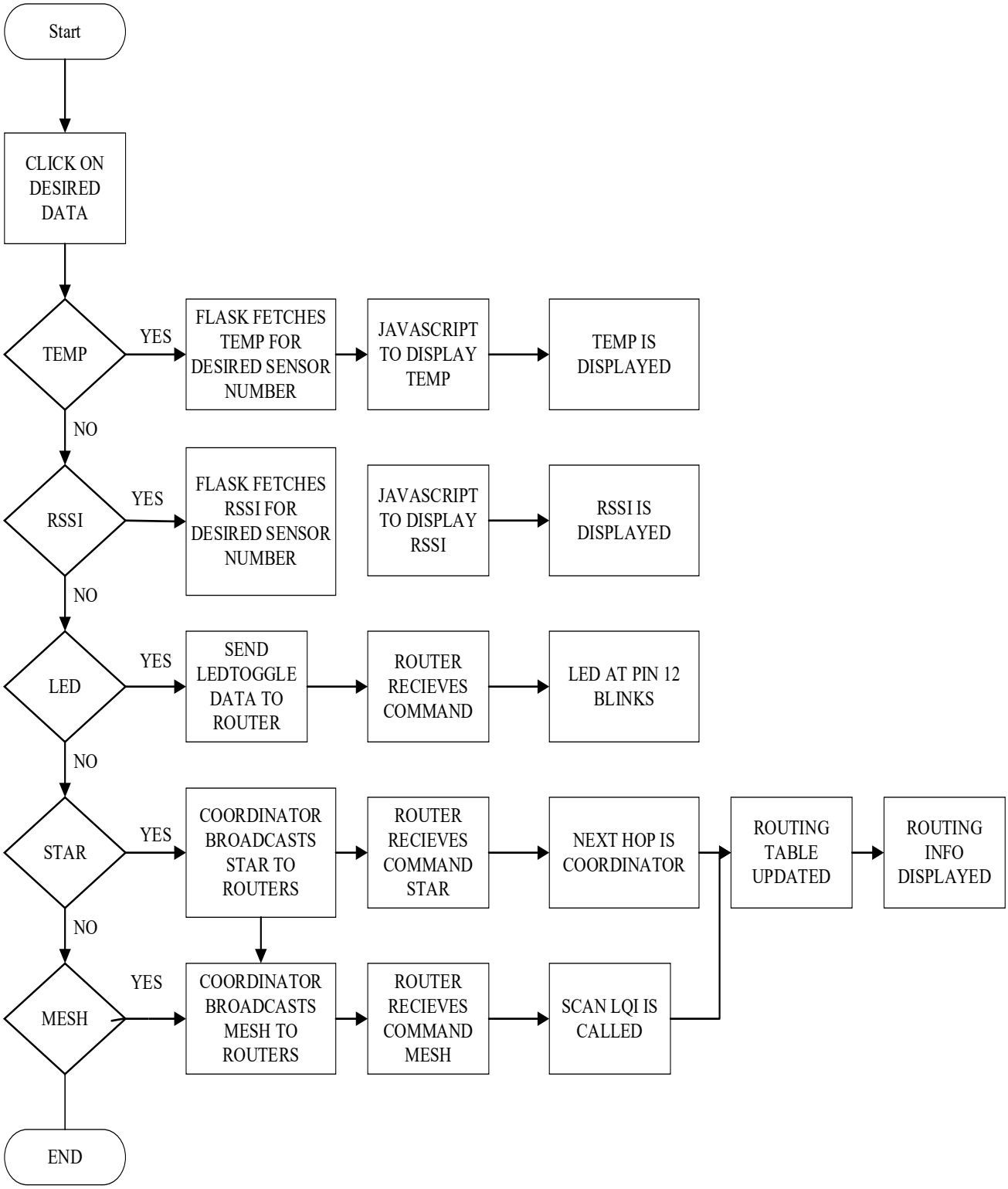


Figure 21. Flow chart for data transfer from portal to network

5.3 Flask web-based Network Management Portal

Flask is a micro-web framework written in python based on Jinja

5.3.1 Accessing data from SQL Database

- Server program running on coordinator shown in Figure 21 decodes and updates the data in the database as soon as packet is received
- Flask is web-based network management portal that fetches the data from database and renders data on the HTML page.

5.3.2 Fetching temperature from sensors

- User click on the temperature for the respective sensor
- Flask fetches the data from database through SQL query written.

"SELECT temperature FROM sensor_data" + *sensor_number* + ***" order by sensor_time desc limit 1"***

- Where sensor number represents the sensors connected routers 1 – 5.
- Temperature is displayed on the portal in Fahrenheit

5.3.3 Fetching RSSI from sensors

RSSI is the received signal strength of the last received packet.

- User click on the RSSI for the respective sensor
- Flask fetches the data from database through SQL query written.:

"SELECT Xbee_RSSI FROM sensor_data" + sensor_number + " order by sensor_time desc limit 1"

- Where sensor number represents the sensors connected to Router1 – Router5

- RSSI is displayed in DBm

5.3.4 Choosing the desired Network Topology

Star Network

- User clicks on star topology in the portal
- Flask identifies the form action /STAR
- Coordinator broadcasts the information to all the routers with respect to 64bit addresses

for index, mac_address in enumerate (SensorSHSLAddress)

- Routers receive the command /STAR on Arduino. Sample code is shown in Figure 22

```
if (!strcmp(cmd, "STAR")) {
 star = true;
```

Figure 22. Star command

- Calls the function update routing table where if star = true ignore the LQI routing protocol and directly connect to coordinator
- Next hop is updated to be coordinator address/remote address
- Data is sent to coordinator
- Routing table is updated and hosted on website

5.3.5 Mesh Topology

- User clicks on Mesh topology in the portal
- Flask identifies the form action /MESH

- Coordinator broadcasts the information to all the routers with respect to 64bit addresses

for index, mac_address in enumerate (SensorSHSLAddress)

- Routers receive the command /MESH on Arduino.

```
if(!strcmp(cmd, "MESH")) {
  star = false;
}
```

Figure 23. Mesh command

- Calls the LQI function
- Updates routing information and sends packet to hop
- Adds 16 -bit network address to the packet
- Data is sent to coordinator, flask retrieves the packet and hosts information on website

5.3.6 Controlling Actuators

LED is connected to pin 12 of Arduino board and ground

- User click on the LED Toggle button.
- Flask identifies the form action /LEDTOGGLE/R2 (Example R2)
- Coordinator sends the information to R2 with 64-bit mac address of R2 set to the destination.
- Router 2 receives LEDTOGGLE command, and calls the LED function

```
if (!strcmp(cmd, "LEDTOGGLE")) {
  LED();
  data = "";
  return true;
}
```

Figure 24. LED Command

- If reading ==1, LED at pin 12 blinks
- If reading ==0, LED is turned off.

5.4 Handling Errors

If a node in the network is removed then the destination device does not receive any data from this node .The server program has a delay of 10 sec to identify if the router is removed. The routing table of other nodes is automatically re-routed and is updated with new route to reach the destination device. If the temperature sensor does not transmit any data, the node waits until it fetches the temperature in order to transmit data and add it to string payload packet. If RSSI was not fetched, the node prints that the DB command could not be fetched and prints a value zero in RSSI field for the desired packet.

