



Österbottens förbund
Pohjanmaan liitto

Regional Council
of Ostrobothnia

SMART ENERGY SYSTEMS RESEARCH PLATFORM

SESP

PROJECT REPORT

Wireless Mesh Network Implementation

(A Proof-of-Concept using ZigBee Wireless Mesh Network)

Akpo Siemuri

24.04.2019



Vaasan yliopisto
UNIVERSITY OF VAASA

TABLE OF CONTENTS

ABBREVIATIONS	2
LIST OF FIGURES	3
ABSTRACT	5
1. INTRODUCTION	7
1.1. Mesh Network and Wireless Mesh Network	7
1.2. Wireless Mesh Network Architecture	9
1.3. Characteristics of Wireless Mesh Network	12
1.4. General Application Scenarios	13
1.5. Fundamental Design Factors	14
2. ZIGBEE PROTOCOL AND COMMUNICATION	17
2.1. Implementing ZigBee Wireless Mesh Network	18
2.2. System Architecture	19
2.3. System Configuration	19
2.3.1. <i>Configuration Modes for XBee Modules</i>	20
2.3.2. <i>X-CTU software</i>	21
2.3.3. <i>XBee Configuration for Coordinator and End Device</i>	22
2.3.4. <i>XBee Configuration for Routers</i>	23
2.3.5. <i>XBee Configuration for End-Device</i>	24
3. IMPLEMENTATION AND RESULTS	25
3.1. Implementing XBee-Arduino, LCD-Arduino and LDR-Arduino Interfaces	25
3.2. Programming the XBee-Arduino Interface	28
3.2.1. <i>Programming XBee Coordinator-Arduino Interface as Receiver</i>	30
3.2.2. <i>Programming XBee End-Device-Arduino Interface as Transmitter</i>	31
3.3. Project Outcome and Results	32
3.3.1. <i>Testing Communications</i>	33
3.3.2. <i>RSSI and Wireless Mesh Network</i>	36
4. CONCLUSION	37
LIST OF REFERENCES	38
APPENDIX	39
APPENDIX 1. Hardware Implementation Setup.	39
APPENDIX 2. Block diagram of hardware implementation.	40
APPENDIX 3. XBee-Arduino TX/RX connection schematic view (Robert Faludi 2011).	41
APPENDIX 4. Arduino IDE environment.	42
APPENDIX 5. Code for XBee End Device-Arduino Interface as Transmitter.	43
APPENDIX 6. Code for XBee Coordinator-Arduino Interface as Receiver.	46

ABBREVIATIONS

MAC	Media Access Control
MIMO	Multiple-Input and Multiple-Output
PC	Personal Computer
WLANs	Wireless Local Area Networks
WMANs	Wireless Metropolitan Area Networks
WMNs	Wireless Mesh Networks
WPANs	Wireless Personal Area Networks
ZC	ZigBee Coordinator
ZR	ZigBee Router
ZED	ZigBee End Device

LIST OF FIGURES

FIGURE 1: WIRELESS TOPOLOGIES (ROBERT FALUDI 2011).....	8
FIGURE 2: INFRASTRUCTURE/BACKBONE WMNS (AKYILDIZ, IAN F., & XUDONG WANG 2009).....	10
FIGURE 3: CLIENT WMNS (AKYILDIZ, IAN F., & XUDONG WANG 2009).....	11
FIGURE 4: HYBRID WMNS (AKYILDIZ, IAN F., & XUDONG WANG 2009).....	11
FIGURE 5: IMPLEMENTATION SETUP BLOCK DIAGRAM.....	19
FIGURE 6: XBEE RADIO PLUG INTO EXPLORER BOARD.....	20
FIGURE 7: CHANNEL AND NETWORK ID SETUP.	21
FIGURE 8: CONFIGURE AS COORDINATOR.....	22
FIGURE 9: COORDINATOR DESTINATION ADDRESS AND DEVICE NAME SETUP.	22
FIGURE 10: ENCRYPTION AND CONFIGURATION MODE (API 2) SETUP.	23
FIGURE 11: CONFIGURE AS ROUTER.	23
FIGURE 12: ROUTER DESTINATION ADDRESS AND DEVICE NAME SETUP.....	24
FIGURE 13: ENCRYPTION AND CONFIGURATION MODE (AT) SETUP.	24
FIGURE 14: XBEE-ARDUINO INTERFACE (ELECTRONICWINGS 2019).	26
FIGURE 15: LCD-ARDUINO INTERFACE (MATTHEW McMILLAN 2019).....	27
FIGURE 16: LDR-ARDUINO INTERFACE (PANKAJ KHATRI 2018).	28
FIGURE 17: XBEE COORDINATOR-ARDUINO FLOWCHART.	30
FIGURE 18: XBEE END-DEVICE-ARDUINO FLOWCHART.....	31
FIGURE 19: WIRELESS MESH NETWORK LAYOUT.	33
FIGURE 20: ARDUINO SERIAL MONITOR INTERFACE FOR COORDINATOR (RECEIVER).....	34
FIGURE 21: ARDUINO SERIAL MONITOR INTERFACE FOR END DEVICE (TRANSMITTER).	34
FIGURE 22: VIEW OF THE GENERATED DATA IN XCTU FOR ROUTER 3.....	35
FIGURE 23: VIEW OF THE DATA BEING RECEIVED FROM THE "END DEVICE (SENSOR NODE)".	35
FIGURE 24: COORDINATOR (RECEIVER) LCD DISPLAY OF RECEIVED DATA FROM ROUTER 3.	36
FIGURE 25: COORDINATOR (RECEIVER) LCD DISPLAY OF RECEIVED DATA FROM END DEVICE (SENSOR NODE).	36

ABSTRACT

Wireless Mesh Networks are an emerging technology which may bring to reality the goal of a world that is seamlessly connected. Making use of Wireless mesh networks, entire cities can be easily, effectively and wirelessly connected using inexpensive, existing technologies. The existing traditional networks depend on a small number of wired access points or wireless hotspots to connect users. Whereas, a wireless mesh network connection spreads out among dozens or even hundreds of wireless mesh nodes that "communicate" with each other to extend the network connection across a very large and wide area.

This project is made up of two parts; the first part uses the X-CTU application to configure the XBee RF modules for the ZigBee network using ZigBee Mesh Kit (Part Number: XKB2-Z7T-WZM) from www.digi.com. The XBee used are the Series 2 model (this is important to note as a series 1 model only allows communication between a maximum of 2 modules). This kit comes with 3 XBee radios included and since the system is modular in terms of expansion, you can add as many radios as you like. The second part involves the use of Arduino development board to implement the ZigBee Wireless Mesh Network using the XBees RF modules.

The project is a proof-of-concept to demonstrate the capability of Wireless Mesh Network and possible application scenario. The sensors (for example, smart NO_x sensor, temperature, Humidity, etc.) are connected to the ZigBee END Device (ED). The sensors data are collected by the END Devices and are routed through the ZigBee Routers (ZR) to the ZigBee Coordinator (ZC). Criteria like maximum transmission distance, reliability and security of the wireless mesh network are considered. The final design is developed, implemented, analyzed and a conclusion is made based on the proof-of-concept of the ZigBee Wireless Mesh Network.

KEY WORDS: RSSI: Received Signal Strength Indicator, Smart NO_x sensor, Wireless Communication, Wireless Mesh Network, Wireless Sensor Network, ZigBee

1. INTRODUCTION

Wireless mesh networks are made up of simple mesh routers, mesh end-devices and mesh clients. The mobility of the mesh router is minimal and forms the backbone of WMNs. They provide network access for both mesh and conventional clients. It is possible to integrate the WMNs with other networks such as the Internet, cellular, IEEE 802.11, IEEE 802.15, IEEE 802.16, sensor networks, etc., making use of the gateway and bridging functions the mesh routers provide. The mesh clients are usually either stationary or mobile and could be used to create a client mesh network among themselves and/or with mesh routers. WMNs can be used to resolve the limitations as well as significantly improve the performance of ad hoc networks, wireless local area networks (WLANs), wireless personal area networks (WPANs), and wireless metropolitan area networks (WMANs) all of which provides wireless services to a large variety of applications in personal, local, campus, and metropolitan areas (Akyildiz & Wang 2009).

It is not difficult to implement and deploy a WMN because all the required components are already available in the form of ad hoc routing protocols. However, to make a WMN to perform to its maximum capability, considerable research efforts are still needed. For instance, the available MAC and routing protocols used in WMNs does not have enough scalability; and the throughput drops significantly as the number of nodes or hops increases. The existing security schemes may be effective to prevent certain types of attack, but they lack a complete mechanism to prevent attacks that may arise from different protocol layers. This type of problems exists in other networking protocols as well. Thus, the existing communication protocols, from the application layer to transport, routing, MAC, and physical layers, need to be re-evaluated and improved and in some cases, new protocols may have to be invented (Akyildiz & Wang 2009).

1.1. Mesh Network and Wireless Mesh Network

In the mesh network, there is no dependency on a single node, and this allows every node to participate in the relay of information. Mesh networks can dynamically self-organize and self-configure, making installation time to be reduced. The network's ability to self-configure

makes the dynamic distribution of workloads possible, specifically in the situation a node fails. This, in turn, contributes to fault-tolerance and reduced maintenance costs. Mesh topology may be compared with the conventional star or tree local network topologies in which the bridges and/or switches are directly linked to only a small subset of other bridges and/or switches, and the links between these infrastructure neighbors are hierarchical. While star and tree topologies are very well established, highly standardized and vendor-neutral, common standards have not yet been agreed by the mesh network devices vendors, therefore, interoperability between devices from different vendors is not yet assured (Wikipedia 2019a).

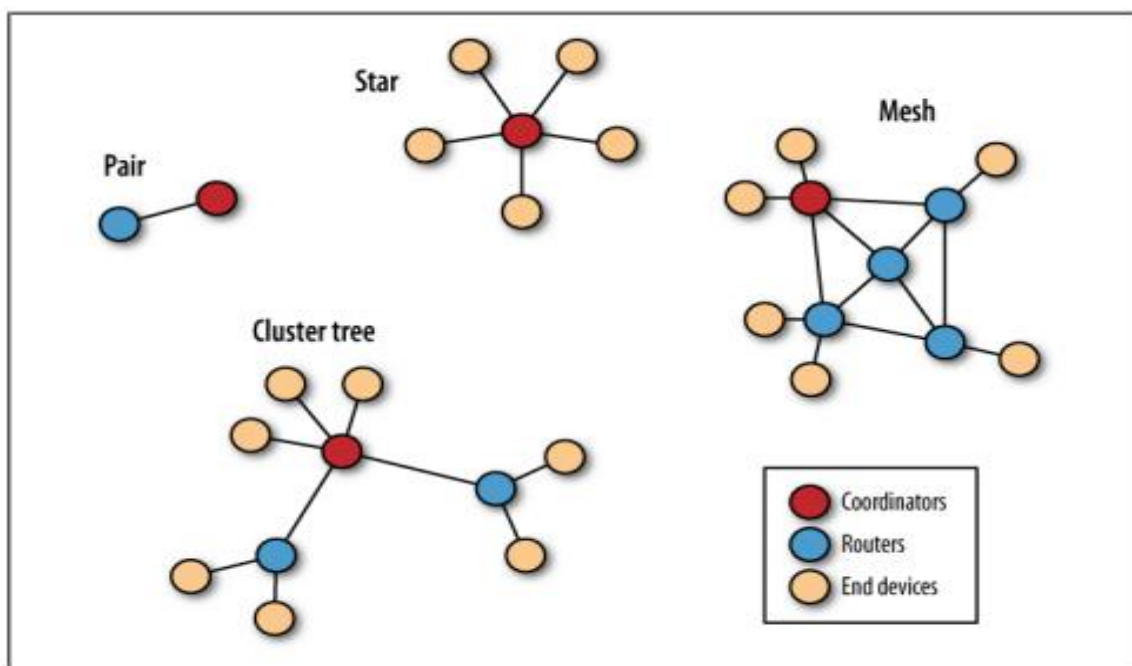


Figure 1: Wireless Topologies (Robert Faludi 2011).

As illustrated in figure 1, wireless mesh networks are usually made up of mesh clients, mesh routers and gateways. Otherwise called in the ZigBee case: Coordinators, Routers and End devices. Mesh clients could be cell phones, laptops, and other wireless devices. Mesh routers forward messages to and from the gateways, which may or may not be connected to the Internet. The mesh network is reliable and provides redundancy, for instance, the failure of one node will not affect the network because the rest of the nodes can still communicate with

each other, either directly or using one or more intermediate nodes. This implies that a wireless mesh network can self-form and selfheal (Wikipedia 2019b).

Mesh networks relay messages making use of either a flooding technique or a routing technique. Routing technique uses message propagation along a path by hopping from node to node until it reaches the destination node. It ensures that all its paths are available by allowing continuous connections and reconfiguring itself around broken paths, making use of self-healing algorithms such as Shortest Path Bridging. (Wikipedia 2019b).

Wireless mesh architecture provides cost-effective and low mobility over a specific coverage area. Wireless mesh infrastructure is a network of routers without the cabling between nodes. It is made up of peer radio devices that do not have to be cabled to a wired port like the traditional WLAN access points (AP). Mesh infrastructure can carry data over large distances by splitting the distance into a series of short hops with the help of several nodes. Intermediate nodes not only boost the signal but cooperatively move the data from point A to point B making use of forwarding decisions based on their knowledge of the network, i.e. it performs the routing by first deriving the topology of the network. Wireless mesh networks are a relatively "stable-topology" network except for instances when there is a failure of nodes or when a new node is been added. High rate of mobility among nodes can lead to link break to occur more frequently and the wireless mesh networks start to break down and have low communication performance (J. Jun & M.L. Sichitiu 2003).

1.2. Wireless Mesh Network Architecture

The WMNs architecture can be classified into three main groups based on the functionality of the nodes.

Infrastructure/Backbone WMNs: The architecture is shown in Figure 2, with the dashed and solid lines representing wireless and wired links, respectively. This type of WMN has the mesh routers forming the infrastructure for clients that connect to them. The WMN infrastructure/backbone can be built using different types of radio technology. The mesh routers are used to form a mesh of self-configuring, self-healing links among themselves. Mesh routers, having gateway functionality can allow connection to the Internet. This approach is called

infrastructure meshing and provides the backbone for conventional clients as well as enables the integration of WMNs with the existing wireless networks, using the gateway/bridge functionalities of the mesh routers (Akyildiz & Wang 2009).

