INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Implementation and Comparison of Low Power Wireless Protocols in a Mesh Topology

Amani Nafkha - 46653

Supervisor:

Prof. Paulo Jorge Teixeira Matos

Master in Industrial Engineering

2020-2021

Implementation and Comparison of Low Power Wireless Protocols in a Mesh Topology

Master in Industrial Engineering

Escola Superior de Tecnologia e Gestão

Amani Nafkha - 46653

2020-2021

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Dedication

First, I thank my God for giving me the strength and patience to do this work.

I dedicate this work to:

To my dear Mother and Father

Despite the distance, they gave me life, tenderness and the courage to succeed.

All I can offer cannot express the love and gratitude I have for them, I wish my God to preserve you and to give you health and life.

To my Brother **KAIS**

To my Sisters **Imen & Lamia**

To my Grandmother

To my Aunts, my Uncles and my Cousins

To all my Friends

Acknowledgements

I would like to warmly thank my thesis supervisor, Prof. **Paulo Jorge Teixeira Matos** for his availability, his competence and his human qualities, as well as his constructive remarks which allowed me to carry out this work.

Abstract

The Internet of Things is one of the upcoming networking that helps bridge the gap between the real world and the virtual world by enabling monitoring and control of certain elements.

The critical point for the future is the design with low power wireless technologies based on mesh topologies because it is very attractive due to their reliability and scalability of failures. In this thesis, we provide an overview of the most popular short-range wireless communication standards, such as BLE, Zigbee, and Thread technologies, comparing their key features and behaviors in terms of various metrics, network topology, security, quality of service, and power consumption.

This study presented in this report will be useful to the application in selecting the best technology for a concrete use of the low power wireless protocol.

Keywords: IOT, BLE, Zigbee, Thread

Resumo

A Internet das Coisas é a nova geração de redes de comunicações que vem preencher a lacuna entre o mundo físico e o mundo virtual, permitindo a monitorização e controlo a partir de qualquer local das "coisas" que fazem parte do nosso quotidiano.

Uma das vantagens está na utilização de tecnologias sem fio de baixa potência baseadas em topologias de malha, que faculta elevada confiabilidade e escalabilidade contra falhas. Nesta dissertação, é feita uma análise das tecnologias de comunicação sem fio de curto alcance mais populares, como é o caso do BLE, Zigbee e Thread. Para o efeito estas são comparadas em termos de características e funcionamento, utilizando várias métricas que incluem: topologias suportadas, segurança, qualidade do serviço e consumo de energia.

O estudo apresentado neste relatório é útil para na seleção da melhor tecnologia para um uso concreto do protocolo wireless de baixa potência.

Palavras-chave: IOT, BLE, Zigbee, Thread.

Contents

List of Figures

Acronyms

6LOWPAN Low power wireless personal area network. 27

AES Advanced Encryption Standard. 25

APS Application Support. 21

ARM Advance RISC Machine. 41

BLE Bluetooth Low Energy. 4

CLI Command-line interface. 41

CoAP Constrained Application Protocol. 30

EUI Extended Unique Identifier. 13

FFD Full-Function Devices. 23

HTTP Hypertext Transfer Protocol. 30

IDE integrated development environments. 35

IOT Internet of Things. 1

IPv6 Internet Protocol version 6. 31

KEK Key Encryption Key. 30

LPWAN Low-power Wide Area Networks. V, 3

MAC Medium Access Layer. 13

MLE Mesh Link Establishment. 27

NWK Network Layer. 21

PAN Personal Area Network. 23

PHY Physical Layer. 21

RAM Random Access Memory. 32

RFD Reduced-Function Device. 23

RTT Real Time Terminal. 41

SDK Software Development Kit. 41

SES Segger Embedded Studio. 41

SKM Security Key Management. 15

UART Universal Asynchronous Receiver Transmitter. 41

UDP User Datagram Protocol. 58

UUID Universally Unique Identifier. 13

Wi-Fi Wireless Fidelity. 3

WLAN Wireless Local Area Networks. V, 3

WPAN Wireless Personal Area Networks. V, 2

ZDO Zigbee Device Objects. 21

Chapter 1

Introduction

1.1 Internet of things

The combination of computer hardware and networking advancements has given birth to the Internet of Things (IOT), with the purpose of letting everyday objects collect and exchange data over the internet with the use of small sensor devices.

Millions, if not billions, of Internet of things embedded around the world provide an incredibly rich set of data that companies can use to collect data about their safety of their operations, track assets and create new business opportunities. These things have different computing and sensorial capabilities, providing complex interactions with their environment or users. The range for IoT applications is large. By installing devices on objects such as lights, fridges, heaters, fans.It can be remotely monitored and controlled with in a wide variety of applications such as smart home, smart cities, health care, transportation, and industries. These applications have different needs and constraints like scalability, coverage and energy that induce a global heterogeneity in IoT. The growth of Internet of Things in terms of number of devices, revenue generated, and data produced has been stunning, but most predictions expect that growth to accelerate. As the figure 1.1 below shows, in 2021 the number of IoT connected device stood around 35.82 billion, and it has practically doubled in the last six years and is expected to increase more than

50% in next four years.