5.5 Integration with Augmented Reality

Augmented reality is the concept where digital image is overlayed on IOT network for real time data visualization. The data from the IOT network is integrated with Augmented Reality from the html page to create a 4D Experience for the user[29] . C sharp is used as a programming language to design the UI for visualization in Unity software. HoloLens used inbuilt tool kit from Microsoft. Vuforia is used to overlay digital image over IOT network in the form of holograms. Fog is the concept used to minimize the latency and which runs the API consisting of server[29]. IOT and Hologram interaction and how the network is visualized is as shown below

1. Ngrok is used to get the data from local host to cloud so that AR device will be able to fetch the data. An example to run ngrok is shown in Figure 25.

2. The user wears the HoloLens and then focus on the network so that it identifies the object.
3. Unity creates a web request and sets the URL as a string argument [30] as shown in Figure 26.

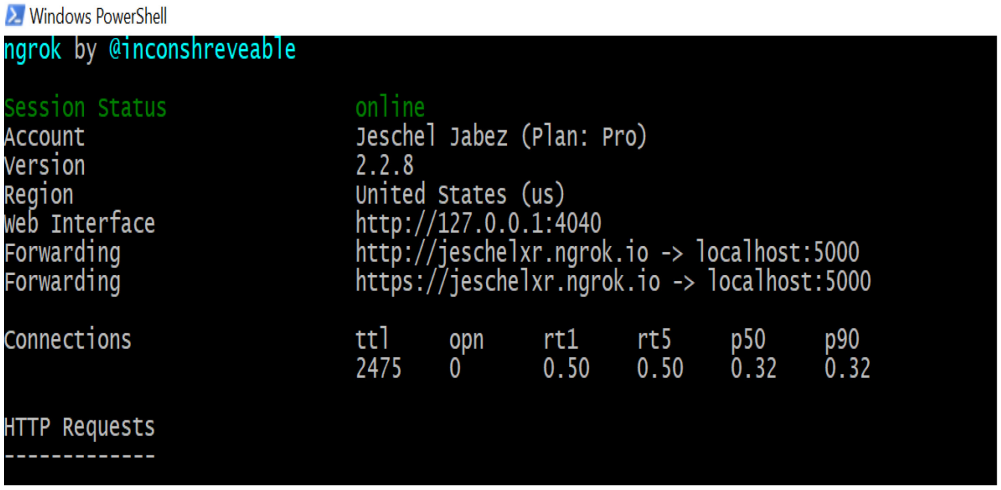


Figure 25. ngrok to host the data to internet from localhost

4. The data from the server is processed and returned as a string [30].
5. This data is accessed by unity using the link.

   For example, https://server address:5000/Sensor data/temperature-json/number=(Number of the router from 1 to 5)
6. With timeout=1sec , the data is fetched at an interval of 1sec from the server.

   Figure below shows the basic UI Design in Unity for the visualization of IOT Network. Visualization of data network can be seen in Results Section.

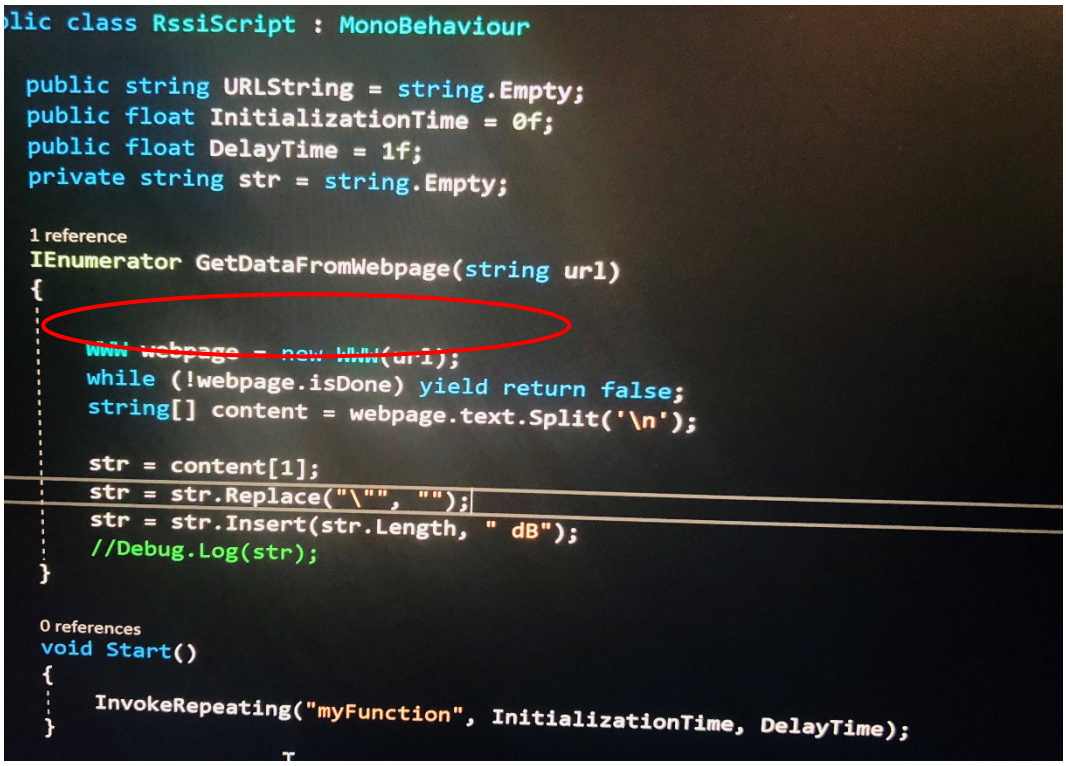


Figure 26. Hologram Script

VI. EXPERIMENTAL RESULTS

6.1 LQI with distance (Indoor Environment)

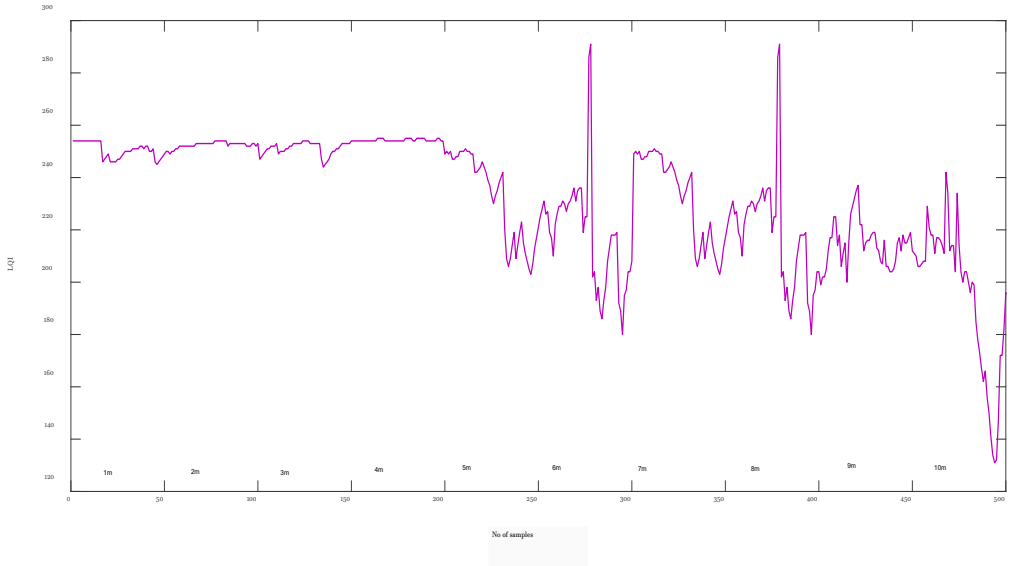


Figure 27. Graph of LQI vs distance

***INFERENCE*:**

In Figure 27 readings were taken by incrementing the distance between router and coordinator by 1 meter for every 100 samples. From the graph, we can infer that for a distance up to 4m the LQI values remained consistent and high indicating a good link. As the distance increased beyond 4m , we see it is deteriorating and is at its lowest when it is at 10m. The graph also suggests that as the distance increases beyond 5m there are variations in the LQI values because even though the distance is within the range of protocol, a lot of external interferences play a part in determining LQI.