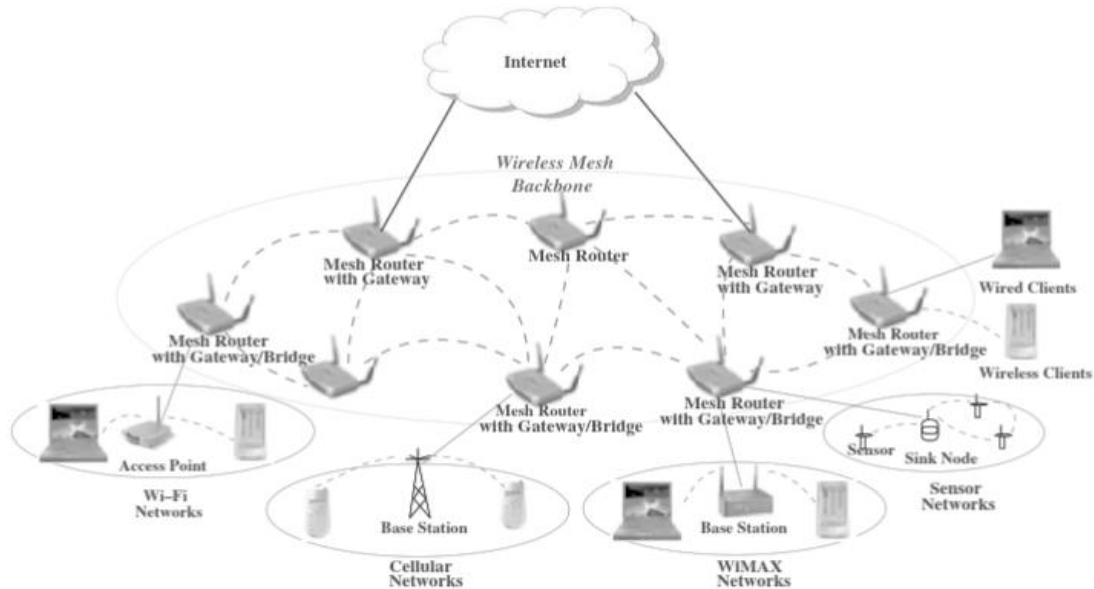


Figure 2: Infrastructure/Backbone WMNs (Akyildiz, Ian F., & Xudong Wang 2009).

- Client WMNs: Meshing the clients will provide a peer-to-peer network among client devices. This type of architecture, the client nodes makes up the network itself and performs routing and configuration functionalities in addition to providing end-user applications to customers. Therefore, this network does not require a mesh router. The basic architecture is shown in Figure 3. In Client WMNs, a packet reaches its destination by hopping through multiple nodes in the network to reach its destination. To create a Client WMNs one radio type is usually used on devices. Since the end users must do additional functions such as routing and self-configuration, the requirements on end-user devices is increased unlike in the case of infrastructure meshing (Akyildiz & Wang 2009).

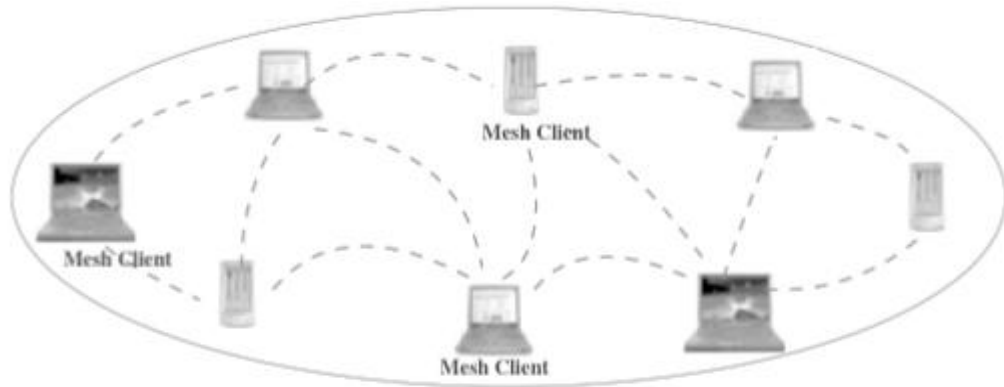


Figure 3: Client WMNs (Akyildiz, Ian F., & Xudong Wang 2009).

- Hybrid WMNs: The hybrid WMNs architecture is the combination of infrastructure and client meshing as illustrated in Figure 4. Mesh clients get access to the network through the mesh routers and through direct communication with other mesh clients. The routing capabilities of clients leads to improved connectivity and coverage inside the WMN, and the infrastructure makes connection to other networks such as the Internet, Wi-Fi, WiMAX, cellular, and sensor networks possible (Akyildiz & Wang 2009).

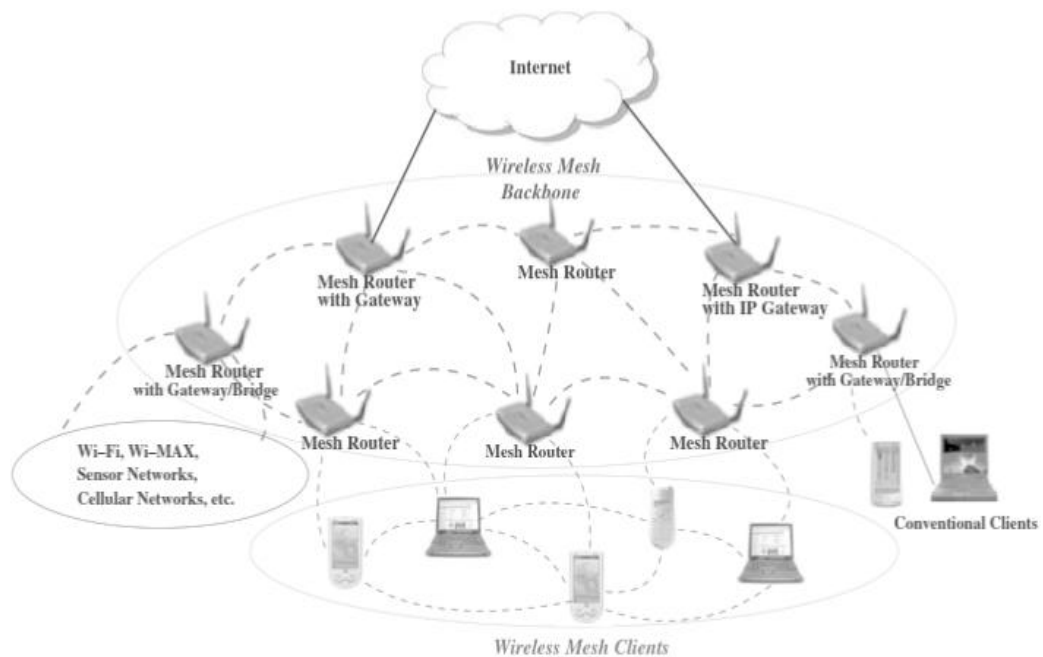


Figure 4: Hybrid WMNs (Akyildiz, Ian F., & Xudong Wang 2009).

1.3. Characteristics of Wireless Mesh Network

- I. Multi-hop wireless network: WMNs are developed to extend the coverage range of current wireless networks without sacrificing the channel capacity. It also provides non-line-of-sight (NLOS) connectivity among users without direct line-of-sight (LOS) links. This facilitates higher throughput with less interference between nodes, and more efficient frequency reuse.
- II. Support for ad hoc networking, and capability of self-forming, self-healing, and self-organization: WMNs does not require huge investments and the network can grow gradually as required. This is because Ad hoc networking enhances network performance, providing flexibility to the network architecture, it is easy to deploy and configure, tolerance to fault, and mesh connectivity (multipoint-to-multipoint communications).
- III. Mobility dependence on the type of mesh nodes: It is recommended that the mesh routers have minimal mobility, however, the mesh clients/end device could be either stationary or mobile nodes.
- IV. The type of mesh nodes determines the power-consumption constraints: The routers in WMNs are usually connected to electric mains and without restriction on power consumption. However, mesh clients and end devices often require power efficient protocols. For example, a sensor node may need to only become active when it wants to send data for power management and efficient. Therefore, the routing protocols enhanced for mesh routers are usually not appropriate for mesh clients, because in wireless sensor networks, power efficiency is the primary concern.
- V. Compatibility and interoperability with existing wireless networks: The WMNs built in line with the IEEE 802.11 technologies must be compatible with IEEE 802.11 standards regarding the support of mesh-capable and conventional Wi-Fi clients and there need to be interoperable with other wireless networks like cellular systems, WiMAX, and ZigBee (Akyildiz & Wang 2009).

1.4. General Application Scenarios

The general applications of WMNs are listed below. These applications cannot be supported using other wireless networks such as the cellular systems, wireless sensor networks, standard IEEE 802.11, ad hoc networks, etc., therefore, this influences the growth in the research and development of WMNs (Akyildiz & Wang 2009).

Broadband home networking: Broadband home networking is currently achieved through IEEE 802.11 WLANs. The problem that arises from this is the location of the access points. A site survey of the home usually shows many dead zones without service coverage. To solve this, is often expensive and not practical for home networking and the installation of multiple access points results to increase in cost. With WMNs dead zones are eliminated this is done by adding mesh routers, changing locations of mesh routers, or adjusting power levels of mesh routers automatically. Communication within home networks can be achieved using mesh networking without having to go back to the access hub all the time.

Enterprise networking: This can be a small network within an office or a medium-size network for all the offices within an entire building, or a large-scale network for offices in multiple buildings.

Metropolitan area networks (MAN): The wireless mesh MAN covers a potentially much larger area than home, enterprise, or building networks. Therefore, the requirement on the network scalability by wireless mesh MANs is much higher than that by other applications.

Transportation systems: IEEE 802.11 or 802.16 has limited access to stations and stops. If we make use of mesh networking technology, the access can be extended into buses, ferries, and trains. Thereby, making convenient passenger information services, remote monitoring of in-vehicle security video, and driver communications to be possible. However, two key techniques are required to enable such mesh networking for a transportation system, these are the highspeed mobile backhaul from a vehicle (that is a car, bus, or train) to the Internet, and mobile mesh networks within the vehicle.

Building automation: The various electrical devices such as power, light, elevator, air conditioner, etc., in a building need to be controlled and monitored. The current installations have

this task being achieved through standard wired networks, which is very expensive because of the complexity in deployment and maintenance of a wired network. If these wired networks access points are replaced by mesh routers, the deployment cost will be significantly reduced. The deployment process will be also much simpler due to the mesh connectivity among wireless routers.

Health and medical systems: The monitoring and diagnosis data in a hospital or medical center need to be processed and transmitted from one location to another for several purposes. The traditional wired networks can provide limited network access to certain fixed medical devices. While Wi-Fi-based networks relies on the existence of Ethernet connections, which may result to high cost to build the system and complexity and does not have the abilities to eliminate dead spots. However, these issues are not applicable to WMNs.

Security surveillance systems: Security is a major, therefore, security surveillance systems are necessary for application in enterprise buildings, shopping malls, and grocery stores, etc. WMNs are a much better solution for effective deployment of the security systems at locations as there are needed compared to wired networks in order to achieve connection of all devices. This type of application demands much higher network capacity than other applications since its data consist of picture and videos as the major components of the traffic flowing in the network.

1.5. Fundamental Design Factors

There are some important design factors to consider before a network is designed, deployed, and operated. These factors critically influence the performance of the network. For WMNs, the critical factors are summarized as follows.

Radio techniques: There has been a significant revolution of the wireless radio as a result of the rapid progress of semiconductor, RF technologies, and communication theory. Presently, MIMO is one of the key technologies for IEEE 802.11n which is the high-speed extension of Wi-Fi. More advanced radio technologies such as reconfigurable radios, frequency agile/cognitive radios, and software radios have been used in wireless communication. The type of

radio technique used is based on the need to improve the wireless communication capability and these has led to further developments of these radio technologies.

Scalability: Multi-hop communication is a common feature of WMNs, and it has been observed that communication protocols using it suffer from scalability issues meaning that when the size of network increases, the network performance degrades significantly. This implies that routing protocols may not be able to find a reliable routing path, transport protocols may lose connections, and MAC protocols may experience significant throughput reduction. Designing a hybrid multiple access scheme using CSMA/CA and TDMA or CDMA can improve the scalability of WMNs. This could be an interesting and challenging research issue.

Mesh connectivity: Mesh connectivity brings about several advantages of the WMNs and it is a critical requirement for MAC and routing protocols design. Network self-organization and topology control algorithms are generally needed. This can significantly improve the performance of WMNs.

QoS: The different factors that determine the quality of service must be considered in a communication protocol and WMNs seems to be addressing these issues like end-to-end transmission delay and fairness, more performance metrics such as delay jitter, aggregate and per-node throughput, and packet loss ratios.

Compatibility and interoperability: WMNs need to be backward compatible with conventional client nodes. This will encourage the deployment of WMNs. For the WMNs to be successfully integrated with other wireless networks, the mesh routers must be capable of interoperating among heterogeneous wireless networks.

Security: The security of WMNs must be convincing and reliable for the WMNs to succeed because customers would only subscribe to reliable services. There have been many security schemes proposed for wireless LANs, but none is ready for WMNs. For example, there is no centralized trusted authority to distribute a public key in a WMN owing to its distributed system architecture. However, the existing security schemes for ad hoc networks could be used for WMNs, but with some limitations as most security solutions for ad hoc networks are still not developed enough to be practically implemented. Furthermore, the network architecture

of WMNs is different from a conventional ad hoc network, as a result, differences in security mechanisms must exist. Hence, there must be development of new security schemes such as encryption algorithms, security key distribution, secure MAC and routing protocols, intrusion detection, and security monitoring.

Ease of use: The design of protocols must be done in such a way that the network becomes autonomic as much as possible. This implies automatic power management, robust to temporary link failure, self-organization, dynamic topology control, and fast network-subscription/user authentication procedure. Furthermore, to efficiently maintain the operation, monitor the performance, and configure the parameters of WMNs, network management tools need to be developed.

2. ZIGBEE PROTOCOL AND COMMUNICATION

ZigBee protocol supports 3 nodes types namely ZigBee Coordinator ZC, ZigBee Router (ZR) and ZigBee End Device (ZED). The ZC initiates the network, protects it and generates the control functions needed.

Coordinator - In a ZigBee network, only one coordinator device is required. The coordinator radio is responsible for creating the network, addresses handling, and managing the other functions that define the network, secure it, and keep it healthy. This implies that each network must be formed by a coordinator and you never require more than one coordinator in your network (DIGI International 2018a).

Router - A router is a full-featured ZigBee node that can join existing networks, send information, receive information, and route information. Routing implies that it forwards a message to enable communications between other devices that are too far apart to receive information directly on their own. It is essential to plug the routers to an electrical outlet because they must always be turned on all the time. A typical network usually has multiple router radios (DIGI International 2018a).

End device - End devices are used in situations where the hardware and full-time power of a router are excessive for what a radio node needs to do. Their main function is to join networks and send and receive information. They cannot route messages to any other devices, therefore, they can use less expensive hardware and can power themselves down intermittently, when its inactive thereby, saving energy by going temporarily into a nonresponsive sleep mode. However, end devices always require a router or the coordinator as a parent device which helps end devices join the network, and stores messages for them when they go to asleep. ZigBee networks can have any number of end devices. The network can comprise of one coordinator, multiple end devices, and no routers at all (DIGI International 2018a).

2.1. Implementing ZigBee Wireless Mesh Network

In ZigBee Wireless Mesh Network, the Zigbee protocols are developed for embedded applications requiring low power consumption and can tolerate low data rates. To pass the Zigbee certification, the network must make use of very little power and each device is required to have a battery life of minimum two years. ZigBee can be used to develop a wireless mesh network. In the mesh network, all the nodes are interconnected creating multiple pathways from one node to another. The connections are dynamically updated and optimized using the sophisticated, built-in mesh routing table (DIGI International 2019).