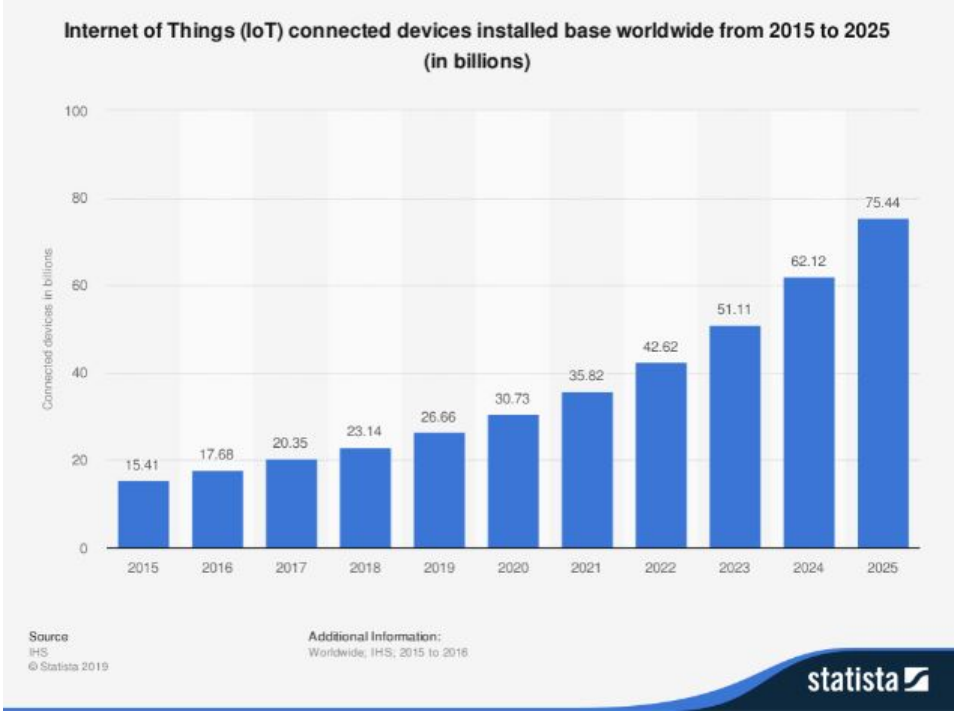

Figure 1.1: Growth in the number of IoT devices [1]

1.2 Wireless IoT Connectivity Technologies

The applications of IoT are limitless.The current limitation is technology, but with continued advances, more and more objects can be reliably connected to the internet at low cost and with low power consumption. The applications fall into two categories, short-range applications and long-range applications.

1.2.1 WPAN

The largest number of IoT devices are connected via short-range technology, which typically has a maximum range of no more than 100 meters.These include devices connected via Bluetooth, such as headsets, but also devices connected via Zigbee, which are mainly found in smart homes, for example to connect smoke detectors or thermostats [2].

1.2.2 WLAN

Another large category is Wireless Local Area Networks that cover connectivity of up to 1 kilometer [2]. Wireless Fidelity (Wi-Fi) is the most common standard in this category and seeing great growth, mostly through the use of home assistants, smart TVs, but also increasingly through use in industrial settings.

1.2.3 LPWAN

A large chunk of the future growth in the number of IoT devices is expected to come from low-power wide area networks [3].The technology, which promises extremely high battery life and a maximum communication range of over 20 kilometers is used by three main competing standards, Sigfox, Lora, and NB-IoT, which are currently being rolled-out worldwide.

1.2.4 Cellular / M2M

2G, 3G, and 4G technology had for a long time been the only option for remote device connectivity [3].

1.3 IOT Protocols

The basic elements of the IoT are devices that gather data. Broadly speaking, they are internet-connected devices, so they each have an IP address. To make that data useful it needs to be collected, processed, filtered and analyzed, each of which can be handled in a variety of ways. Moving the data can be done wirelessly using a range of technologies, or on wired networks. The data can be sent over the internet to a data center or a cloud that has storage and compute power or the transfer can be staged, with intermediary devices aggregating the data before sending it along.When IoT gadgets talk to other devices, they can use a wide variety of communications standards and protocols. For instance, as show below in figure 1.2, the different wide-area section in the wireless technology.

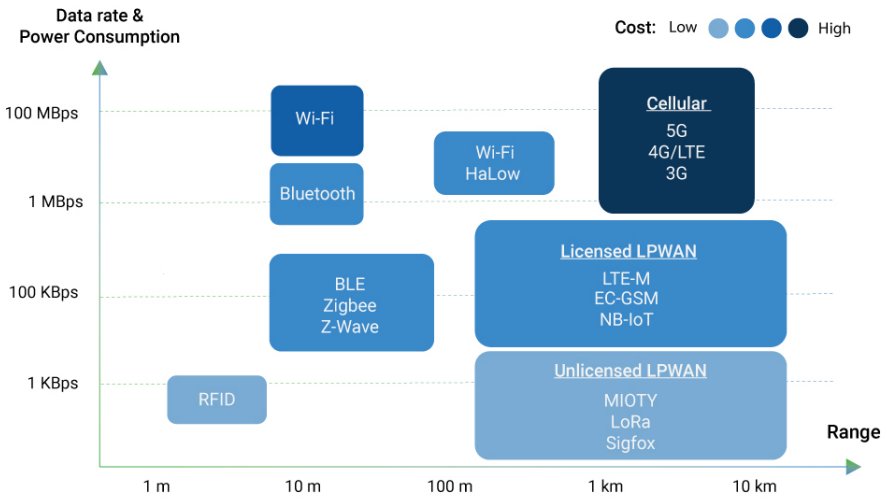

Figure 1.2: IoT wireless technology [3]

This work focus in the short-range wireless technologies such as Zigbee, Thread and BLE that support mesh networks to extend applications that require devices to communicate beyond their range and provide long distances with low transmission power and can achieve high reliability.