6.2 Faraday cage

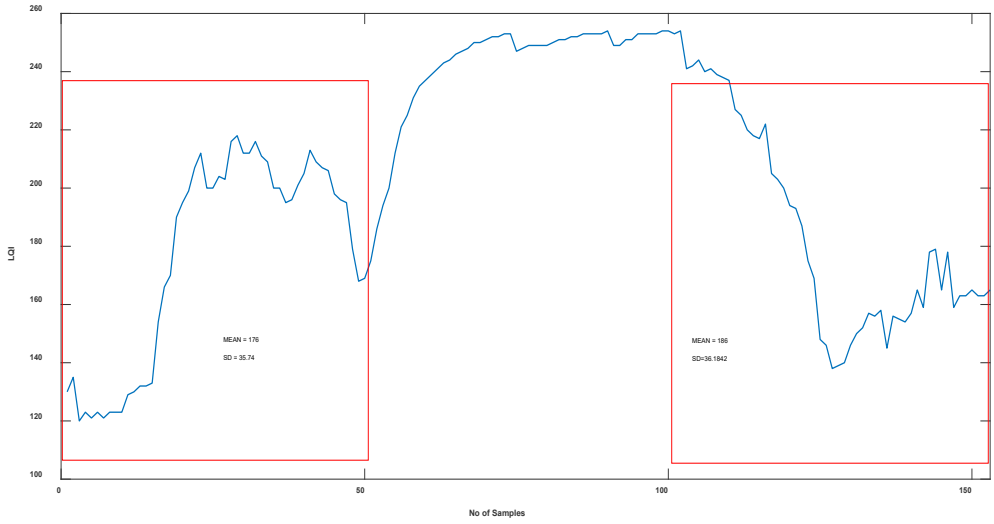


Figure 28. Graph-Faraday cage experiment

INFERENCE:

In Figure 28 , we see that the LQI remained consistent when the routers were placed within a 5m range. To observe a significant drop for a short-range communication to effect topological changes, the router was placed within a faraday cage. The figure shows that for the first 50 samples the LQI drastically when the router was placed inside the faraday cage and increased gradually as soon as the router was removed from the faraday cage. When the router is completely covered by faraday cage the LQI drops to 0 which blocks the signal completely. The mean value when the node is places inside a Faraday cage is approximately 180 when compared to the mean value of 246 when the router is placed outside the faraday cage. Both experiments were performed in the same range between the coordinator and router at a distance of 1m.

6.3 LQI when there is obstruction

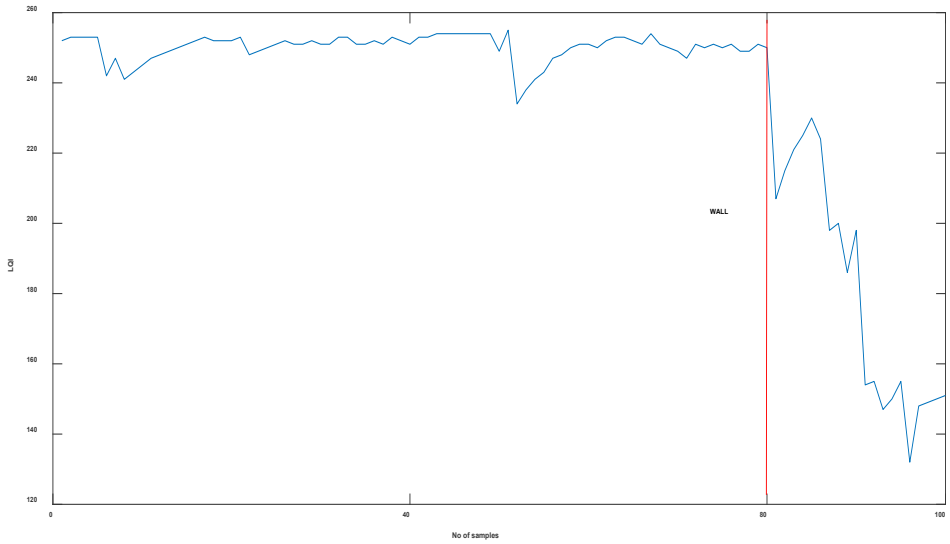


Figure 29. LQI with obstruction(Wall)

INFERENCE:

In this setup readings were taken by keeping the distance between router and coordinator at 1 meter for all the samples. From the graph, we can infer that when there is no obstruction between the router and coordinator , there were very little variations in link quality with a mean value of 247. But when there was an obstacle such as wall in between the nodes for the same distance, we could see that LQI dropped significantly due to disturbance although the link quality is still in the communication range of ZigBee and is considered as a good link for transmission. The x – axis represents the total no of samples and the y-axis represents the LQI value for each sample reading. Also, up-to 50 samples, other obstructions like metal plates , steel were places in between the nodes and the observation was made that there was no change in LQI. The reading was noted down for both with obstruction and without obstruction.

6.4 Teaching Network Topology and Routing to Students

Considering the current educational premise for network topology, there are existing challenges on how the functioning of a routing protocol can be easily explained in a discernible way to a network enthusiast. The teaching methods often involve pictorial and theoretical representations of the routing protocols and how they define the network. There are gaps in addressing changes in link behavior and detection of link failures and the pressing need to understand the same as part of a learning process. Although there are ways and means to simulate the real-world networks in a lab environment, it leaves a lot to be desired from a mapping perspective.

By leveraging Augmented reality as part of the learning experience, a student will be exposed to the functioning and the finer technical details of route creation and network formation process. A visual experience along with the ability to physically affect the different components and understand the after-effects of the same can be achieved. Considering our setup for example, if a router is manually disconnected from the network say by plugging the power cord, the process of a different route being formed by comparing the already visible LQI values can all be visualized. This kind of experience will do a world of good to the student in order to understand the routing concepts and network formation process.