ZigBee is an IEEE 802.15.4 based specification for high-level communication protocols used to create personal area networks with low-power digital radios. The following are the major features of XBee radio devices:

- They work on 2.4 GHz (Unlicensed Radio Band) radio frequency
- Low data rate ($\approx 250\text{Kbps}$)
- Low power consumption (1mW, 6mW, 250mW etc.)
- Wireless communication over short distances (90m, 750m, 1mile etc.)

Therefore, typical application areas include:

- Home automation
- Wireless sensor networks
- Industrial control systems
- Embedded sensing
- Medical data collection
- Smoke and intruder warning
- Building automation
- Remote wireless microphone configuration

Zigbee is not suitable for applications that require high mobility among nodes. Therefore, it is not the best for application in tactical ad hoc radio networks on the battlefield, because high data rate and high mobility are needed.

2.2. System Architecture

The system is made up of the following hardware components Multiprotrotocol Radio Shield, Arduino development board, XBee S2 module, XBee Explorer boards, XCTU tool, LCD Screen (compatible with Hitachi HD44780 driver), LDR (Light Dependent Resistor), Resistor (100k; 220-ohm;330ohm) and LED (Light Emitting Diode). The hardware setup of the system is illustrated in figure 5.

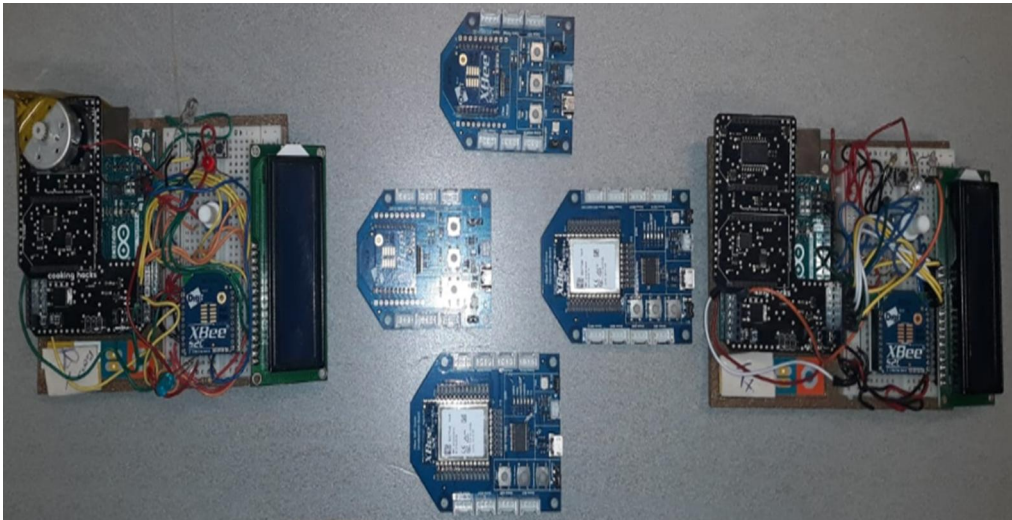


Figure 5: Implementation Setup Block Diagram.

2.3. System Configuration

The ZigBee Mesh Kit (Part Number: XKB2-Z7T-WZM) from www.digi.com is used in this project. The XBee radios Series 2 model (note that a series 1 model only allows communication between a maximum of 2 modules) is used. The kit comes with 3 XBee radios included. In terms of expansion, the system is modular making it possible to add more radios as needed. The second part involves the use of Arduino development board to implement the ZigBee Network using the XBees RF modules. The program XCTU is used to configure the XBee radios to create the ZigBee Mesh Network. Once the XCTU program is installed, next step is to plug one of your XBee radios into one of the explorer boards provided in the mesh kit (as in figure 6), then connect the explorer board to your PC with the included cable and you can configure the radio.



Figure 6: Xbee radio plug into explorer board.

2.3.1. Configuration Modes for Xbee Modules

XBee devices use either AT (transparent) or API operating mode to send and receive data over the serial interface. The network could also have a mixture of devices using either AT or API mode.

AT Mode – The default configuration of XBee devices is the AT (transparent) mode. In the AT mode, all the data is received through the serial input and queued up for radio transmission. The data received wirelessly is sent to the serial output exactly as it is received, without additional information. When a device operates in Transparent mode, it cannot identify the source of a wireless message it receives. In order to identify the sources of all the data it receives from different devices, the devices sending the message must include extra identifier information known by all the devices which can be extracted later. This can be achieved by defining a robust protocol that comprises of all the information needed for the transmissions (DIGI International 2018b).

API Mode – API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between devices without having to define your own protocol. Since the data destination is included as part of the API frame structure, you can use API mode to transmit messages to multiple devices. The API frame includes the source of the message, so it is easy to identify where data is coming from (DIGI International 2018b).

2.3.2. X-CTU software

The configurations of the XBee are done using X-CTU software provided by Digi International. For best results on this project, the firmware on XBee was updated to DigiMesh 2.4 TH (to achieve this, click the "update" icon above the "system parameters" section in the XCTU software with the XBee module plug to the PC). The most important parameters are the first two parameters (Channel and Network ID). These parameters must be the same for all modules in your system. They can be set to any value and these values are the frequency with which your XBees will communicate. This is shown in figure 7. All the XBee module (Coordinator, Routers and End Device) are programmed with the "Destination Address" FFFF to allow broadcast of data to all the XBee modules on the network.

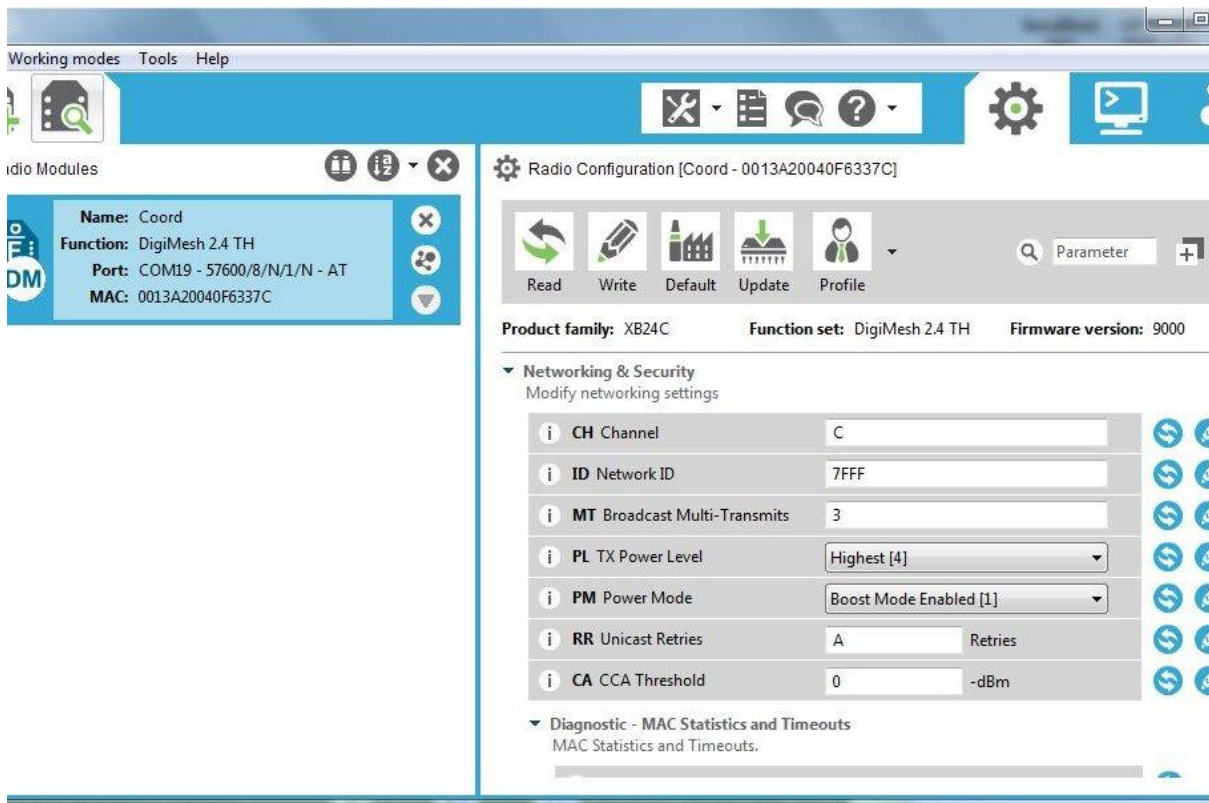


Figure 7: Channel and Network ID setup.

2.3.3. XBee Configuration for Coordinator and End Device

Simply select the "Coordinator" Option from the CE Coordinator/End Device line as illustrated in figure 8. Then follow the following steps shown in figure 9 to 10.

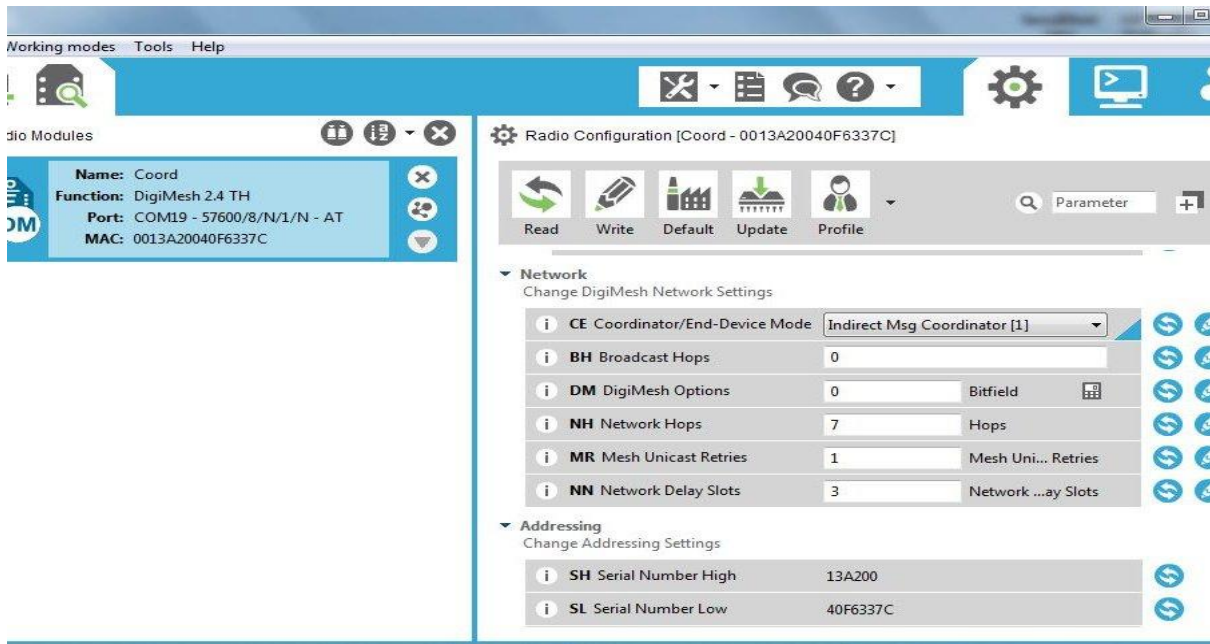


Figure 8: Configure as Coordinator.

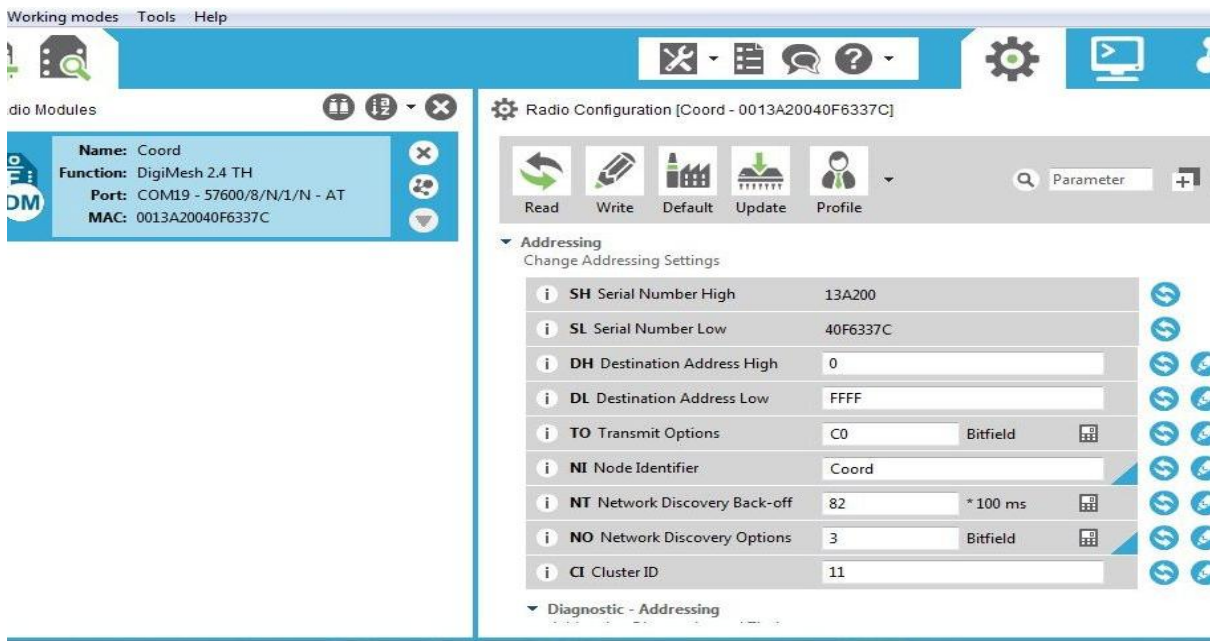


Figure 9: Coordinator Destination Address and Device Name setup.

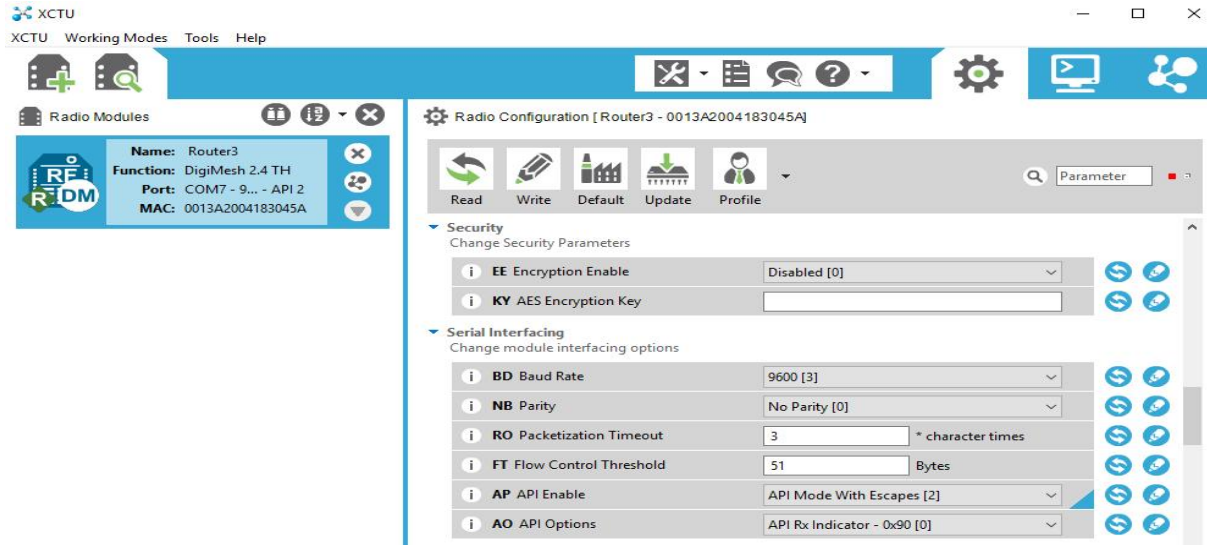


Figure 10: Encryption and Configuration Mode (API 2) setup.