1.4 Objectives

The main object of this work is to analyze three wireless networking technologies available today for short range wireless internet of things: Bluetooth Low Energy (BLE), Thread and Zigbee.To achieve this main objective, the following structure for the analysis was proposed:

- Present the features and specifications of each technology.

- Put the difference that exist between the three technology.

- Implement mesh networks using these three technologies to assess which is the best technology for a concrete use.

1.5 Contributions

The main contributions of this work are:

1. Detailed study and comparative analysis of three IoT mesh protocols i.e., Zigbee, Thread and Bluetooth Low Energy.The analysis focuses on the study of functions, such as application configuration, device commissioning and security key management.

2. Evaluate the three technologies and associated SDKs, on the implementation of mesh networks.

3. Evaluate the three low energy technologies, choose the best one after implementing it with the mesh network.

1.6 Report summary

This thesis study is organized into six chapters:

Chapter 1: Introduces the background of different technology in internet of things and describe the purposes and goals.

Chapter 2: states some literature of some other work and provide further information of the network such as types of networks, network topologies and general devices types and roles.

Chapter 3: Introduce the three technologies: Zigbee, Thread and BLE, its features, characteristics, specifics, and security.

Chapter 4: Describe the hardware and software used to implement the project.

Chapter 5: Test and simulation of mesh network with the three technologies.

Chapter 6: Present the conclusion and the future works.

Chapter 2

Network Overview

This chapter reviews some current work related to the deployment of the Internet of Things in a common area of concern and explore network terminology, topology, attributes, and security.

2.1 Related works

There are few studies that have examined the IoT mesh protocols and compared them.

In [4], the authors have studied IoT protocols that are interesting, including Zigbee and Bluetooth. On the basis of range, data rate, power consumption, and payload size, they compared the IoT protocols.A significant takeaway from this work is that the authors suggest a mix of technologies based on their results to create an application.

In paper [5], the author compared general characteristics of some wireless networks such as applicability for some use cases, usefulness in IoT, power consumption, data rate, and distance. He also discussed "last 100 meters connectivity" of small IoT devices to services on the internet using wireless technologies.

In [6] "Neelakanta" and "Dighe" provided methods of evaluating the performance of ZigBee and Bluetooth wireless communication when they operate in an industrial environment.They compare these two short-range wireless technologies which both operate on the same 2.4 GHz ISM band but use different modulations and analyze their robustness

to interferences presented on a factory floor.

In article [7], authors present a comparative study of Bluetooth, ZigBee, Wi-Fi, WIMAX, GSM/GPRS. The main goal of the paper is selecting a better communication technology that guarantees the quality of communication, minimizes energy consumption, reduces the implementation cost. They use different theoretical performance evaluation metrics: network size, transmission time, transmission power and range, energy consumption, chipset power consumption.

In [8], author study the energy consumption of BLE and ZigBee/802.15.4 by measuring real devices and derive a model of typical energy consumption from the measurement results.

The following section defines common terminology and its definitions for the different roles and types of mesh devices, attributes and network security.

2.2 Terminology

2.2.1 Types of networks

According to the management method of network security, two types of mesh networks can be formed: centralized security network and distributed security network.

1.Centralized Security Network:is a type of mesh network with one device taking the role of managing the network security by generating and modifying the keys used to encrypt network messages.

2. Distributed Security Network:is a mesh network without dedicated equipment to manage network security.The security of the network is managed by each node in a distributed manner.

2.2.2 Types of IoT Protocols

Figure 2.1: IoT Network protocol/ IoT Data protocol [9]

1. IoT Network Protocols

IoT network protocols allow data communication within the scope of the network.For instance, the popular IoT Network protocols are Bluetooth, Zigbee and LoraWAN.

2. IoT Data Protocols

IoT data protocols are designed to connect low power IoT devices. Without any internet connection, while the connectivity in IoT data protocols can be done via a wired or cellular network.The popular IoT data protocols are CoAP, MQTT and AMQP.

2.2.3 Network Topologies

There are several types of IoT topologies for networking, the most common being mesh topology, star topology, and tree topology.

- **Star Topology**

  In the star topology, the communication is established between a single central controller called hub and devices or nodes, as shown figure below 2.2.This topology doesn't allow direct communication between devices, a device must have to communicate through hub [10].A star network is less expensive because each device need one I/O and needs to be connected with central with one link.

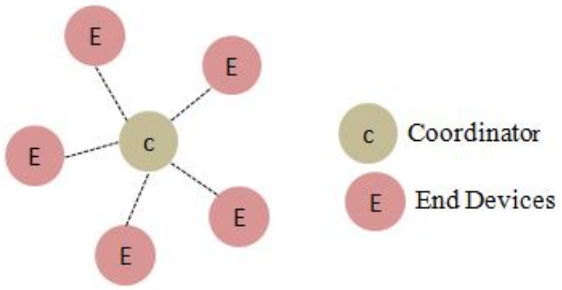

Figure 2.2: Star topology

- **Tree Topology**

  In this topology, the network consists of a central node, which is a coordinator, several routers, and end devices, as shown in Figure 2.3. The function of the router is to extend the network coverage. The end nodes that are connected to the coordinator or the routers are called children. Each end device is only able to communicate with a router or a coordinator. The coordinator and routers can have children and, therefore, are the only devices that can be parents. A special case of tree topology is called a cluster tree topology [10].