- User wears the holo-lens and sees the UI as shown in Figure 30. The user can choose the desired network topology for visualization

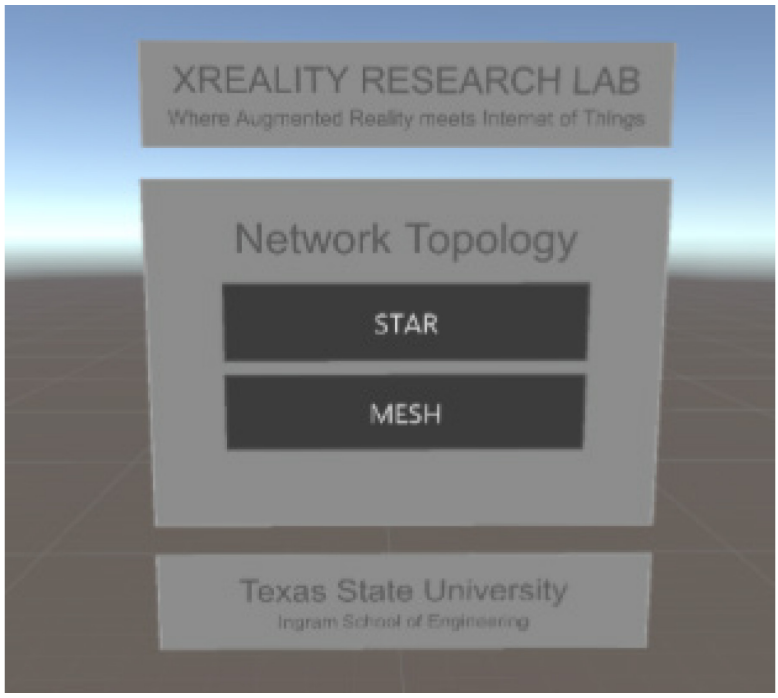


Figure 30. User Interface on Holo-lens

- To visualize the data in real time, change in temperature and signal strength, we can focus on the temp and RSSI field. The change in temp can also be observed under different conditions. In order to interact with the actuator, we can click on the LED Button. A snippet of visualization is shown in Figure 31

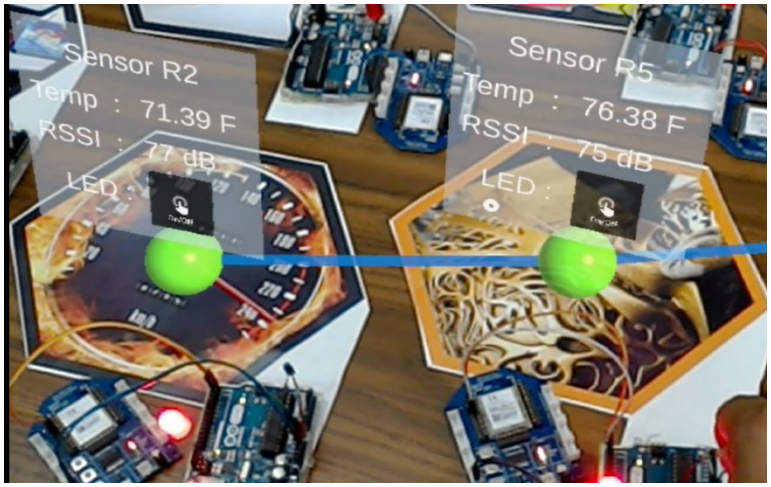


Figure 31. Temperature and RSSI Visualization

- We can visualize the star topology where all the nodes are directly connected to coordinator. Data from all the nodes is sent to the Gateway node/Coordinator as shown in Figure 32

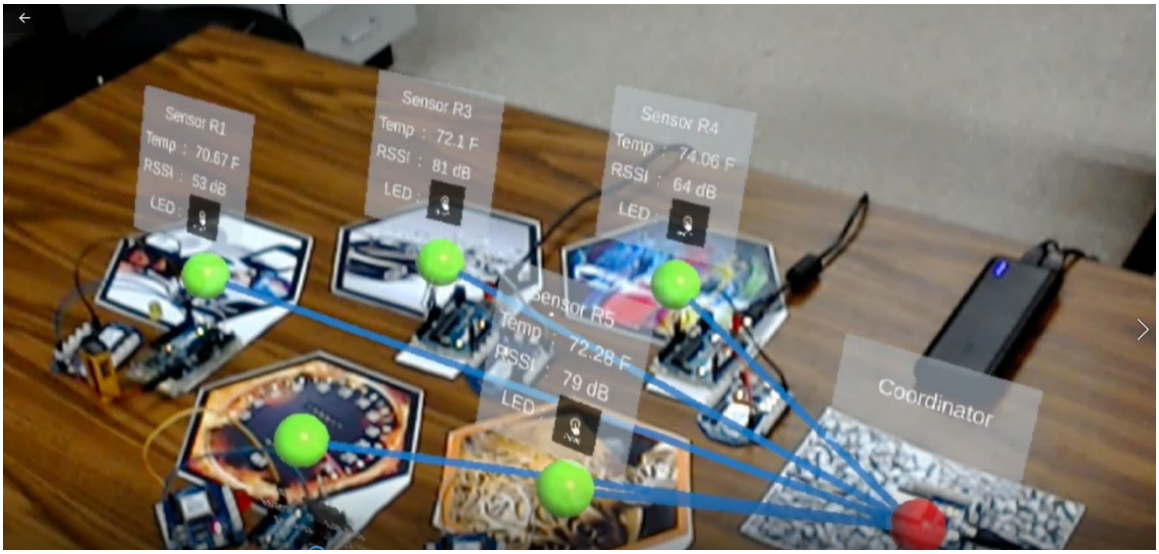


Figure 32. Star Network Visualization

- Mesh Topology is visualized where the routing is based on Link Quality Indicator which is described in detail in Chapter 5. The destination node is the coordinator/Gateway which is shown in Figure 33