2.3.4. XBee Configuration for Routers

The XBee can be configured as a router. To do this, perform the discovery actions for the router module as was done for the coordinator and set the parameters of the router in an exact way as the coordinator. The exception in the parameters for the Router is illustrated in the following figures 11 and 12. Simply select the "Standard Router" Option from the CE Coordinator/End Device line as illustrated in figure 11.

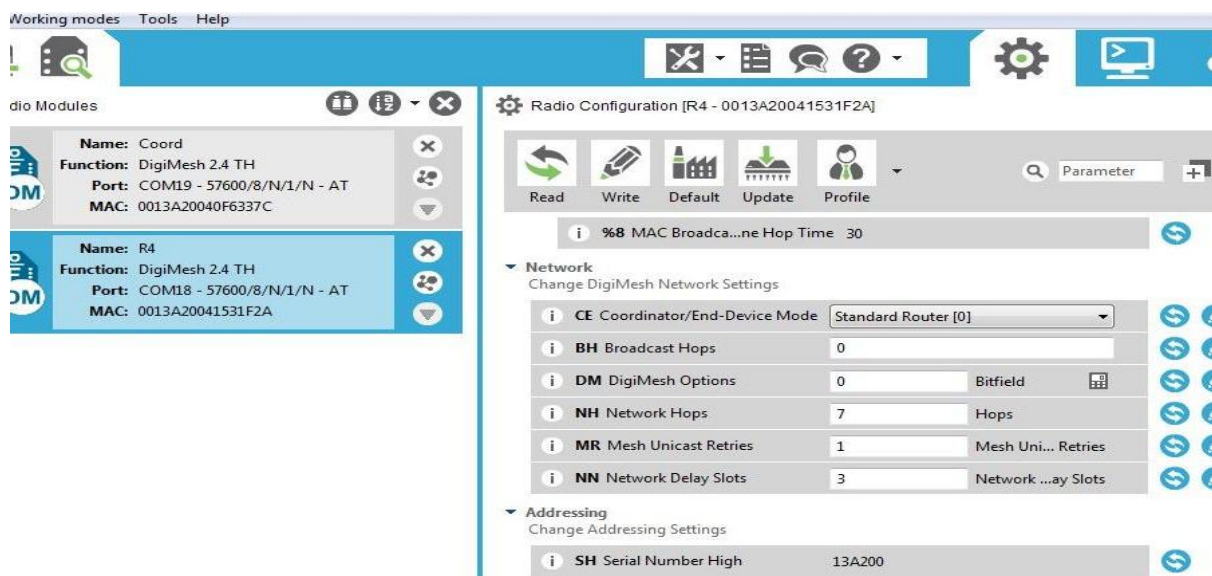


Figure 11: Configure as Router.

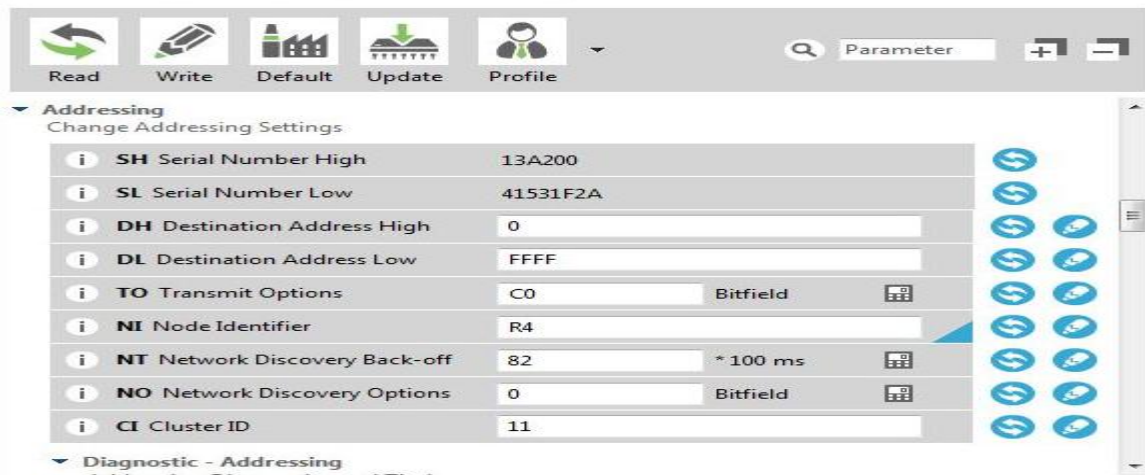


Figure 12: Router Destination Address and Device Name setup.

2.3.5. XBee Configuration for End-Device

The XBee can be configured as an End Device. To do this, perform the discovery actions for the End device module as was done for the coordinator and set the parameters of the router in an exact way as the coordinator. The exception in the parameters for the End device is that you select the "End Device" Option from the CE Coordinator/End Device line as illustrated in figure 13.

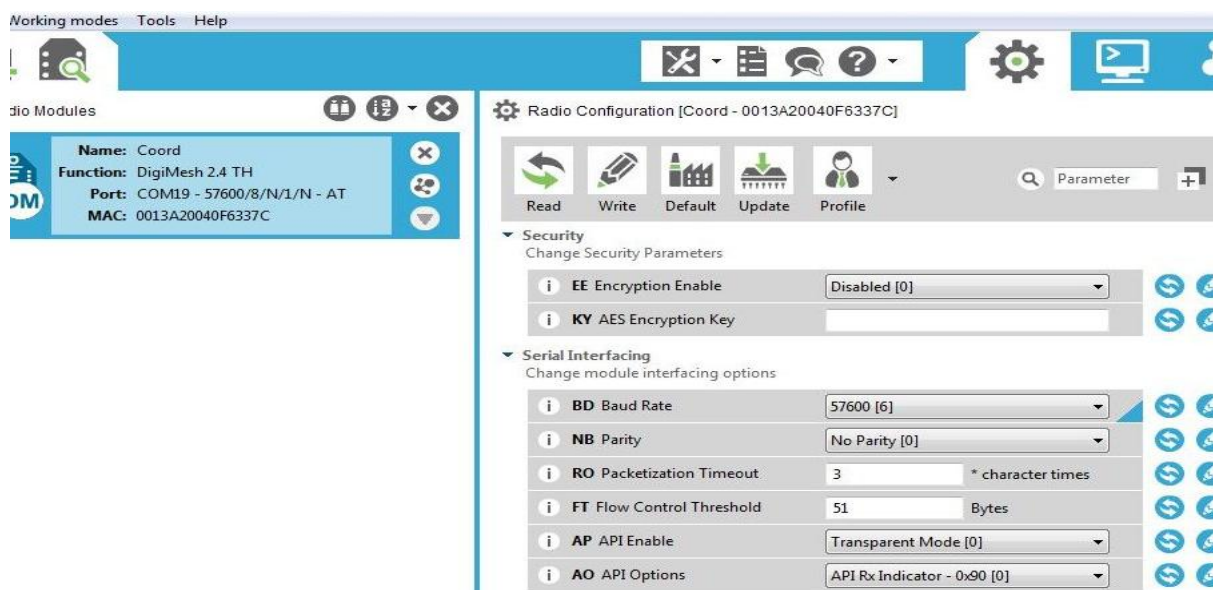


Figure 13: Encryption and Configuration Mode (AT) setup.

3. IMPLEMENTATION AND RESULTS

The block diagram for the complete “coordinator” and “end-device” hardware implementation is illustrated in APPENDIX 2. In this section, the main interface discussed is the XBee-Arduino interface because this is the main implementation responsible for achieving the wireless mesh network. Other interfaces implemented includes the LCD-Arduino interface and LDR-Arduino interface illustrated in figure 15 and 16 respectively.

3.1. Implementing XBee-Arduino, LCD-Arduino and LDR-Arduino Interfaces

The Arduino-XBee S2 TX/RX connection on breadboard is seen in figure 15. The schematic view is illustrated in APPENDIX 3.

Coordinator and End-device – The coordinator and end-device XBee configured modules are implemented on the Arduino development board. The Coordinator is implemented as a receiver, while the end-device is implemented as a transmitter to which a sensor (in our case, the smart NOx) is connected. The Routers are mounted on the XBee explorer board.

In total we have six (6) XBee Radios, one XBee is configured as a Coordinator in API mode with API enable set to 2 (it will use escape characters along with the data) and interfaced with the Arduino as seen in figure 14. Another XBee device is configured as an End-Device in AT mode (Transparent mode) and interfaced with the Arduino as seen in figure 14. SoftwareSerial of Arduino is used for communication with XBee. Serial of Arduino is used to display the received data on the serial monitor.

The Routers are configured in API mode with API enable set to 2 (it will use escape characters along with the data). The same API enable setting must be used for all the XBee devices configured in API mode. The Routers are left mounted on the XBee explorer board as illustrated in figure 6.

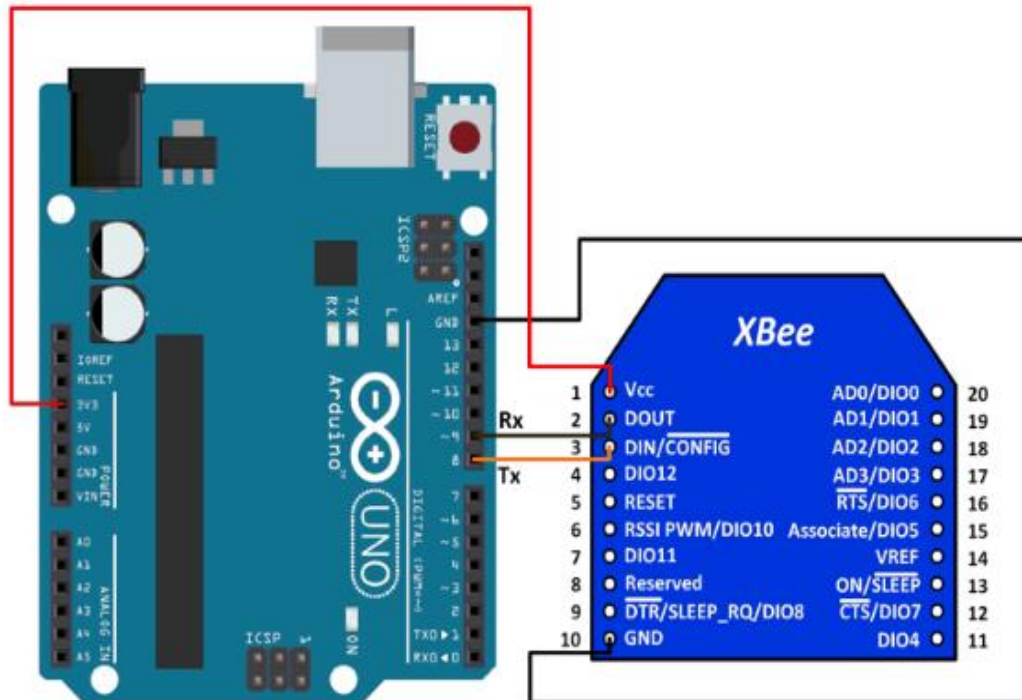


Figure 14: XBee-Arduino Interface (ElectronicWings 2019).

To wire your LCD screen to your board, connect the following pins:

- LCD RS pin to digital pin 9
- LCD Enable pin to digital pin 8
- LCD D4 pin to digital pin 7
- LCD D5 pin to digital pin 6
- LCD D6 pin to digital pin 5
- LCD D7 pin to digital pin 4

Furthermore, wire a 10k pot to +5V and GND, with its output to LCD screens VO pin (pin3). A 220ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector.

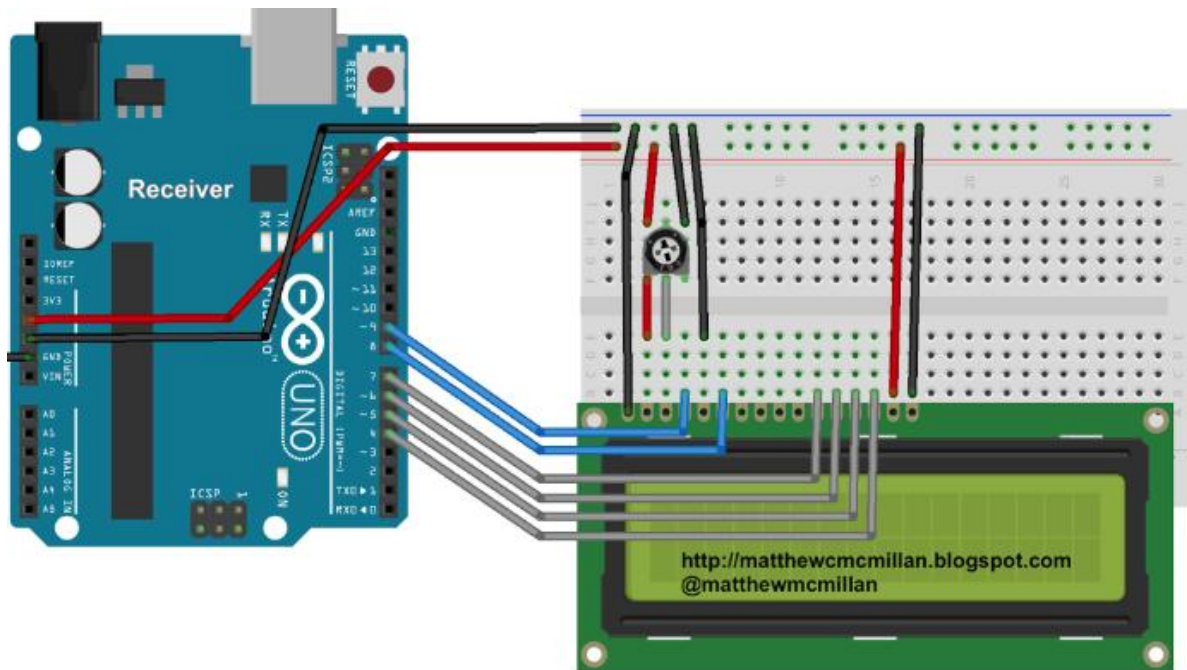


Figure 15: LCD-Arduino interface (Matthew McMillan 2019).