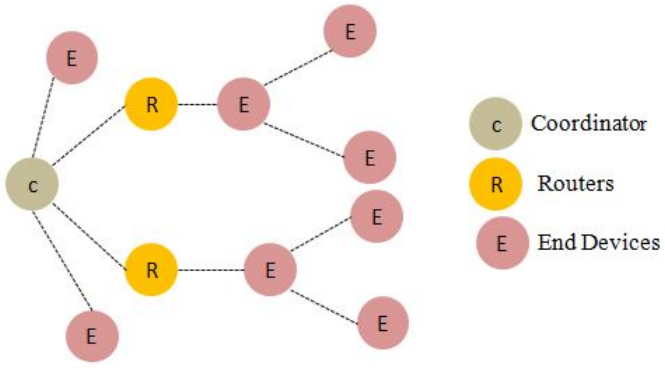

Figure 2.3: Cluster Tree Topology

- **Mesh Topology**

  In mesh, topology data can be sent via several paths from source node to the gateway.This topology has the ability to keep the list of nodes that have been given the access to use the network. When the node wakes up, the radio is turned to ON state and is able to scan the network, locate the sink node in order to join the network and begin taking the required measurements. For the reason that each node is joining the network upon its wake-up, there is a possibility of finding an alternative path to the sink node in case some paths are not available for data transmission [11].

  The main advantage of this topology is that it is fault tolerant, in case of a failure of one link an alternative link can be established to route information to the gateway.

  The following figure 2.4 shown the mesh topology.

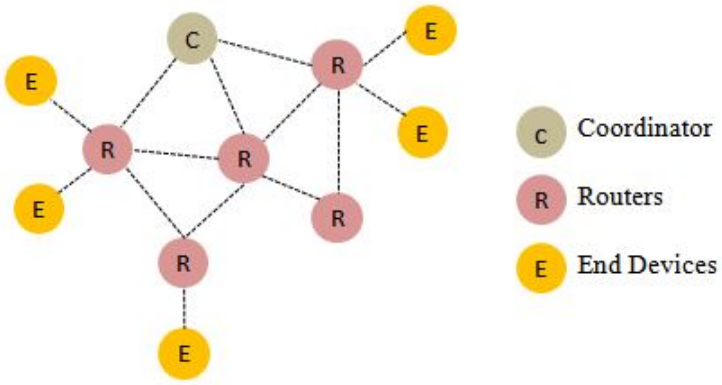

Figure 2.4: Mesh Topology

2.2.4 General devices types and roles

The IoT mesh network is composed of several device types and roles. This section defines the types and roles of general equipment.

- A mesh device: is any hardware that contains an interface intended to be part of an IoT mesh network.

- Joiner: is any Mesh device to be added to the network.

- Node: is a mesh device that can successfully send, receive and relay messages in the network.

- The commissioner: is any device or node that can authorize the joiner to be added to the IoT mesh network.

- Leaf nodes: are nodes that cannot forward messages in a mesh network.

- Non-leaf nodes: are nodes that can forward messages in a mesh network.

- Mesh Gateway: is a node that provides external access to the mesh network.

- The network manager: is a non-Leaf node that can authorizes nodes/devices to act as commissioners, and it has a centralized role in maintaining network data.

- The address attribute: is any device or node that assigns the address of the node.

- Joiner Manager: is a non-leaf node role during the commissioning of the device, also when the joiner is not within radio range of the commissioner, the joiner manager relays messages between them.

- Security manager: is responsible for generating and managing encryption keys in the mesh network.

2.2.5 Attributes of Network

- The network ID: is a unique identifier for a mesh network. It is used to distinguish other networks on the selected channel.

- The network security type: indicates the type of security used in a mesh network, that is, a centralized security network or a distributed security network.

- The type of commissioning of the network: indicates the type of commissioner the network may have. That is, in the case of decentralized commissioning, all non-leaf nodes can act as commissioners, and in the case of centralized commissioning, a designated specialist can add joiners to the mesh network.

- Network data: is data specific to the network, i.e. the address of each node, the number of nodes on the network, the address of the commissioner, etc. It is kept up to

date by the network manager.

- Network Channel: is a logical identifier of the wireless channel on which the mesh network can operate. The channel is selected during network formation, and it is not fixed and can be changed at anytime.

- Network Security Credentials: is a general term that refers to a set of security attributes.

2.2.6 Security Attributes

The nodes in the IoT mesh network use symmetric keys to encrypt and decrypt messages due to their resource constraints.The security key used to protect the debugging process and the encrypted data at the network layer and application layer are as follows:

- The layer key: is a symmetric key used to encrypt messages at the network layer or Medium Access Layer (MAC) layer.

- Joiner Secret: is an encryption key used to authenticate a joiner during device commissioning.

- Joiner Secret Type : indicates the type of joiner Key, for example unique or global.

- Network secrets: are network-specific codes used to authenticate external devices.

- Device key : is the only key, shared only between the commissioner and the nodes in the network. It is used to encrypt the message between the principal and the node.

- Application keys: are used to encrypt messages at application layer.Every application in the network can have a unique application key to prevent message decryption by nodes that are not part of the application.

2.2.7 Attributes of Mesh Device

- The application object ID: is a unique identifier of the application object.

- Mesh device ID : is a unique identifier for each mesh device, for example, Universally Unique Identifier (UUID) and Extended Unique Identifier (EUI).

- The node address : is a unique identifier specific to the node of the mesh network.