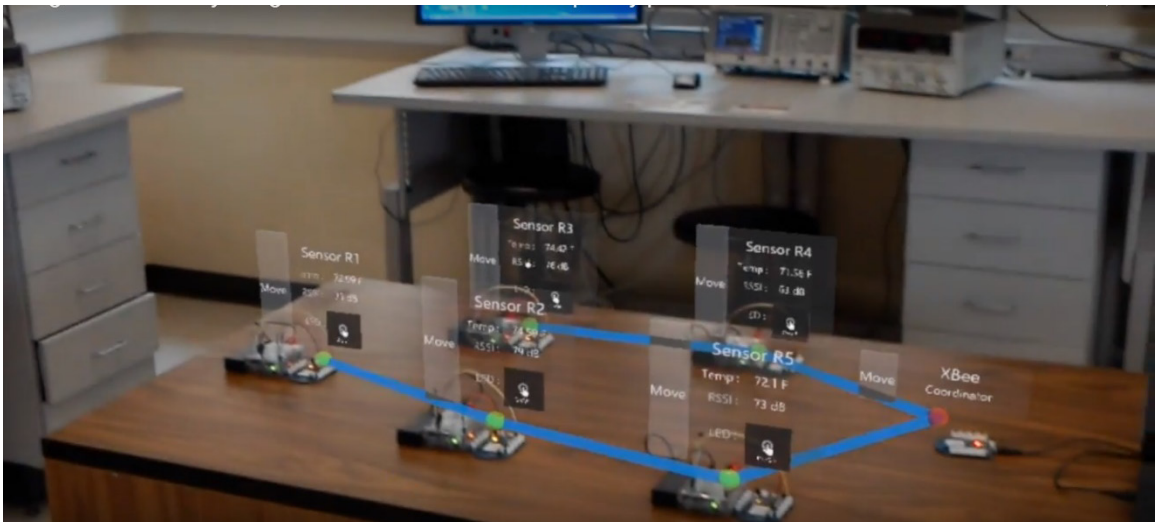


Figure 33. Mesh Network Visualization

VII. CONCLUSION AND FUTUREWORK

A proposal to implement a routing protocol that improvises on the existing ZigBee protocol to facilitate the creation of mesh networks for nodes within a short range was implemented using Arduino , Raspberry pi and XBee. Considering the existing challenges of offering an immersive visual experience to the student, the proposed protocol promises immense potential in that regard. It helps by providing network management and control that can be leveraged by augmented reality in academia. A network management portal was created to interface with AR application and validated for functionality. A comparative analysis was performed to measure the LQI and RSSI behavior under different environments and distance. It was observed that RSSI is not a good measurement for distance estimation because of its continuous fluctuations from other environmental factors. LQI measurement also showed how the link quality changed only when the distance was beyond 4m and hence for a closed range results to observe the drop in LQI we used faraday cage to see drop in LQI to reflect change in routing. This plays a significant role keeping in mind the effectiveness it provides in a classroom environment where distance between nodes is key. By the proposed protocol and portal, we can see that IoT of Wireless sensor networks in coherence with augmented reality can be a huge step in the evolution of the learning and teaching purpose in the field of education. This can be further widened to other routing realms and the possibilities are limitless. On the flipside, since augmented reality applications are growing at a rapid rate, it needs to be disseminated to satisfy the academic needs accordingly. Also, the cost model should scale for mass implementation of teaching methods.

The future work can be enhanced further by considering other parameters on offer by the XBee protocol along with LQI and RSSI to make it more efficient. Also, the behavior needs to be tested under sandboxed environment with no external interference in order to obtain benchmark data. The interfacing of the network management portal can be tested for other augmented reality applications as future work and continuing research in this field. The protocol for routing based on LQI can also be performed in other wireless technologies like thread or low power Bluetooth devices and a comparison can also be made. Another drawback which can be addressed as a future work is that the coordinator is single point of contact , so it can be implemented with having additional coordinator so that if one gateway node/coordinator for a network fails, the nodes should automatically identify the secondary gateway node and should start transmitting packets.

An evaluation for RSSI and Zigbee can also be made in a theoretical way in order to confirm the values that we are getting for the desired packet. Additional parameters can also be monitored and sent as a payload packet for better visualization and understanding of various real time data. It would also be interesting to use additional database in case the single database exceeds memory.