Light Dependent Resistors (LDRs) are made from semiconductor materials which gives them their light-sensitive properties. There are many types, the most common material used is the cadmium sulfide (CdS). LDRs or Photo Resistors works on the "Photo Conductivity" principle. This means that whenever light falls on the surface of the LDR, the conductance of the element increases or in other words, the resistance of the LDR falls when the light falls on the surface of the LDR. From the circuit diagram in figure 16, a voltage divider circuit is created using LDR and 100k resistor. The voltage divider output is feed to the analog pin of the Arduino. The analog Pin senses the voltage and gives some analog value (which changes according to the resistance of LDR) to Arduino. Therefore, when the light intensity on the LDR increases, the resistance of the LDR decreases and hence the voltage value of the analog pin increase and the light goes OFF. If the light intensity on the LDR decreases, the resistance of the LDR increases and hence the voltage value of the analog pin decreases and the light comes ON. In the Arduino code in APPENDIX 6, when the analog value falls below 70, we consider it as dark and the light turns ON. However, If the value comes above 70, we consider it as bright and the light turns OFF. The LDR-Arduino interface is implemented only on the Sensor Node (End Device side).

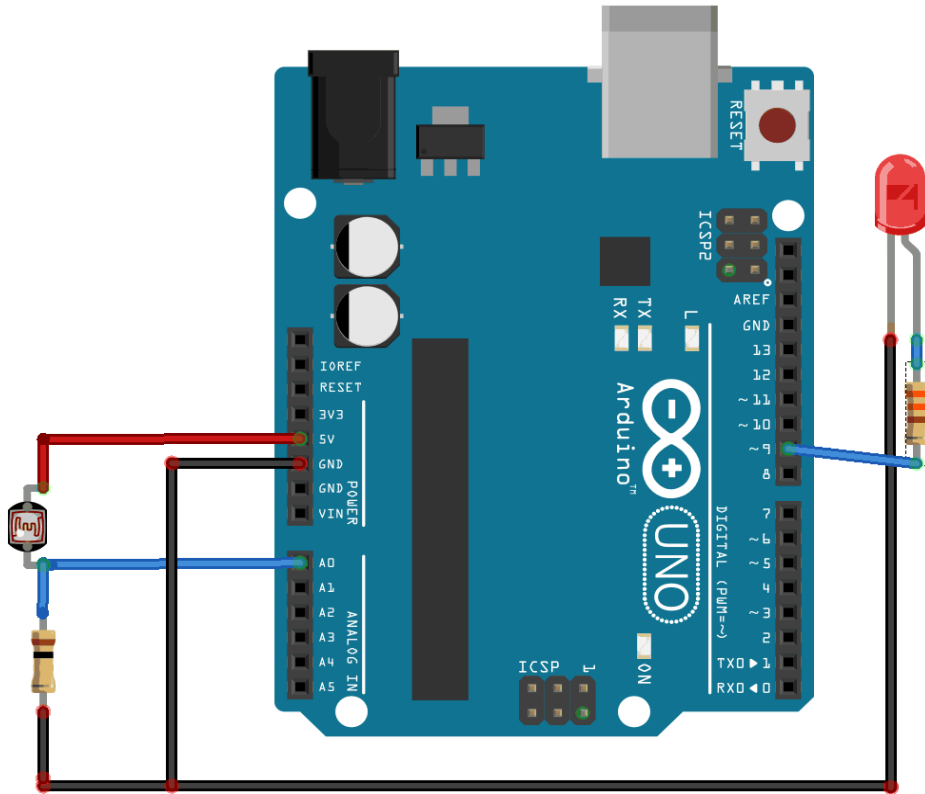


Figure 16: LDR-Arduino interface (Pankaj Khatri 2018).

3.2. Programming the XBee-Arduino Interface

Arduino is an open-source microcontroller system which has been designed for easy learning and fast development. It is easy to use and flexible. Arduino can interact with the environment through data received from a variety of sensors and can be used to control its surroundings by controlling lights, motors, and other actuators. The Arduino microcontroller is programmed using the Arduino programming language, which is based on Wiring, and the Arduino development environment which is based on Processing.

The Arduino language is based on C/C++ and therefore, has a similar specific set of structures which have been made simple for people who are new to programming. A simple Arduino program is illustrated below:

```
// variable definitions always come first  
  
// The setup () method runs once, when the sketch starts  
  
void setup()  
{  
  lcd.begin(16, 2);  
  CANconfig();  
  XBeeConfigSend();  
  Serial.begin(9600);  
  xbee_ss.begin(9600); /* Define baud rate for software serial communication */  
  xbee.setSerial(xbee_ss);  
}  
  
// The loop () method runs continuously, if the Arduino is powered ON.  
  
void loop()  
{  
  SendHeatCANMsg();  
  receiveCANDataSend();  
  xbee.send(zbTx);  
  delay(10);  
}
```

The code is divided into two basic parts namely setup and loop. They are executed in a sequential order with the setup being the first part of the code that is run only once on initialization of the code. This is the part of the code where it is recommended to include the initialization of the modules which are to be used. The loop part of the code runs continuously, in an infinite loop. This is where the main part of the code to perform the desired function is included. (Tutorialspoint 2018.)

3.2.1. Programming XBee Coordinator-Arduino Interface as Receiver

The programming was done on the Arduino IDE environment as seen in APENDIX 4. The XBee Coordinator-Arduino Interface is programmed as the receiver. It listens to incoming data from all the routers in the same network (that is, having the same Channel and Network ID) as the coordinator.

The programming flowchart is illustrated in figure 17.

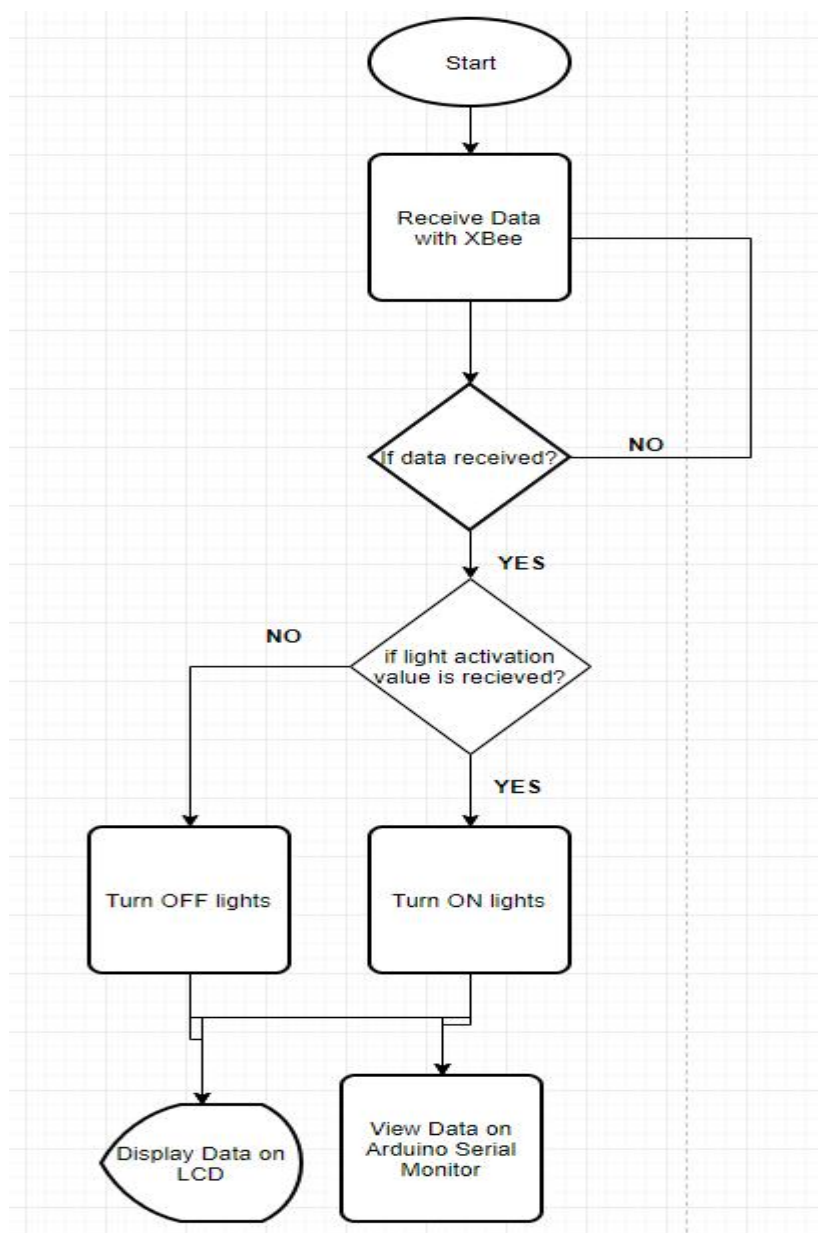


Figure 17: XBee Coordinator-Arduino Flowchart.

3.2.2. Programming XBee End-Device-Arduino Interface as Transmitter

The programming was done on the Arduino IDE environment as seen in APENDIX 4. The XBee End-Device-Arduino Interface is programmed as the transmitter connected to a sensor. It extracts data from the sensor and transmits the data through the routers in the same network (that is, having the same Channel and Network ID) to the coordinator. The path it takes depends on how close the coordinator is to the End Device. This implies that it could transmit the data directly to the coordinator if it is close enough to the coordinator.

The programming flowchart is illustrated in figure 18.

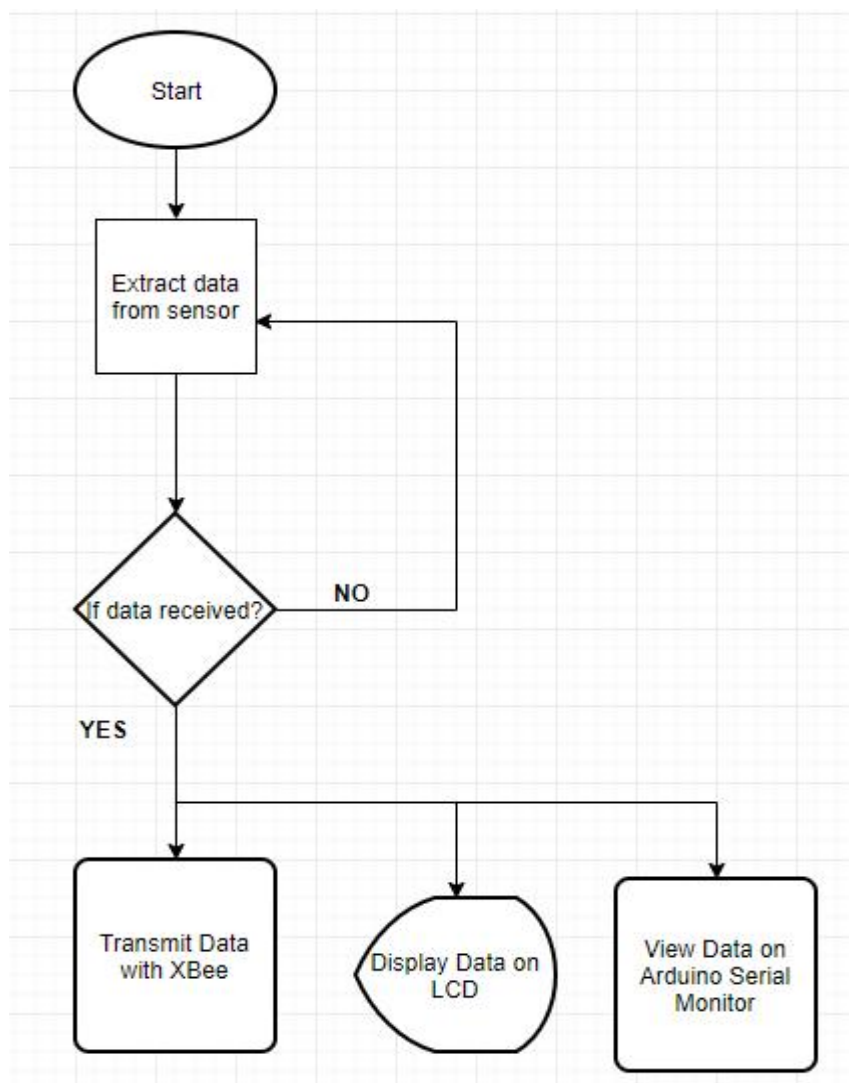


Figure 18: XBee End-Device-Arduino flowchart.

3.3. Project Outcome and Results

This chapter gives the result of the test of an implemented ZigBee wireless mesh network. The wireless mesh network layout is illustrated in figure 19. From simulations, the best maximum distance between the Sensor Node (End Device) and the router for the router to receive and forward the transmitted data was approximately 25 meters. The best maximum distance between any of the router to the coordinator for the coordinator to be able to receive the data forwarded by the router was also approximately 25 meters. Therefore, the total maximum distance between the sensor node and the coordinator having one router in between was approximately 50 meters. As the number of routers between the sensor node and coordinator increases, the distance will increase approximately by +25 meters.

The layout of the routers takes the mesh topology, we have a total of four routers in the mesh network. This implies that failure in one of the routers, for example, Router 2 will not affect the network as the data will be routed through the other routers in the network. In our case study, one of the routers, Router 3, was also configured to transmit data to the coordinator, serving as both a sensor node and a router. This was done because of the limited number of XBee devices, however, a separate XBee device could be configured as an End device and placed close to the Router 3.

The test of the mesh network communication is further discussed in section 3.3.1. The code for the implementation of the End Device (Sensor Node) and Coordinator (Receiver) are illustrated in APPENDIX 5 and APPENDIX 6 respectively. The codes were implemented in the Arduino development environment. The simulation in the project involved using data readings from an LDR and generating data from the XCTU software as the sources of the data being transmitted in the mesh network. The LDR data was used to remotely control (turn ON and OFF) LED lights and a motor (Fan) connected to the coordinator/receiver in the mesh network.