- Mesh device type : indicates the ability of the mesh device to forward messages in the network to form a mesh network.

- Security type indicates: is the type of security that the mesh device may use in the network.

- The channel list: contains the channel list, which joiner will scan during device debugging to discover the mesh network.

Chapter 3

Mesh protocol analysis

In this chapter, the three IoT technologies Zigbee, Bluetooth, and Thread will be examined in detail: the layers, the role of devices, the addresses of related functions, and Security Key Management (SKM). The analysis will be used as an input to the defined implementation of these technologies for the next chapter.

3.1 Bluetooth Low Energy

3.1.1 Definition

Bluetooth Low Energy operates in the 2.4 GHz ISM band, it is a robust communication standard with an extensive number of features available in almost every commercial electronic device. It is capable of file transfer, real time phone calls, etc. On the other hand, Bluetooth Low Energy was specially developed for low-power critical applications, like battery-operated IoT devices.

3.1.2 Bluetooth Low Energy Architecture

This section take a closer look at the Bluetooth Low Energy architecture, its layers and their respective responsibilities as shown in the figure 3.1.

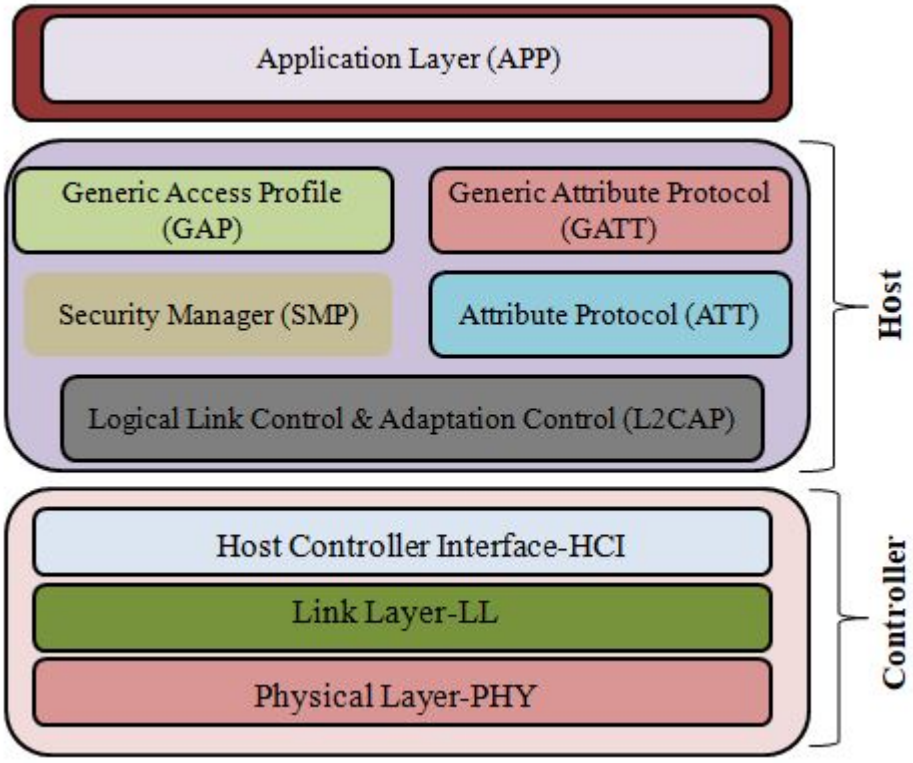

Figure 3.1: BLE Protocol Stack

Let us now review functions of different layers of this BLE protocol stack [12]

3.1.3 BLE Stack

In the following sections, all the layers of the protocol stack will be described in detail:

- **Physical Layer**

  The physical layer is the lowest layer of the protocol stack and provides its services to the link layer. It works in 2.4GHz ISM unlicensed band and have 40 RF channels with 3 of them as advertising channels and all others are data channels.

  Advertisement channel are used for device discovery, during connection establishment and for broadcasting. While the data channels are used for bidirectional communication between connected devices and advertisements extensions.

- **Link Layer LL**

  This is the layer that interface directly with the physical layer. It is responsible for managing the state of the radio, as well as, the timing requirements for adhering to the Bluetooth Low Energy specification.Link layer states are defined in the figure 3.2.