APPENDIX SECTION

APPENDIX A

Coordinator – cooordinator.py

```
from xbee import XBee, ZigBee
import serial
import time, sys, datetime
import MySQLdb
import MySQLdb as mysql
import mysql.connector
from mysql.connector import (connection)
from mysql.connector import errorcode
import json
import flask
from Chai_Mesh_SR import *
from xbee import ZigBee
import serial,time
import threading

#Connecting to MySQL server Using Connector/Python
config = {
  'user': 'chaithra',
  'password': 'chaithra',
  'host': '127.0.0.1',
  'database': 'temperatures',
  'raise_on_warnings': True,
  'buffered':True,
}

PORT = 'COM5'
BAUD_RATE = 9600
```

```
XBeeCoordinatortable = ["0013a200", "41680BE0", "fffe"]
datadict = {"MAC":"routerstat"}

#ZigBeeSerialData = None
zigbee = None
mutex_zigbee = threading.Lock()

def init_zigbee():
    print("init_zigbee")
    global  zigbee, mutex_zigbee
    XBeeSerialData = serial.Serial(PORT, BAUD_RATE, timeout=1)
    zigbee = ZigBee(XBeeSerialData, escaped=True)
    #mutex_zigbee.acquire()
    #try:
        #zigbee = ZigBee(ZigBeeSerialData)
    #finally:
        #mutex_zigbee.release()

def read_from_sensor():
    print("read from sensor")
    # Initialization parameters
    # Coordinator XBee SH : 0x0013a200, XBee SL : 0x41680ca1
    create_db_tables()
    create_db_tables1()
    global  zigbee, mutex_zigbee
    receive_num = 1
    latencydelta_temp = [0.0] * 5
    timedelta_temp = [0.0] * 5
    XBeeroutingtables = ''
    #time.sleep(10)
    init_zigbee()
```

```
print "Wait for temperature from sensors.."
while True:
    try:
        mutex_zigbee.acquire()
        data = zigbee.wait_read_frame()
        mutex_zigbee.release()
        if data['id'] in ['route_record_indicator', 'tx_status']:
            continue
        if not data.has_key('rf_data'):
            print "rf_data not in decoded data"
            continue
        decodedData = decodeReceivedFrame(data)
        print 'decodedData is %r' % decodedData
        try:
            json_value = json.loads(decodedData[1])
        except:
            continue
        '''
        source_mac, source_nw = get_source_of_packet(json_value)
        if not source_mac:
            print 'Ignoring packet since source mac not yet identified'
            continue
        decodedData[0] = source_mac
        decodedData[5] = source_nw
        '''

        source_mac = decodedData[0]
        source_nw = decodedData[5]
        for index, mac in enumerate(SensorSHSLAddress):
            if mac == source_mac:
```

```
                router_nw_address[index] = source_nw
            ind_router = SensorSHSLAddress.index(source_mac)
            if (receive_num <= 6):
                print str('Waiting for read sensor data No. ' + str(receive_num))
                # sensordataclearmysql(config, SensorTablesName)
            elif (receive_num > 6 and receive_num < 12):
                sensordatatemp = sensordatadecode(json_value, source_mac, ind_router)
                temperature_value = 0
                lqi_values =0
                XBeeroutingtables = "ZigBee Mesh Initialization"
                latencydelta = 0
                timedelta = 0
                print 'Initialization parameter No. ' + str(receive_num - 6);
                sensordatainsertmysql(decodedData, config, sensordatatemp,
SensorTablesName, temperature_value,XBeeroutingtables,lqi_values);
                #if receive_num==11:
                  #threading.Thread(target=threadloop).start()
            else:
                current_time = int(time.time())
                #prev_routers = device_nw_topo.keys()
                #prev_rssi_values = list(router_rssi_values)
                configure_network(source_mac, current_time)
                print "Nw topo: %r" % device_nw_topo
                sensordatatemp = sensordatadecode(json_value, source_mac, ind_router)
                print str('sensordatadecoded is ' + str(sensordatatemp))
                temperature_value = float(sensordatatemp[1].strip('\x00'))
                router_lqi_list[ind_router] = [int(x) for x in sensordatatemp[4:]]
                lqi_values = [str(x) for x in sensordatatemp[4:]]
                lqi_values =','.join(lqi_values)
                #print("lqi_values",lqi_values)
                route_table = makeroutingtable(decodedData[0], ind_router, router_lqi_list)
```

```
            #print "routing table for index: " + str(ind_router) + "," + str(route_table)
            #XBeeroutingtables = makeroutingtable_2(decodedData[0], ind_router,
json_value)
            XBeeroutingtables = makeroute(ind_router, route_table)
            latencydelta_timedelta_temp = getlatencytimedelta(latencydelta_temp,
timedelta_temp, decodedData,
                                                float(sensordatatemp[2]), SensorSHSLAddress)
            latencydelta = float(latencydelta_timedelta_temp[0])
            timedelta = float(latencydelta_timedelta_temp[1])
            sendData(SensorSHSLAddress[ind_router].decode("hex"),
"{\"DT\":\"ACK\"}")
            #new_routers = device_nw_topo.keys()
            #if set(prev_routers) != set(new_routers) or prev_rssi_values !=
router_rssi_values:
               #sendselfInfo(decodedData[0], decodedData[5])
                #sendrouterstatus(decodedData[0])
                #sendrouterandselfstatus()

            setRouterStatusinSQL(lqi_values, source_mac)
            print 'Writing Sensor data to MySQL database...'
            sensordatainsertmysql(decodedData, config,
                                  sensordatatemp, SensorTablesName, temperature_value,
                                  XBeeroutingtables,lqi_values)

         if (receive_num <= 10000):
            receive_num += 1
         else:
            receive_num =10000
            # file.close()
      except KeyboardInterrupt:
         pass
```

```
def setRouterStatusinSQL(lqi_values, mac_address):
    cnx = mysql.connector.connect(**config)
    cursor = cnx.cursor()
    print("mac_address",mac_address)
    if mac_address in device_time_map.keys():
        cursor.execute('''UPDATE ROUTERSTATUSTABLE SET Router_Status = %s ,lqi
= %s  WHERE MACADDRESS= %s ''',
                    (1,lqi_values, mac_address))
        print lqi_values
        #cursor.execute("""UPDATE ROUTERSTATUSTABLE + SET Router_Status = '1'
+ WHERE MACADDRESS = "mac_address"""");

        # UPDATE db with router status 1 for mac_address

    else:
        # UPDATE into db with router status 0 for mac_address
        cursor.execute('''UPDATE ROUTERSTATUSTABLE SET Router_Status = %s ,lqi
=%s WHERE MACADDRESS= %s ''',
                    (0,0, mac_address))

    cnx.commit()
    cnx.close()

def sendData(address, datatosend):
    global  zigbee, mutex_zigbee ,delay
    UNKNOWN = '\xff\xfe'
    mutex_zigbee.acquire()
    try:
        if zigbee:
            zigbee.send('tx',dest_addr_long=address, dest_addr=UNKNOWN,
```

```
data=datatosend)

    finally:
        mutex_zigbee.release()



def create_db_tables():
    cnx = mysql.connector.connect(**config)
    cursor = cnx.cursor()
    for i in range(0, len(SensorTablesName), 1):
        try:
            cursor.execute("SELECT COUNT(*) FROM %s" % SensorTablesName[i])
            dataCount = cursor.fetchone()
            print "table %s already exists, rows:%r" % (SensorTablesName[i], dataCount)
        except mysql.connector.ProgrammingError as e:
            if e.errno == 1146:
                print "%r" % e
                sensor_TABLES = "CREATE TABLE %s ( XBee_SH varchar(36) NOT
NULL, XBee_SL varchar(36) NOT NULL, XBee_RSSI_DB varchar(36) NOT NULL,
XBee_Routing_Tables varchar(999) NOT NULL, sensor_time varchar(36) NOT NULL,
temperature varchar(36) NOT NULL, Star_Routing_Table varchar(200) NOT NULL)"
%SensorTablesName[i]
                cursor.execute(sensor_TABLES)
            else:
                print "Unknown error"
    #cursor.execute(sensor_TABLES)
    cnx.commit()
    cursor.close()
    cnx.close()
```

```
def create_db_tables1():
    cnx = mysql.connector.connect(**config )
    cursor = cnx.cursor()
    try:
        cursor.execute("SELECT COUNT(*) FROM ROUTERSTATUSTABLE")
    except mysql.connector.ProgrammingError as e:
        Router_TABLE = "CREATE TABLE ROUTERSTATUSTABLE( MACADDRESS
varchar(36) NOT NULL, Router_Status varchar(36) NULL,lqi varchar(200) NOT
NULL)"
        cursor.execute(Router_TABLE)
        for mac_address in SensorSHSLAddress:

            cursor.execute("""INSERT INTO ROUTERSTATUSTABLE VALUES
(%s,%s,%s)""",(mac_address,"0","0"))

    cnx.commit()
    cursor.close()
    cnx.close()

def hex(bindata):
    return ''.join('%02x' % ord(byte) for byte in bindata)



# Decode Received XBee Frame
def decodeReceivedFrame(data):
    source_addr_long = hex(data['source_addr_long'])
    source_addr = hex(data['source_addr'])
    sensor_data = data['rf_data']
    frametype = data['id']
    frametime = datetime.datetime.now().strftime('%d-%m-%Y %H:%M:%S')
    timedelta = time.time() * 1000
```

```
    device_mac_names[source_addr_long.lower()] = source_addr.lower()
    return [source_addr_long.lower(), sensor_data, frametype, frametime, timedelta,
source_addr.lower(), data['source_addr_long']]

# Decode Sensor data from XBee frame
def sensordatadecode(json_value, source_mac, router_index):
    sensordatatemp = ['','0','0','0']
    if json_value and json_value.get('DT') and json_value.get('DT') == 'PD':
        list_of_rfdata = json_value.get('IN').split(",")
        sensordatatemp = [source_mac]
        sensordatatemp += list_of_rfdata
    return sensordatatemp

def get_source_of_packet(json_value):
    source_index = json_value.get('SI')
    if source_index and source_index != "-1":
        return (SensorSHSLAddress[int(source_index)], json_value.get('SNA'))
    return ('','fffe')

# Calculate XBee Latency and Time delta
def getlatencytimedelta(latencydelta_temp, timedelta_temp, decodedData, arduino_delay,
SensorSHSLAddress):
    print "Latency: " + str(latencydelta_temp) + ","+ str(timedelta_temp) + "," +
str(arduino_delay)
    if not decodedData[0]:
        return [0, 0]
    i = SensorSHSLAddress.index(decodedData[0])
    latencydelta = decodedData[4] - latencydelta_temp[i] - arduino_delay
    if latencydelta < 0.0:
        latencydelta = 0.0
    timedelta = latencydelta - timedelta_temp[i]
```

```
    latencydelta_temp[i] = decodedData[4]
    timedelta_temp[i] = latencydelta
    print "Latency after: " + str(latencydelta_temp) + ","+ str(timedelta_temp) + "," +
str(arduino_delay)
    return [latencydelta, timedelta]


# Insert sensor data into the MySQL database
def sensordatainsertmysql(decodedData, config,
                          sensordata_decoded, SensorTablesName, temperature_value,
                          XBeeroutingtables,lqi_values):
    cnx = mysql.connector.connect(**config)
    cursor = cnx.cursor()
    i = 0
    ind_router = SensorSHSLAddress.index(decodedData[0])
    star_routing_table = router_name[ind_router]
    star_routing_table += '->' + "co-ordinator"
    while (str(decodedData[0][0:16]) != SensorSHSLAddress[i]):
        i += 1
    else:
        add_sensor_count = "INSERT INTO " + SensorTablesName[
            i] + " (XBee_SH, XBee_SL, XBee_RSSI_DB, XBee_Routing_Tables,
sensor_time, temperature,Star_Routing_Table) VALUES ( %(XBee_SH)s,
%(XBee_SL)s, %(XBee_RSSI_DB)s,  %(XBee_Routing_Tables)s, %(sensor_time)s,
%(temperature)s, %(Star_Routing_Table)s)"
    data_sensor_count = {
        'XBee_SH': str(decodedData[0][0:8]),
        'XBee_SL': str(decodedData[0][8:16]),
       # 'XBee_API_Mode_AP': str(sensordata_decoded[5]),
       # 'XBee_Sleep_Mode_SM': str(sensordata_decoded[6]),
        'XBee_RSSI_DB': str(int(sensordata_decoded[3], 16)),
```

```
        #'XBee_Baud_Rate_BD': Baud_Rate_BD_Strings
        'XBee_Routing_Tables': XBeeroutingtables,
        'sensor_time': decodedData[3][11:19],
        'temperature': temperature_value,
        'Star_Routing_Table': lqi_values,
    }
    cursor.execute(add_sensor_count, data_sensor_count)
    cnx.commit()
    cursor.close()
    cnx.close()
    print 'Sensor data have been written to MySQL database'
    print '------------------'

# Clear MySQL database
def sensordataclearmysql(config, SensorTablesName):
    cnx = mysql.connector.connect(**config)
    cursor = cnx.cursor()
    for i in range(0, len(SensorTablesName), 1):
        cursor.execute("truncate table %s" % (SensorTablesName[i]))
    cnx.commit()
    cursor.close()
    cnx.close()

is_star= False
def star():
    global is_star
    is_star = True

def mesh():
    global is_star
    is_star = False
```

```
def makeroutingtable(mac_address, router_index, router_lqi):
    route_table = []
    while route_table == [] or route_table[-1] != -1:
        next_hop = -1
        hop_index = router_index
        if  route_table:
            hop_index = route_table[-1]
        if is_star:
            route_table.append(next_hop)
            return route_table

def makeroutingtable_2(mac_address, router_index, json_value):
    routing_table = ""
    router_index = SensorSHSLAddress.index(mac_address)
    routing_table += router_name[router_index]
    hops = json_value.get('HP')
    if hops:
        for nw_address in hops:
            try:
                router_index = router_nw_address.index(nw_address)
                routing_table += "->" + router_name[router_index]
            except:
                 continue

    routing_table += '->' + "co-ordinator"
    return routing_table

def threadloop():
    prev_routers = []
    while True:
```

```
        time.sleep(1)
        current_time = int(time.time())
        remove_missing_devices(current_time)
        new_routers = device_nw_topo.keys()
        if set(prev_routers) != set(new_routers):
            sendrouterandselfstatus()
            prev_routers = new_routers

def sendrouterandselfstatus():
    datadict = {}
    datadict['DT'] = 'ST'
    datadict['ST'] = 0
    for index, mac_address in enumerate(SensorSHSLAddress):
        if mac_address in device_time_map.keys():
            datadict['ST'] = datadict['ST'] | (1 << index)
        else:
            datadict['ST'] = datadict['ST'] & (0xFF ^(1 << index))
        #if router_index == index:
        #    datadict[str(index)] = router_rssi_values[index]

    for index, mac_address in enumerate(SensorSHSLAddress):
        print("sending router status: to ", mac_address, json.dumps(datadict))
        time.sleep(20/1000);
        sendData(mac_address.decode("hex"), json.dumps(datadict))
```