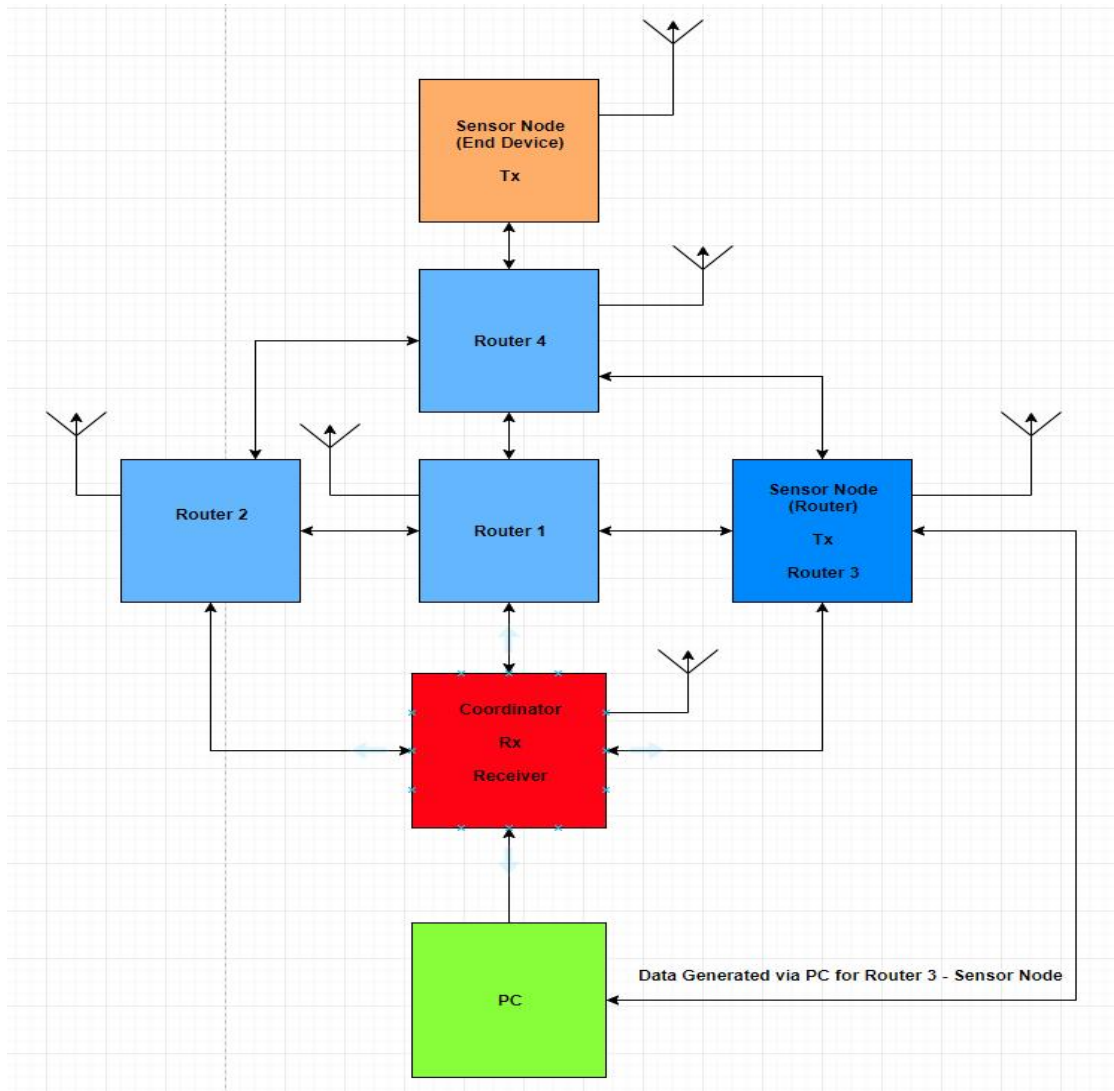


Figure 19: Wireless Mesh Network Layout.

3.3.1. Testing Communications

The test is done to ensure that communication has been achieved. When a message is sent by the End Device the message is received by the Coordinator. Also, the message "hello world" sent from the Router 3 was also received by the coordinator in hexadecimal format as illustrated in figure 20 which is the Arduino Serial monitor interface. Figure 21 is the End Device (sensor node) Arduino Serial monitor interface showing the data from the sensor.

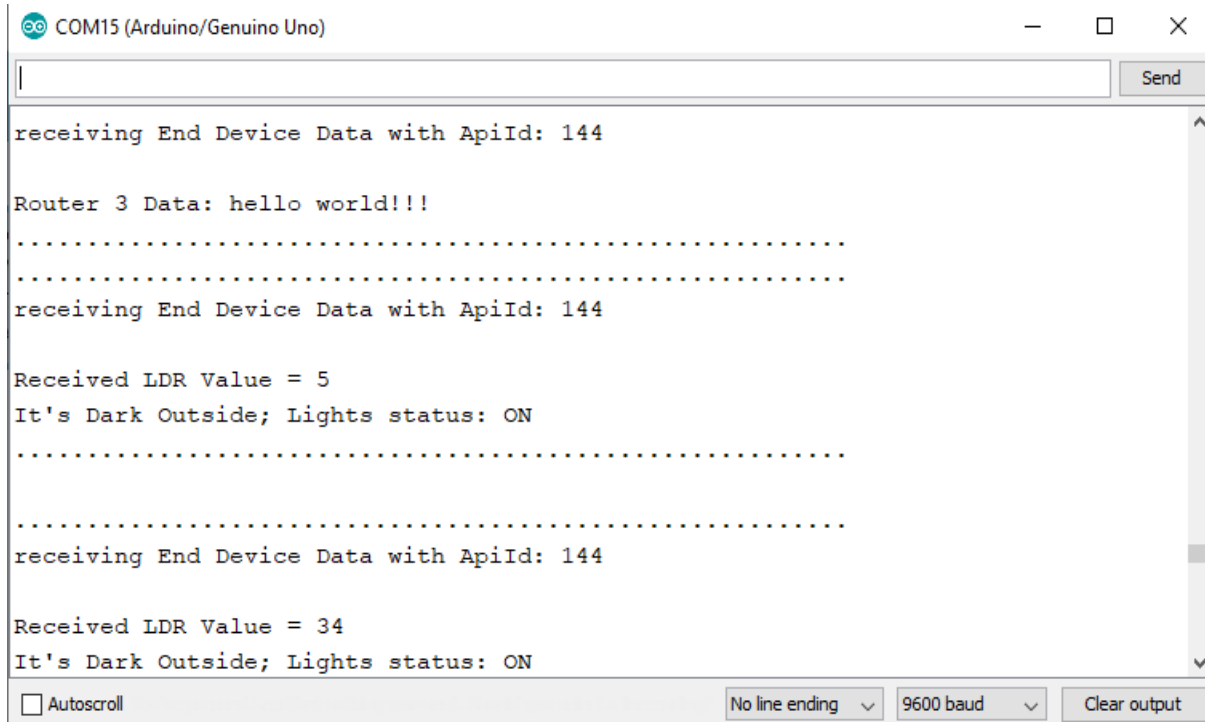


Figure 20: Arduino Serial monitor interface for Coordinator (Receiver).

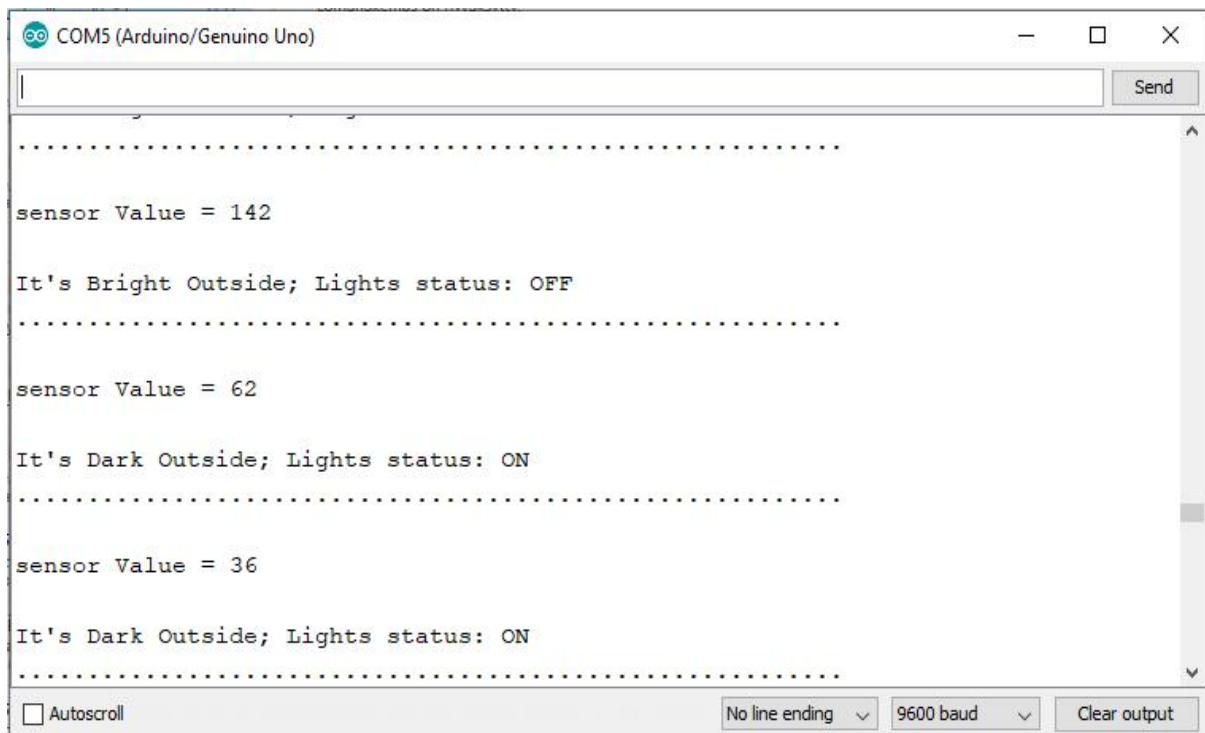


Figure 21: Arduino Serial monitor interface for End Device (Transmitter).

Since the Router 3 is connected to the PC in order to generate and send the data using the XCTU software, you can view the generated data and also view the data being received from the “End Device Transmitter” and routed by the Router 3 to the coordinator as seen in figure 22 and 23 respectively.

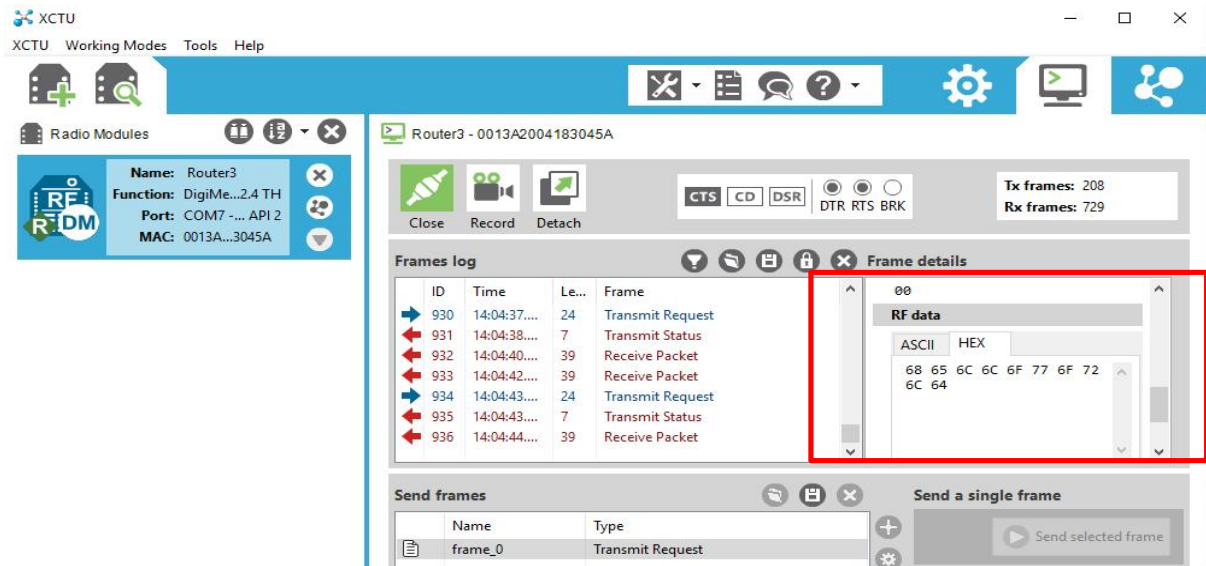


Figure 22: View of the generated data in XCTU for Router 3.

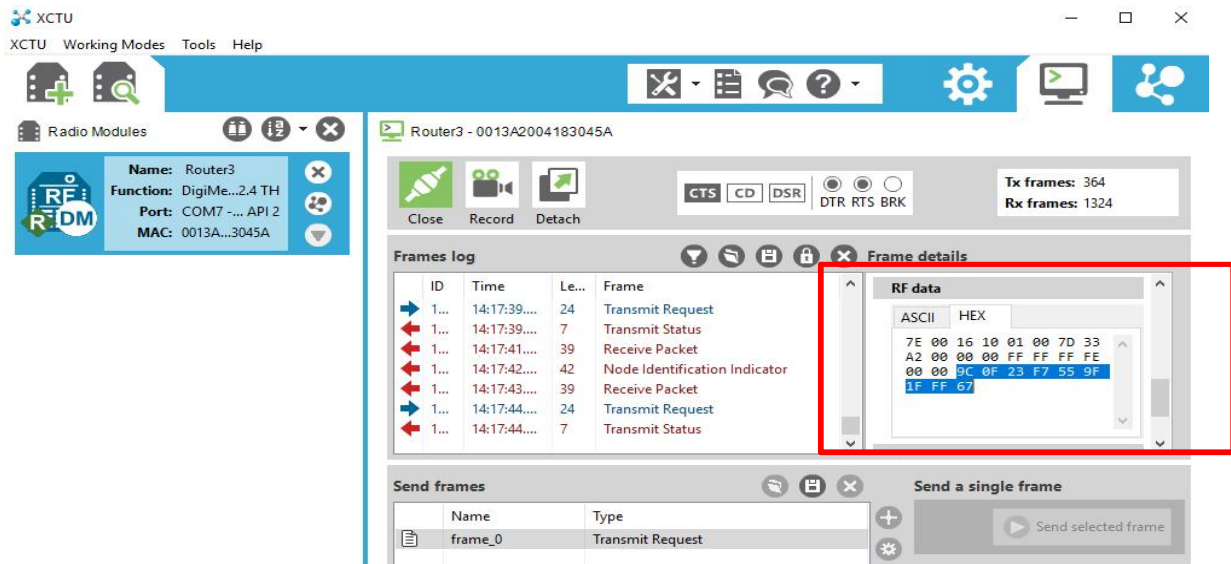


Figure 23: View of the data being received from the “End Device (Sensor Node)”.

Figure 24 and 25 are the Coordinator's LCD display view of the received data from the Router 3 and End Device (Sensor Node – for the LDR or smart NOx sensor) respectively.

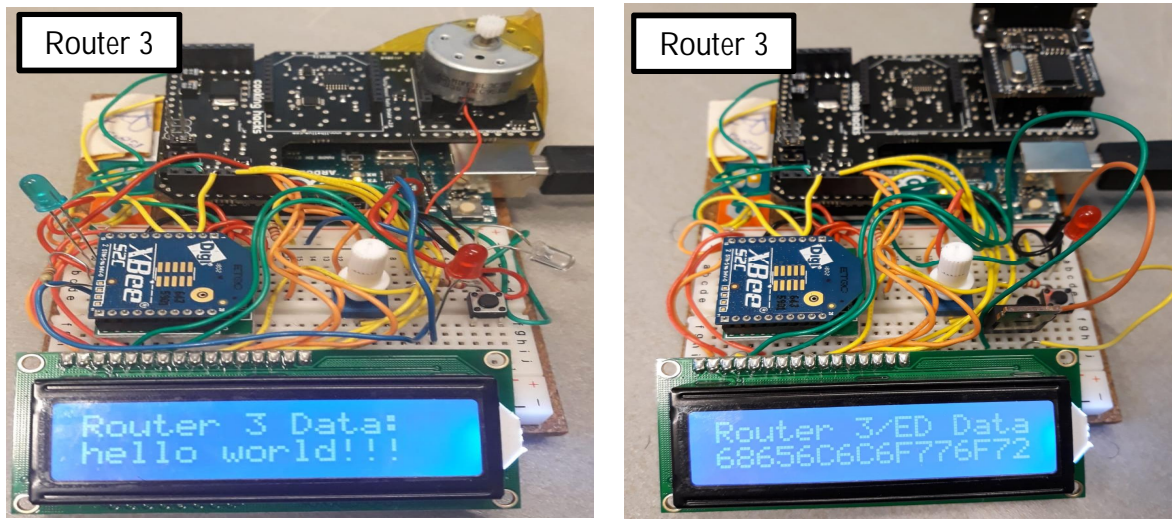


Figure 24: Coordinator (Receiver) LCD display of received data from Router 3.