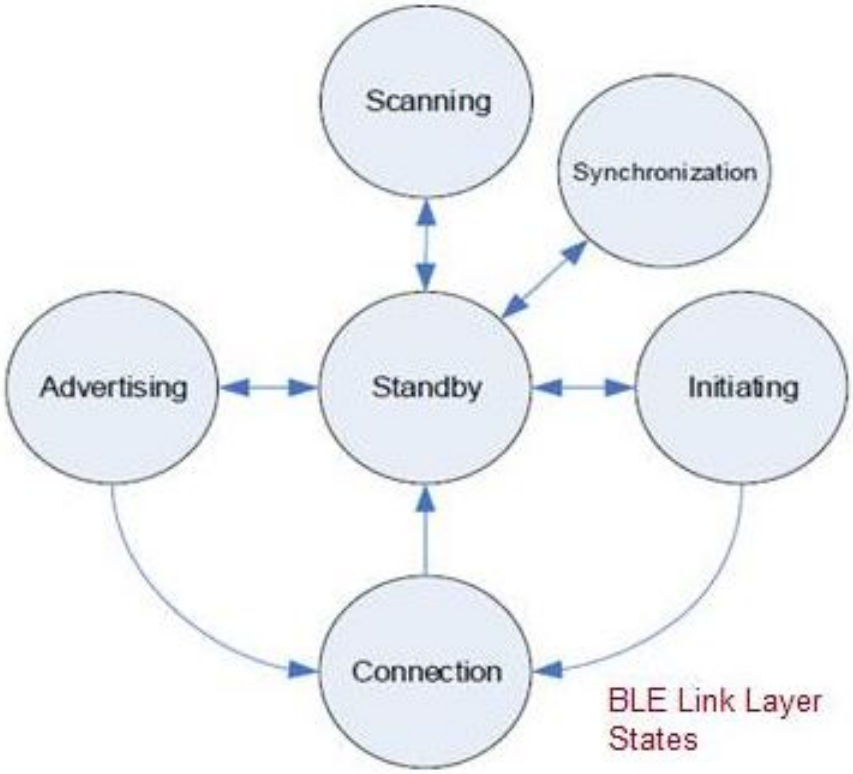

Figure 3.2: Link layer state machine [13]

- **Host controller interface**

  The host controller interface layer should have the ability to communicate data without prior knowledge of the data being transferred.Bluetooth Low Energy specification defined three host controller interfaces as USB, UART and SPI.

- **Generic access profile**

  The generic access profile defines fundamental operations in standard, discovering

device and connecting with peer device, establishing secure connections, broadcast packets.

- **Generic attribute Protocol**

  The Generic Attribute Profile (GATT) defines the format of the data exposed by a BLE device. It also defines the procedures needed to access the data exposed by a device.The GATT [14] have two roles, a GATT server and GATT client, they are completely independent of the GAP roles.The GATT server organizes data in attribute table and the attributes that contain actual data.

  - Service: Group of characteristics.Some GATT services are defined by SIG group, and they belongto the standard profiles.

  - characteristic: A characteristic contains at least two attributes, a characteristic declaration and the characteristic which holds attribute value.

  - Profiles: allow to define the behavior of the client and the server in terms of services and characteristics.

  - Attributes: generic term designating any type of data exposed by the server and defining the structure of this data.

- **Attribute Protocol**

  This layer allows BLE device to expose certain pieces of data or attributes.

- **Security Manager**

  The security manager layer provides methods for device pairing and key distributions. It offers services to other protocol stack layers in order to securely connect and exchange data between BLE devices.

- **Application Layer**

  This is the layer which will contain the user interface, application logic, and the overall application architecture.

3.1.4 Mesh topology of BLE

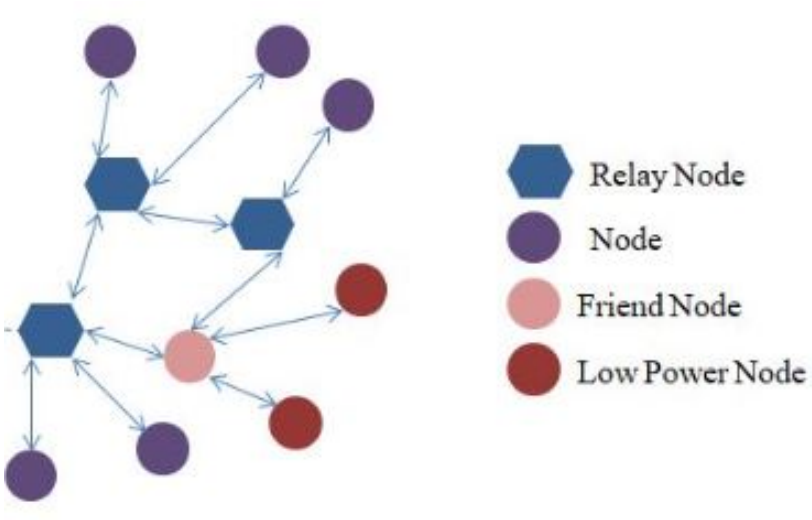

Figure 3.3: Bluetooth Mesh Network

The Bluetooth mesh defines few functions for the devices in the network. Those are:

Relay: Receiving and retransmitting mesh messages through advertising bearers to enable larger networks.

Proxy: Receiving and forwarding mesh messages between GATT and advertising bearers.

Low power: Only used with nodes that support the friend function, it can operate with a significantly reduced receiver duty cycle in a mesh network.

Friend: The ability to help nodes that support low-power features operate by storing messages sent to those nodes.

3.1.5 Mesh BLE Addresses

Public Device Address

This is the standard, IEEE-assigned 48-bit universal LAN MAC address which must be obtained from the IEEE Registration Authority. It is divided into two fields:

- IEEE-assigned company ID held in the 24 most-significant bits

- Company-assigned device ID held in the 24 least significant bits

Random Device Address Since all BLE packets include a Device Address, it's possible to track the BLE device as it's moving and communicating, unless it changes its address periodically. BLE adds the ability to periodically change the address. Two Random Address Types are provided: static address and private address

3.1.6 Message Security

All messages in the Bluetooth mesh network are encrypted and authenticated. Security is divided into different levels: network level security, application level security and device level because of this separation there are different types of security keys that can be used to solve each level of security issues as below [15] :

Network Key: This key is used to encrypt messages at the network layer and is generated by the pre-configuration device. A network can have multiple network keys, which can form subnets.

Application Key: This key is used to encrypt messages at the application level. The provisioner generates an application key, and then distribute it to the nodes participating in the application. Distribution is encrypted by the device key of each node that receives the application key.