APPENDIX B

Flask program – flask.py

```
from flask import Flask
from flask import render_template, jsonify, request ,url_for
from flask_mysqldb import MySQL
import MySQLdb
from flask_table import table, columns
import numpy
import serial , time
from xbee import ZigBee
from flask_assets import Bundle ,Environment
from Chai_Coordinator_SR import sendData, read_from_sensor,star,mesh
from threading import Thread
from time import sleep
from Chai_Mesh_SR import SensorSHSLAddress



print("flask init")
app = Flask(__name__, template_folder='D:\\Courses\\Python\\Python-
courseera\\Chaithra_Thesis\\templates'
            ,static_folder='D:\\Courses\\Python\\Python-courseera\\Chaithra_Thesis\\static')

js = Bundle('api.js','jquery-1.11.3.js','jssor.slider-
25.2.1.min.js','jquery.min.js',output='gen/main_js')
js1
=Bundle('jquery.min.js','bootstrap.min.js','ResponsiveNav.min.js','script.min.js','docs.js','g
oTop.min.js','livereload.js',output='gen/main_js1')
css = Bundle('bootstrap.min.css','responsive.css','master.css','docs.css','flag-
icon.min.css','font-awesome.min.css',
             'cssd.css',output='gen/main_css')
```

```
assets = Environment(app)
assets.register('main_js',js)
assets.register('main_js1',js1)
assets.register('main_css',css)

@app.route('/sensor-data/temperature-json/')
def temperature_json():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()

    sensor_number = request.args.get('number')
    cur.execute("SELECT temperature   FROM sensor_data" + sensor_number + " order
by sensor_time desc limit 1")
    temp = cur.fetchall()
    i = 0
    if len(temp) > 0:
        for row in temp:
            alltemp.append(row[0])
    cur.close()
    connection.close()
    return jsonify(alltemp)



@app.route('/sensor-data/routingtable-json/')
def routingtable_json():
```

```
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    sensor_number = request.args.get('number')
    cur.execute("SELECT XBee_Routing_Tables   FROM sensor_data" + sensor_number
+ " order by sensor_time desc limit 1")
    temp = cur.fetchall()
    i = 0
    if len(temp) > 0:
        for row in temp:
            alltemp.append(row[0])
    return jsonify(alltemp)
    cur.close()
    connection.close()


@app.route('/sensor-data/rssi-json/')
def rssi_json():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
```

```
    sensor_number = request.args.get('number')
    cur.execute("SELECT XBee_RSSI_DB   FROM sensor_data" + sensor_number + "
order by sensor_time desc limit 1")
    temp = cur.fetchall()
    i = 0
    if len(temp) > 0:
        for row in temp:
            alltemp.append(row[0])
    cur.close()
    connection.close()
    return jsonify(alltemp)

@app.route('/sensor-data/lqi1/')
def lqi1():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    cur.execute("SELECT lqi   FROM routerstatustable WHERE MACADDRESS =
'0013a20041680ca1'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data
```

```
@app.route('/sensor-data/lqi2/')
def lqi2():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    cur.execute("SELECT lqi   FROM routerstatustable  WHERE MACADDRESS =
'0013a20041481713'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data

@app.route('/sensor-data/lqi3/')
def lqi3():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    cur.execute("SELECT lqi FROM routerstatustable WHERE MACADDRESS =
'0013a20041481719'")
```

```
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data

@app.route('/sensor-data/lqi4/')
def lqi4():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    cur.execute("SELECT lqi FROM routerstatustable WHERE MACADDRESS =
'0013a200414812f4'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data

@app.route('/sensor-data/lqi5/')
def lqi5():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
```

```
                        user='chaithra',
                        passwd='chaithra',
                        db='temperatures'
                        )
    cur = connection.cursor()
    cur.execute("SELECT lqi   FROM routerstatustable WHERE MACADDRESS =
'0013a20041481300'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data

@app.route('/sensor-data/rs1/')
def rs1():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                        user='chaithra',
                        passwd='chaithra',
                        db='temperatures'
                        )
    cur = connection.cursor()
    cur.execute("SELECT Router_Status   FROM routerstatustable WHERE
MACADDRESS = '0013a20041680ca1'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
```

```
    return data

@app.route('/sensor-data/rs2/')
def rs2():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    cur.execute("SELECT Router_Status   FROM routerstatustable  WHERE
MACADDRESS = '0013a20041481713'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data

@app.route('/sensor-data/rs3/')
def rs3():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
```

```
    cur.execute("SELECT Router_Status FROM routerstatustable WHERE
MACADDRESS = '0013a20041481719'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data

@app.route('/sensor-data/rs4/')
def rs4():
    alltemp = []

    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    cur.execute("SELECT Router_Status FROM routerstatustable WHERE
MACADDRESS = '0013a200414812f4'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data

@app.route('/sensor-data/rs5/')
def rs5():
    alltemp = []
```

```
    connection = MySQLdb.connect(host='localhost',
                                 user='chaithra',
                                 passwd='chaithra',
                                 db='temperatures'
                                 )
    cur = connection.cursor()
    cur.execute("SELECT Router_Status   FROM routerstatustable WHERE
MACADDRESS = '0013a20041481300'")
    temp = cur.fetchone()
    i = 0
    cur.close()
    connection.close()
    data = jsonify(temp)
    return data



@app.route("/R1")
def controlon1():
    WHERE = '\x00\x13\xA2\x00\x41\x68\x0c\xa1'
    sendData(WHERE, '{"DT":"LEDTOGGLE"}')
    return render_template("xreality.html")

@app.route("/R2")
def controlon2():
    WHERE = '\x00\x13\xA2\x00\x41\x48\x17\x13'
    sendData(WHERE, '{"DT":"LEDTOGGLE"}')
    return render_template("xreality.html")

@app.route("/R3")
```

```
def controlon3():
    WHERE = '\x00\x13\xA2\x00\x41\x48\x17\x19'
    sendData(WHERE, '{"DT":"LEDTOGGLE"}')
    return render_template("xreality.html")



@app.route("/R5")
def controlon5():
    WHERE = '\x00\x13\xA2\x00\x41\x48\x13\x00'
    sendData(WHERE, '{"DT":"LEDTOGGLE"}')
    return render_template("xreality.html")

@app.route("/STAR")
def controlonstar():
    star()
    #for index, mac_address in enumerate(SensorSHSLAddress):
        #time.sleep(20 / 1000);
    return render_template("xreality.html")

@app.route("/MESH")
def controlonmesh():
    mesh()
    #for index, mac_address in enumerate(SensorSHSLAddress):
     #   time.sleep(20 / 1000);
    return render_template("xreality.html")

@app.route("/")
def thisfunc():
    return render_template("xreality.html")

if __name__ == "__main__":
```

```
#init_zigbee()
thread = Thread(target=read_from_sensor)
thread.start()
#thread.join()
print "thread for read_from_sensor started"
app.run(debug=False )
thread.join()
```