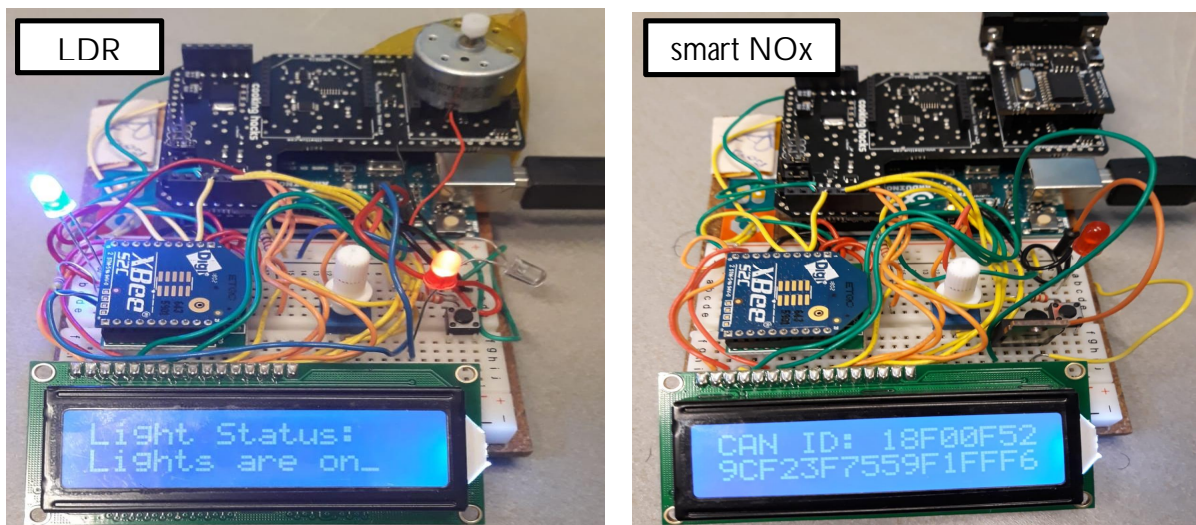


Figure 25: Coordinator (Receiver) LCD display of received data from End Device (Sensor Node).

3.3.2. RSSI and Wireless Mesh Network

The XBee Series 2 modules make use of the ZigBee mesh protocol. Therefore, they do not include the RSSI information in the packet. Note that the RSSI is only good for the last hop or node. Therefore, you can not measure practically the RSSI of the entire mesh network.

4. CONCLUSION

The aim of the project was to implement and test a ZigBee wireless mesh network as a proof-of-concept for Wireless Mesh Networks. The wireless mesh network in this project is implemented using an XBee mesh network that can send and receive data in a self-healing system. There are six nodes in this network and the communication amongst all the nodes works as expected. When using the broadcast mode, the other nodes in the network can receive data from the sending node. When all the nodes in the network have the same DL address and the DL address is equal to the MY address of the transmitter node (set to broadcast mode – FFFF), all nodes would be capable of transmitting data to the broadcast node. The mesh network can be expended by adding as many routers as required to suit your application.

The communication in a wireless network involves several nodes which provide a mesh network with a wider coverage area and reliable self-configuring, self-healing system. This is because when one of the nodes in the network fails, the communication can continue because the other nodes in the network would be used for communication. This reduces the possibility of interruption, service time-outs, and failures in wireless communication, thereby, improving the reliability of the network. In summary, the simulation involved using data readings from an LDR and generating data from the XCTU software as the sources of the data being transmitted in the mesh network. The LDR data was used to remotely control (turn ON and OFF) LED lights and a motor (Fan) connected to the coordinator/receiver in the mesh network.

There are several aspects for future work on this project. These include building a mesh network with more nodes and connecting the node to more sensors to investigate the behavior of the network when it is active in sending and receiving data from various Sensor Nodes in the network. The project could also be extended to focus on a specific application, for example, in wireless sensor or home automation as a case study. These sensor values can be uploaded to the cloud for remote monitoring and control. Instead of controlling an LED based on the light intensity outside, we can control our home lights or any electrical equipment. This can be achieved by connecting a relay module to the system and set the parameter needed to turn ON and OFF any AC appliance we choose.

LIST OF REFERENCES

Akyildiz, Ian F., and Xudong Wang (2009). *Wireless Mesh Networks*, John Wiley & Sons, Incorporated, 2009. ProQuest Ebook Central. Available online: <http://ebookcentral.proquest.com/lib/tritonia-ebooks/detail.action?docID=437489>.

DIGI International (2018a). *Device types*. Available online: https://www.digi.com/resources/documentation/Digidocs/90002002/Content/Concepts/c_device_types.htm?TocPath=zigbee%20networks%7Czigbee%20networking%20concepts%7C_____1.

DIGI International (2018b). *API mode in detail*. Available online: https://www.digi.com/resources/documentation/Digidocs/90001496/concepts/c_api_mode_detailed.htm?TocPath=XBee%20API%20mode%7C_____1.

DIGI International (2019). *Zigbee Wireless Mesh Networking*. Available online: <https://www.digi.com/resources/standards-and-technologies/zigbee-wireless-standard>.

ElectronicWings (2019). *XBee S2 (ZigBee) Interfacing with Arduino UNO*. Available online: <https://www.electronicwings.com/arduino/xbee-s2-zigbee-interfacing-with-arduino-uno>.

J. Jun, M.L. Sichitiu, "The Nominal Capacity of Wireless Mesh Networks", in IEEE Wireless Communications, Vol 10, 5 pp 8-14. October 2003.

Matthew McMillan (2019). *Arduino - Sending data over a CAN bus*. Available online: <http://matthewcmcmillan.blogspot.com/2013/10/arduino-sending-data-over-can-bus.html>.

Pankaj Khatri (2018). *Arduino Light Sensor Circuit using LDR*. circuitdigest 2019. Available online: <https://circuitdigest.com/microcontroller-projects/arduino-light-sensor-using-ldr>.

Robert, Faludi (2011). *Building Wireless Sensor Networks*. Available online: <https://ablog.ru/files/File/books/WirelessSensorNetwork.pdf>.

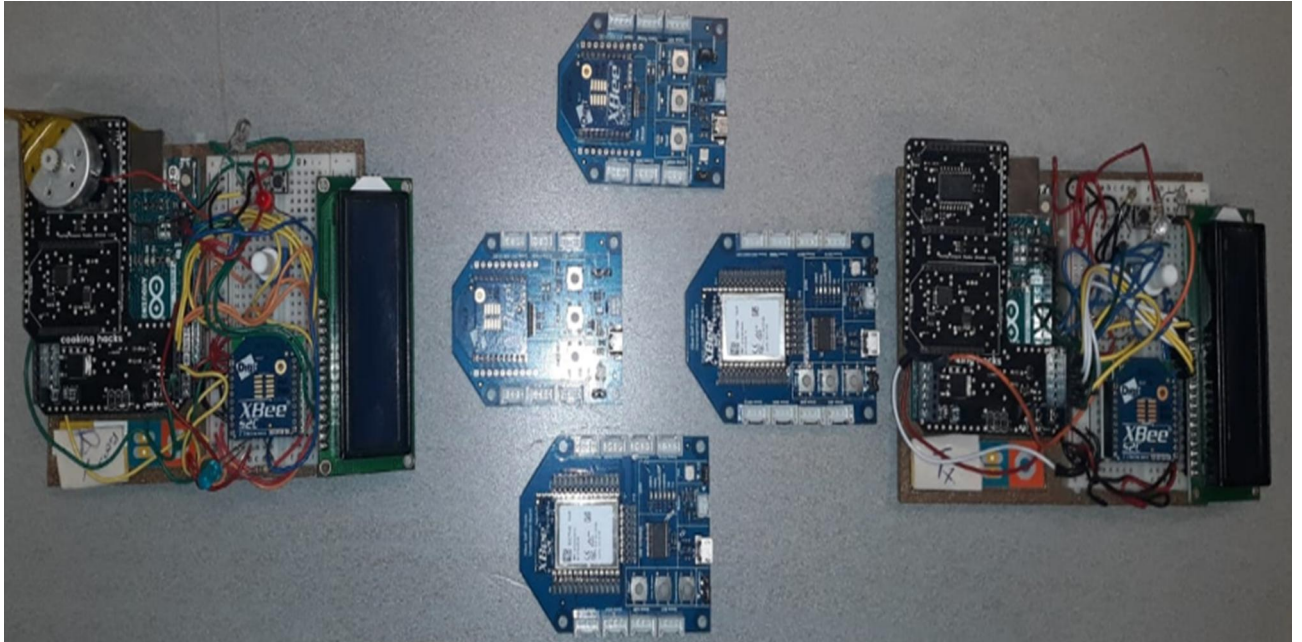
Tutorialspoint (2018). *Arduino - Program Structure*. Available online: https://www.tutorialspoint.com/arduino/arduino_program_structure.htm.

Wikipedia (2019a). *Wireless mesh network*. Available online: https://en.wikipedia.org/wiki/Wireless_mesh_network#cite_note-3.

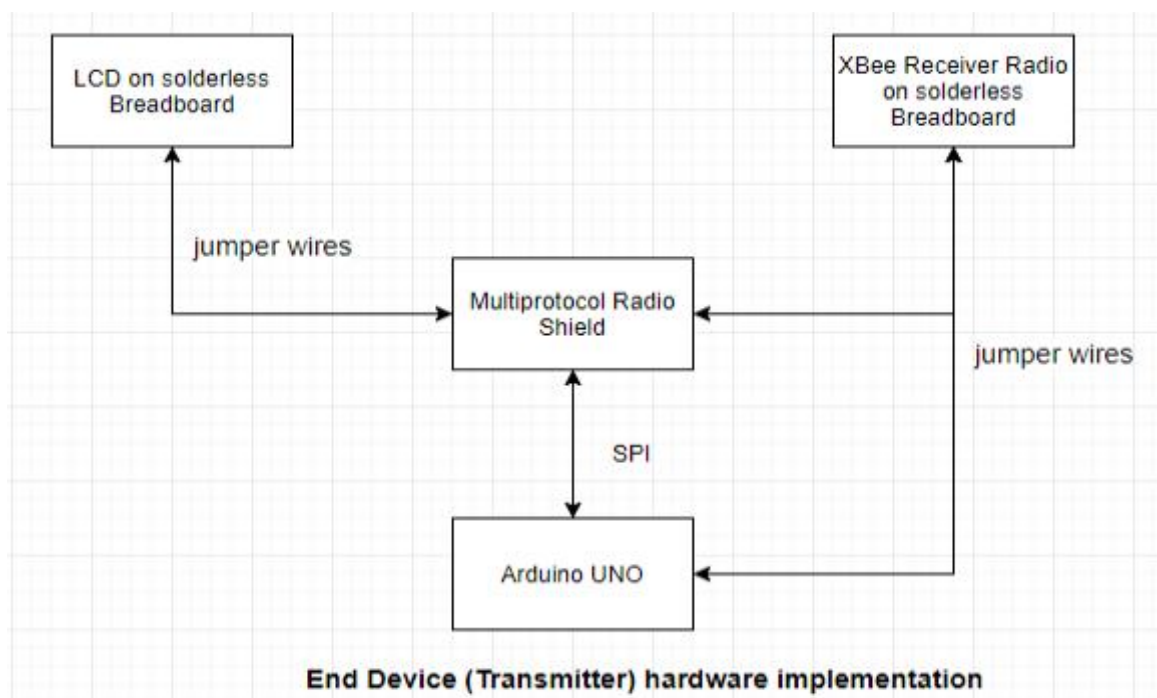
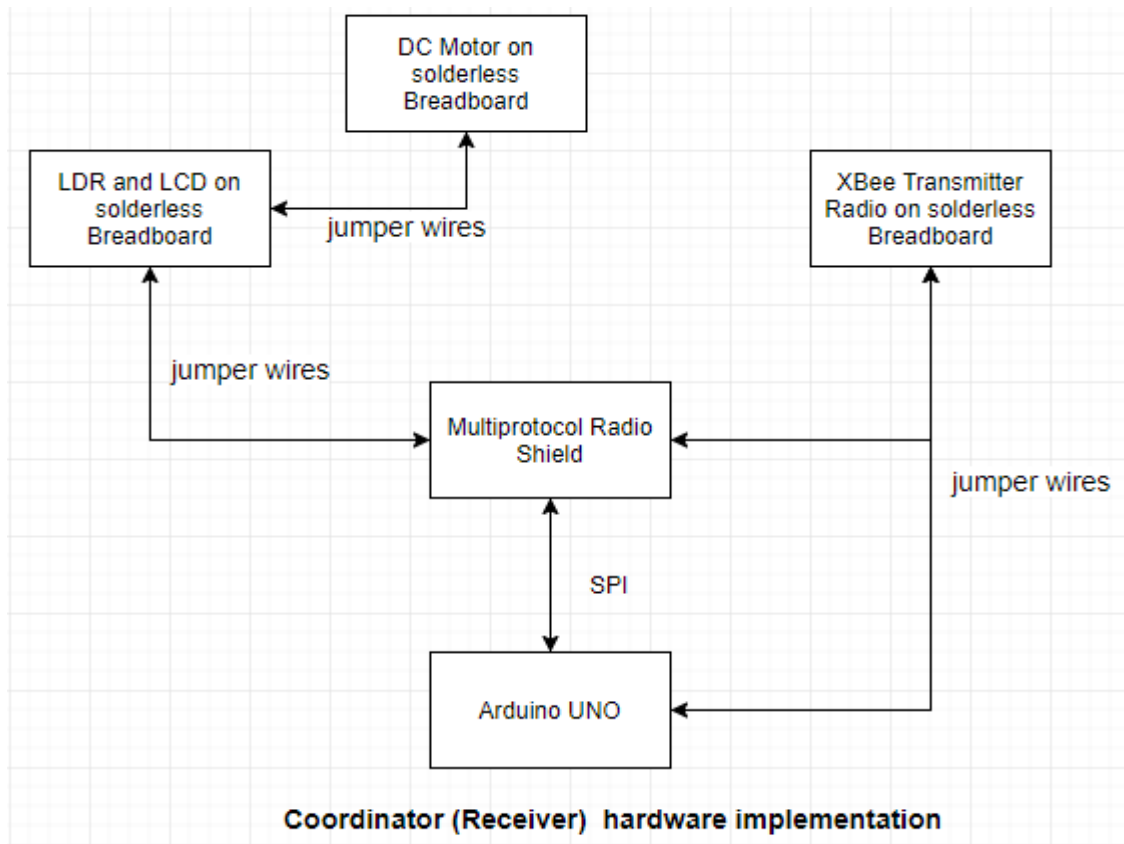
Wikipedia (2019b). *Mesh network*. Available online: https://en.wikipedia.org/wiki/Mesh_networking.