Device keys: These keys are assigned to each provisioned mesh node by the provisioner. This allows for unique identification of mesh nodes.

3.1.7 Provisioning

Provisioning is the process of adding a device to the mesh network managed and becomes a node.It involves several stages, results in various security keys being generated. The provisioning is usually accomplished using a smartphone or other mobile computing device.

The provisioning procedure consists of five steps such as beaconing, invitation, exchange of public keys, authentication, distribution of provisioning data and application configuration.

3.2 Zigbee

3.2.1 Definition

Zigbee technology is a wireless communication standard that defines a set of protocols for use in low data rate, short to medium range wireless networking devices like sensors and control networks.It operates in the unlicensed bands like 2.4 GHz also, there are devices that use a different set of frequency bands like 784 MHz, 868 MHz and 915 MHz in China, Europe and USA [16].

Zigbee can be used on many applications such as home patient monitoring, where a patient wears a Zigbee device, which periodically collects the information.Another application of Zigbee is the building's structural health monitoring.Several Zigbee based wireless sensors like accelerometers are installed throughout the build.

3.2.2 Zigbee Architecture

The following image illustrates the Zigbee stack architecture. The Physical Layer (PHY) and Medium Access Layer MAC layers are defined by the IEEE 802.15.4 standard. On this foundation, the Zigbee Alliance provides the Network Layer (NWK) and the framework for application layer.The Application Support (APS) sub-layer, Zigbee Device Objects (ZDO) and manufacturer's application objects are all part of the application framework, which is controlled by the Zigbee Alliance [17].

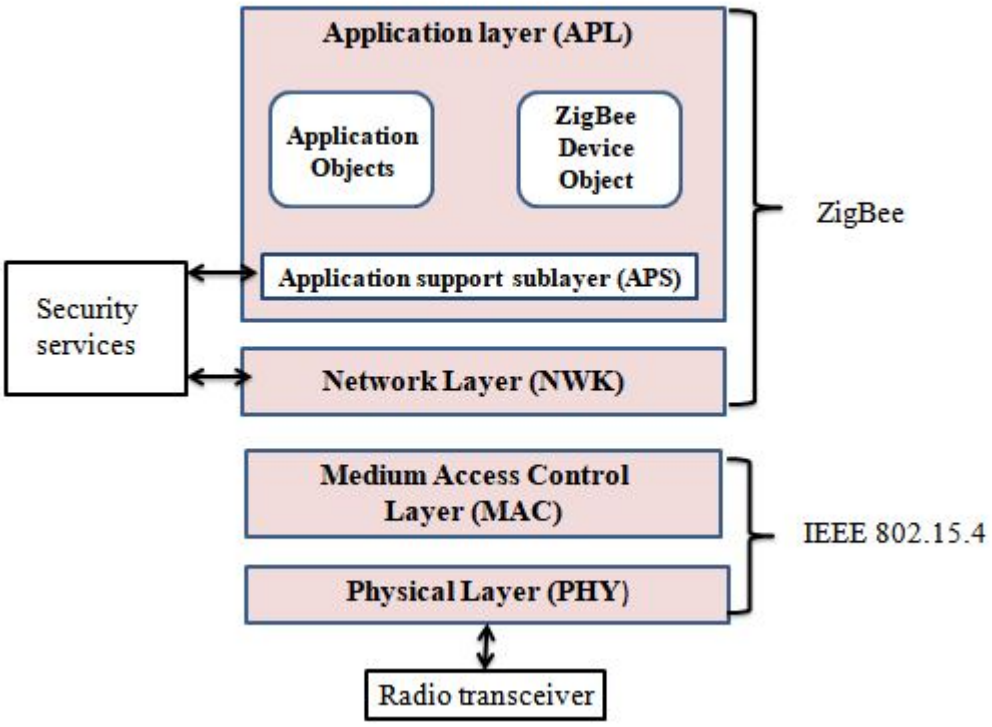

Figure 3.4: Zigbee Protocol Stack

The last four layers, namely transport, session, presentation and application layer, are introduced in the Application Support Sublayer (APS) and Zigbee Device Object (ZDO).

The ZigBee stack doesn't exactly fit the OSI Networking Model because the three lowest layers Physical, Data Link and Network Layers are present in the Zigbee Stack in the form of PHY, MAC and NWK.

3.2.3 Zigbee Layers

- **Physique Layer**

  PHY and MAC are defined by the IEEE 802.15.4 Specification. The PHY layer convert the data packets in to over-the-air bits for transmission and vice-versa during reception.The PHY layer is closest to the hardware, it directly controls and communicates with the Zigbee radio [16].

- **Mac Layer**

  The MAC layer is responsible for providing Personal Area Network (PAN) ID and network discovery through beacon requests, and it acts also as interface between PHY and NWK layers.

- **Network (NWK) Layer**

  The Network Layer (NWK) is responsible for interface between MAC and the Application Layers.It is also in charge of mesh networking (network formation and routing) [16].In addition, the NWK Layer protects Zigbee networks by encrypting all data in the NWK Frame.

- **Application Layer**

  The highest protocol layer in the Zigbee stack is the application layer. It is made up of application support (APS) sub-layer and Zigbee device object (ZDO) [16].The APS Sub-layer is responsible for discovery and binding services, and Zigbee device Object (ZDO) looks over the local and over-the-air management of the network.

3.2.4 Zigbee Devices

The IEEE 802.15.4 specification defines two types of devices: Full-Function Devices (FFD) and Reduced-Function Device (RFD).An FFD Device can communicate with any device in the network, and it must activate and always listening in the network.