REFERENCES

[1] http://www.logistiikanmaailma.fi/en/logistics/digitalization/internet-of-things-iot/

[2] Lee, K. (2012). Augmented reality in education and training. *TechTrends*, *56*(2), 13-21

[3] Lee, J. S., Su, Y. W., & Shen, C. C. (2007, November). 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* (pp. 46-51). Ieee

[4] Khanh, H. S., & Kim, M. K. (2015). LQDV Routing Protocol Implementation on Arduino Platform and Xbee module. In *Information Science and Applications* (pp. 173-180). Springer, Berlin, Heidelberg

[5] Shokri, M., & Gheisari, M. A New Algorithm for Routing in Zigbee Networks

[6] Mayalarp, V., Limpaswadpaisarn, N., Poombansao, T., & Kittipiyakul, S. (2010, May). Wireless mesh networking with XBee. In *2nd ECTI-Conference on Application Research and Development (ECTI-CARD 2010), Pattaya, Chonburi, Thailand* (pp. 10-20).

[7] Yang, T., Yang, Q., & Cheng, L. (2015). Experimental study: a LQI-based ranging technique in ZigBee sensor networks. *International Journal of Sensor Networks*

[8] Filsoof, R., Bodine, A., Gill, B., Makonin, S., & Nicholson, R. (2014, June). Transmitting patient vitals over a reliable ZigBee mesh network. In *Humanitarian Technology Conference-(IHTC), 2014 IEEE Canada International* (pp. 1-5). IEEE

[9] Sunehra, D., & Ramakrishna, P. (2016, December). Web based patient health monitoring system using Raspberry Pi. In *Contemporary Computing and Informatics (IC3I), 2016 2nd International Conference on* (pp. 568-574). IEEE

[10] Singh, P., & Saikia, S. (2016, December). Arduino-based smart irrigation using water flow sensor, soil moisture sensor, temperature sensor and ESP8266 WiFi module. In *Humanitarian Technology Conference (R10-HTC), 2016 IEEE Region 10* (pp. 1-4). IEEE

[11] Kaufmann, H. (2003). Collaborative augmented reality in education. *Institute of Software Technology and Interactive Systems, Vienna University of Technology*

[12] Haramaki, T., & Nishino, H. (2015, November). A Device Identification Method for AR-Based Network Topology Visualization. In *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2015 10th International Conference on* (pp. 255-262). IEEE

[13] https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf

[14] https://books.google.com/books?hl=en&lr=&id=up8Oa7456I8C&oi=fnd&pg=PP1&dq=+Gislason,+D.+2008.+ZigBee+Is+Highly+Reliable.+ZIGBEE+WIRELESS+NETWORKING.+2008+Elsevier+Inc.+on+pp.+4-5.+&ots=IFEw7-zPDu&sig=YKIkIEcC2GOCEDPVIpOfgY-G9t4#v=onepage&q&f=false

[15] https://en.wikipedia.org/wiki/Zigbee

[16] https://electronicsforu.com/resources/learn-electronics/zigbee-technology-applications

[17] https://www.digi.com/resources/documentation/Digidocs/90002002/Content/Reference/r_zb_stack.htm?TocPath=zigbee%20networks%7C_____3

[18] https://www.researchgate.net/publication/309669667_A_Comparative_Study_of_Thread_Against_ZigBee_Z-Wave_Bluetooth_and_Wi-Fi_as_a_Home-Automation_Networking_Protocol

[19] https://www.digi.com/resources/documentation/digidocs/pdfs/90001500.pdf

[20] https://www.elprocus.com/what-is-zigbee-technology-architecture-and-its-applications/

[21] Othman, M. F., & Shazali, K. (2012). Wireless sensor network applications: A study in environment monitoring system. *Procedia Engineering*, *41*, 1204-1210.

[22] Ngoc, T. V. (2008). Medical applications of wireless networks. *Washington University, St. Louis, Student Reports on Recent Advances in Wireless and Mobile Networking*.

[23] https://www.alibaba.com/product-detail/Zigbee-Wireless-Magnetic-Vehicle-Sensor-and_1634043792.html

[24] https://en.wikipedia.org/wiki/Arduino_Uno

[25] http://www.ti.com/product/LM35

[26] https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/

[27] http://docs.digi.com/pages/viewpage.action?pageId=3408008

[28] http://ftp1.digi.com/support/images/APP_NOTE_XBee_ZigBee_Device_Profile.pdf

[29] Koutitas, G., Jabez, J., Grohman, C., Radhakrishna, C., Siddaraju, V., & Jadon, S. (2018, April). Demo/poster abstract: XReality research lab—Augmented reality meets Internet of Things. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*(pp. 1-2). IEEE

[30] https://docs.unity3d.com/Manual/UnityWebRequest-RetrievingTextBinaryData.html