APPENDIX

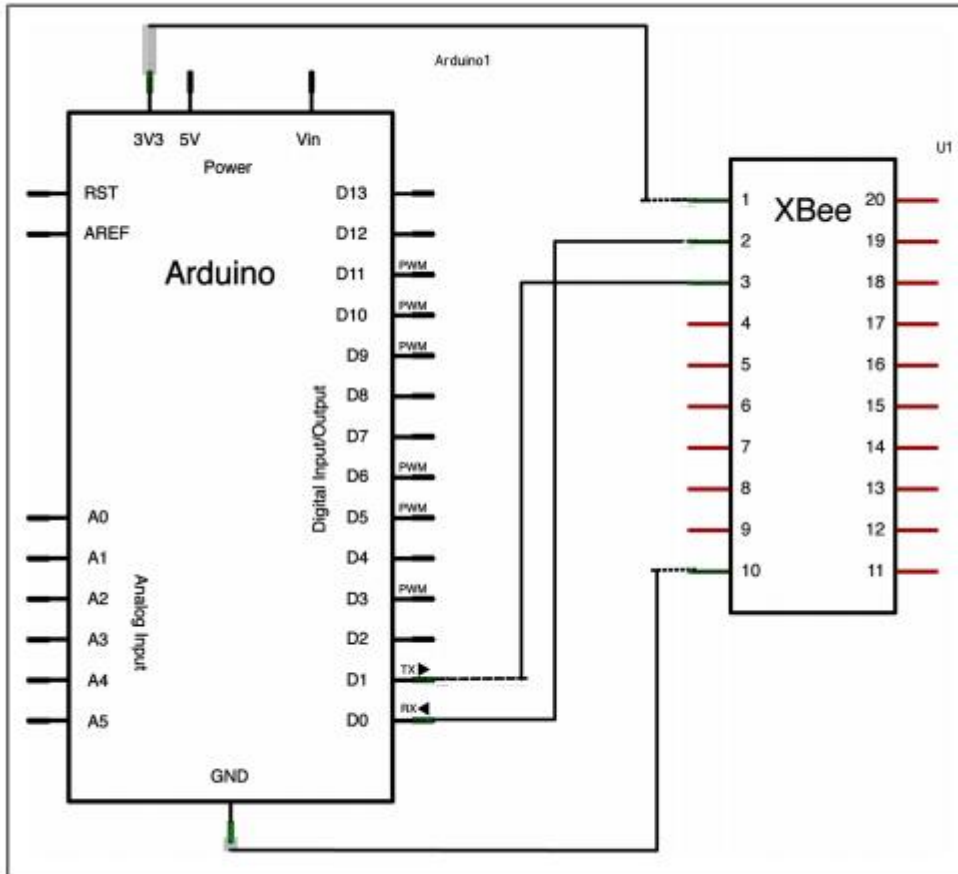
APPENDIX 1. Hardware Implementation Setup.



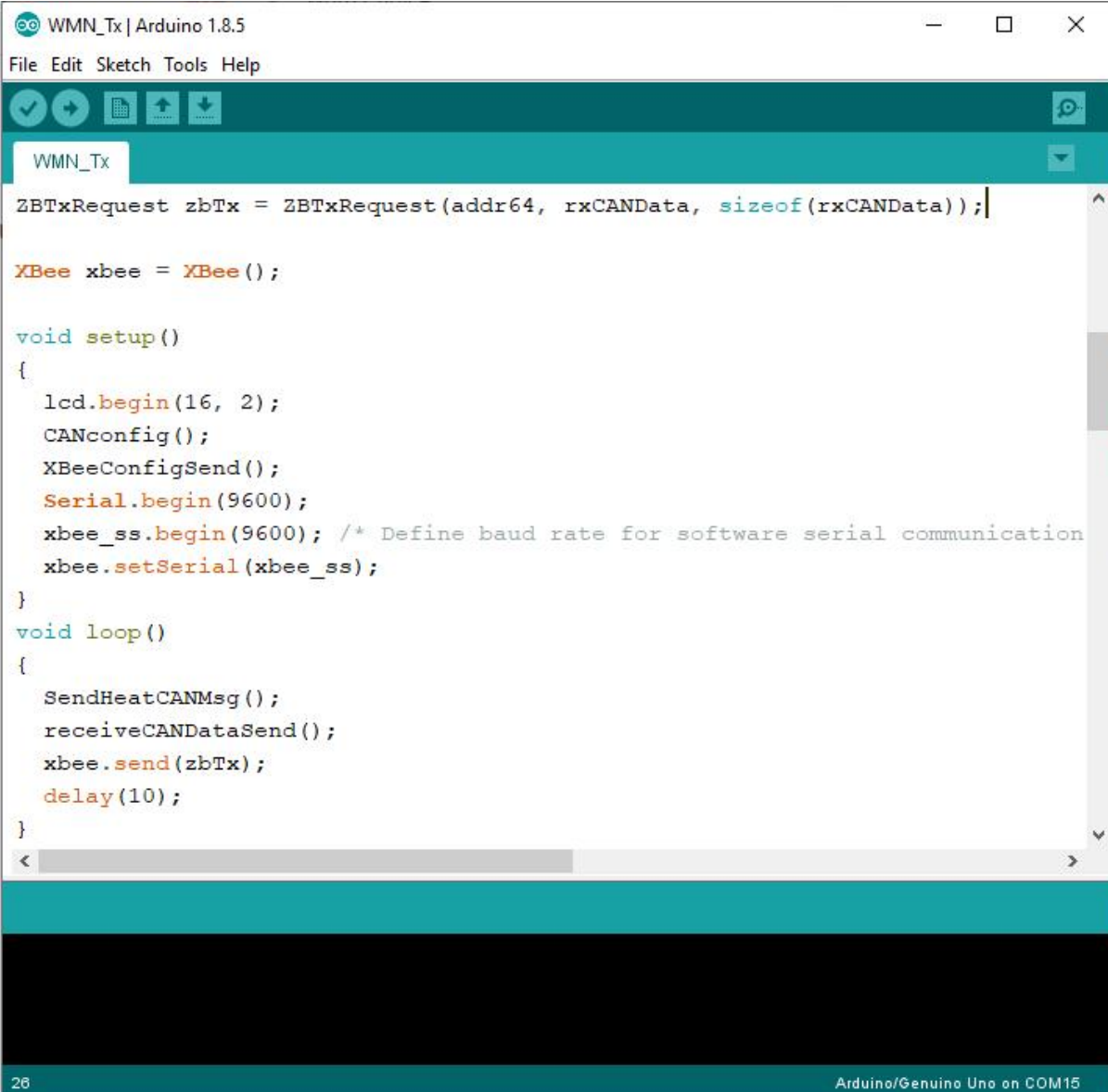
APPENDIX 2. Block diagram of hardware implementation.



APPENDIX 3. XBee-Arduino TX/RX connection schematic view (Robert Faludi 2011).



APENDIX 4. Arduino IDE environment.



```
WMN_Tx | Arduino 1.8.5
File Edit Sketch Tools Help
WMN_Tx
ZBTxRequest zbTx = ZBTxRequest(addr64, rxCANData, sizeof(rxCANData));
XBee xbee = XBee();
void setup()
{
  lcd.begin(16, 2);
  CANconfig();
  XBeeConfigSend();
  Serial.begin(9600);
  xbee_ss.begin(9600); /* Define baud rate for software serial communication
  xbee.setSerial(xbee_ss);
}
void loop()
{
  SendHeatCANMsg();
  receiveCANDataSend();
  xbee.send(zbTx);
  delay(10);
}
26 Arduino/Genuino Uno on COM15
```

APENDIX 5. Code for XBee End Device-Arduino Interface as Transmitter.

```

#include <XBee.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal.h>
#include <multiprotocolShield.h>
#include <Wire.h>
#include <MCP23008.h>
#define ssRX 2 /* Rx pin for software serial */
#define ssTX 3 /* Tx pin for software serial */
int ledBlink = 10;
#define relay 12
int LED = 10;
int LDR = A0;
String lightON = "It's Dark Outside; Lights status: ON";
String lightOFF = "It's Bright Outside; Lights status: OFF";
LiquidCrystal lcd(9, 8, 7, 6, 5, 4);
/* Create object named xbee_ss of the class SoftwareSerial */
/* Define pins for software serial instance named xbee-ss(any name of your choice) to be
connected to xbee */
/* ssTx of Arduino connected to Din (pin 3 of xbee) */
/* ssRx of Arduino connected to Dout (pin 2 of xbee) */
SoftwareSerial xbee_ss(ssRX, ssTX);
// Create an instance of the object
// Variables
long unsigned int rxCANId;
unsigned char len = 0;
unsigned char TxLDRValue[1];
XBeeAddress64 addr64 = XBeeAddress64(0x0013A200, 0xFFFF);
ZBTxRequest zbTx = ZBTxRequest(addr64, TxLDRValue, sizeof(TxLDRValue));

XBee xbee = XBee();

void setup()
{
  Serial.begin(9600);
  xbee_ss.begin(9600); /* Define baud rate for software serial communication */
  xbee.setSerial(xbee_ss);
  pinMode(LED, OUTPUT);
  pinMode(relay, OUTPUT);
  pinMode(LDR, INPUT);
  lcd.begin(16, 2);
  XBeeConfigSend();
}

void loop()
{
  int LDRValue = analogRead(LDR);
  int j,m,index, indi, numb=sizeof(TxLDRValue);
  index=0; indi=0;

```

```

// Put LDRValue into TxLDRValue
for(m = 0; m<numb; m++)
{
  TxLDRValue[index] = LDRValue;
  index++;
}
Serial.print("sensor Value = ");
for(m = 0; m<numb; m++)
{
  Serial.print(TxLDRValue[m]);
}
Serial.println(" ");
xbee.send(zbTx);
delay(100);
/*****
BLINK LED to indicate TRansmission is ONGOING
*****/
if (LDRValue !=0)
{
  digitalWrite(ledBlink, HIGH);
  delay(100);
  digitalWrite(ledBlink, LOW);
  Serial.println(" ");
}
/*****
Send Command to TURN Lights ON Remotely if LDRValue <=70
*****/
if (LDRValue <=70)
{
  //digitalWrite(LED, HIGH);
  //digitalWrite(relay, HIGH);
  Serial.println(lightON);
  Serial.println(" ..... ");
  Serial.println(" ");
}
/*****
LCD Display code
*****/
  lcd.setCursor(0, 0);
  lcd.print(" Light Status: ");
  lcd.setCursor(0, 1);
  lcd.print(" Lights are on_");
}
else
/*****
Send Command to Turn OFF the Lights Remotely if LDRValue > 70
*****/
{
  Serial.println(lightOFF);
  Serial.println(" ..... ");
  Serial.println(" ");
}
/*****

```

LCD Display code

```

*****/
  lcd.setCursor(0, 0);
  lcd.print(" Light Status: ");
  lcd.setCursor(0, 1);
  lcd.print(" Lights are off");
}
}

void XBeeConfigSend()
{
  socket1.ON();
  delay(100);
  socket1.setMUX(); // <----- Configure the MUX at MUX0
  Serial.begin(9600);
  delay(1000);
}
/*****
  END OF FILE
*****/

```



```

/*****
  READ and FORMAT Incoming Packets
  *****/
xbee.readPacket(); /* Read until a packet is received or an error occurs */
if(xbee.getResponse().isAvailable()) /* True if response has been successfully parsed and
is complete */
{
  Serial.println(".....");
  Serial.print("receiving End Device Data with ApiId: ");
  Serial.println(xbee.getResponse().getApiId());
  //Serial.println(rx.getDataLength());
  if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
  {
    xbee.getResponse().getZBRxResponse(rx);
    for (int i = 0; i < rx.getDataLength() ; i++)
    {
      RxLDRValue [i] = rx.getData(i);
      sample += (char)rx.getData(i);
    }
  }
}
/*****
BLINK LED to indicate Reception of Incoming Data
*****/
if (rx.getDataLength() !=0)
{
  digitalWrite(relay, HIGH);
  delay(10);
  digitalWrite(relay, LOW);
}
/*****
  Print "Received LDR Value" to Serial Monitor
  *****/
if ((rx.getDataLength())>= dataLenght)
{
  for (int i = 0; i < 1; i++)//(int i = 0; i < rx.getDataLength(); i++)
  {
    Serial.println(" ");
    Serial.print(" Received LDR Value = ");
    Serial.print(RxLDRValue[i]);
  }
  Serial.println(" ");
}
/*****
Turn ON the Lights if (RxLDRValue[0] <=70)
*****/
analogWrite(motorPin, 0);
if (RxLDRValue[0] <=70)
{
  digitalWrite(LED, HIGH);
  digitalWrite(relay, HIGH);
  analogWrite(motorPin, 130);
  Serial.println(lightON);
  Serial.println(".....");
}

```



```

Serial.println(" ");
/*****
LCD Display code
*****/
    lcd.setCursor(0, 0);
    lcd.print(" Light Status: ");
    lcd.setCursor(0, 1);
    lcd.print(" Lights are on_ ");
}
else
/*****
Turn OFF the Lights if (RxLDRValue[0] > 70)
*****/
{
digitalWrite(LED, LOW);
digitalWrite(relay, LOW);
Serial.println(lightOFF);
Serial.println(" ..... ");
Serial.println(" ");
/*****
LCD Display code Sensor Node Data (LDR)
*****/
    lcd.setCursor(0, 0);
    lcd.print(" Light Status: ");
    lcd.setCursor(0, 1);
    lcd.print(" Lights are off ");
}
}
/*****
Print Router 3 Data to Serial Monitor
*****/
else if (sample.length() <= 20)
{
    Serial.println(" ");
    Serial.print(" Router 3 Data: ");
    Serial.print(sample);
    Serial.println(" ");
    Serial.println(" ..... ");
/*****
LCD Display of Router 3 Data
*****/
    lcd.setCursor(0, 0);
    lcd.print(" Router 3 Data: ");
    lcd.setCursor(0, 1);
    lcd.print(sample);
    delay(100);
}
}
}
/* If error detected, print the error code */
else if (xbee.getResponse().isError())

```

```

    {
    Serial.print(" Error reading packet. Error code: ");
    Serial.println(xbee.getResponse().getErrorCode());
    }
/* error detection ends*/
//     else
//     {
//     Serial.println(" ..... ");
//     Serial.print(" No Incoming Data");
//     lcd.setCursor(0, 0);
//     lcd.print(" No Incoming Data ");
//     lcd.setCursor(0, 1);
//     lcd.print(" CheckSensorNodes  ");
//     Serial.println(" ");
//     Serial.println(" ..... ");
//     }
}
void XBeeConfigSend()
{
  socket1.ON();
  delay(100);
  socket1.setMUX(); // <----- Configure the MUX at MUX0
  Serial.begin(9600);
  delay(1000);
}
/*****
  END OF FILE
  *****/

```