An RFD Device can only communicate with an FFD device and is intended for simple applications like turning on or off a switch.

Zigbee protocol device	IEEE Device Type	Typical Function
Coordinator	FFD	One per network, allocates network addresses,holds binding table.
Router	FFD	Extend the physical range of network.Allows more nodes to join the network.
End devices	FFD or RFD	Perform monitoring and/or control function.

Table 3.1: Zigbee Protocol device types

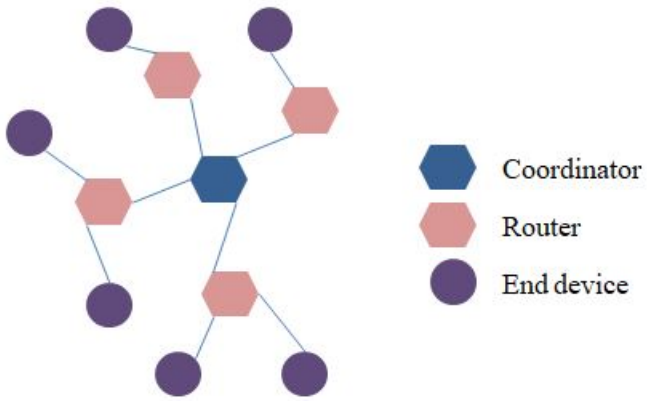

Figure 3.5: Zigbee mesh network types [18]

The Zigbee standard has created 3 Zigbee protocol device as mentioned in the figure 3.5

1. Zigbee coordinator node: A Zigbee coordinator is a PAN coordinator in the IEEE 802.15.4 Network, and it is responsible for forming the network. After establishing the network, it allocates network address for the devices that are allowed to join the network. It also routes the messages between the end devices.

2. Full-Function Devices: It is designed to store information about the network.There is one, and only one, ZigBee coordinator in each network to act as the router to other networks. It is designed to store information about the network

3. Reduced-Function Devices: This device is just network; it cannot relay data from other devices. Requiring even less memory, an RFD will thus be cheaper than an FFD. This device talks only to a network coordinator.

3.2.5 Identifiers

The devices in the Zigbee network have the following identifiers [19]:

- Extended PAN ID: is a 64-bit number that identifies the network. It must be unique to differentiate a network. All the nodes in the same network should have the same PAN ID.

- The operating 16-bit PAN ID: is set by the coordinator when the network is created, and it is set to routers and end devices when they join the network.Thus, the 16-bit PAN ID can not be configured by the user.

- The MAC address: It identifies uniquely a node inside a network due to it can not be modified, and it is given by the manufacturer.

3.2.6 Security

Zigbee only uses symmetric key encryption, and all encryption is based on Advanced Encryption Standard (AES) AES-128 [20]. The following is a list of keys supported by the Zigbee network:

1- Network Key: This key is used to encrypt all messages in the Zigbee network at the network layer.

2- Global Link Keys: Pre-configured public key for each Zigbee device. The key is used for encrypted distribution during device debugging network key.

3- Unique Link Key: A pre-configured key, which is unique to each device in the Zigbee network.It is only shared between the trust center and Zigbee devices.

4- Trust Center Link Key: A unique key generated by the trust center in a centralized secure network. It is only in the trust center and specific Zigbee devices to hide messages between them from other devices in the network.

5- Application Link Key: In the Zigbee network, the application link key can be used to encrypt the communication between related devices at the application layer.

3.3 Thread

3.3.1 Definition

Thread is a standards-based IPv6-based mesh networking protocol developed for directly and securely connecting products around the home to each other.It developed to run on low-power, low-cost IEEE 802.15.4 devices and can support networks of 250 nodes [21]. Thread's approach to wireless networking offers a secure and reliable mesh network with no single point of failure and support for low-power end devices.

3.3.2 Thread architecture

Table 3.2 shows the OSI-model used in the Thread network from the application-layer down to the physical-layer. It is not a new standard, but a combination of existing standards.Thread include CoAP, UDP, DTLS, 6LowPan, vector routing and LR-WPAN.

Layers	**Protocol**	**Standards and comments**
Application	CoAP	RFC's: 7252
Transport	UDP+DTLS	RFC's: 768, 6347, 4279, 4492v, 3315, 5007
Network	6LowPAN + Vector Routing	RFC's: 4944 ,4862, 6775, 1058, 2080
Link	Low rate WPAN	IEEE 802.15.4 MAC
Physical	Low rate WPAN	IEEE 802.15.4 PHY

Table 3.2: The OSI-model for the Thread protocol stack

3.3.3 Thread Layers

This section discusses the Thread network layers corresponding to the standards used in each layer of the stack.

- **IEEE 802.15.4 Layer**

  IEEE 802.15.4 standard is suitable for home networking and for wireless sensor network applications.It operates at the frequency of 2.4 GHz ISM band with 16 channels.Additionally, it supports low latency devices and has low power consumption. The MAC layer in 802.15.4 standard is responsible for management of data, channel acquisition, addressing, and error correction.The physical layer is used for transmission and reception of packets over physical medium with activation and deactivation of radio trans-receiver.

- **6LowPAN Layer**

  Low power wireless personal area network (6LOWPAN) is a simple low cost network which provides wireless connectivity to devices with limited power.Additionally, 6LoWPAN supports star and mesh topologies, and it is suitable for low powered devices having low bandwidth and requiring small packet size. IPv6 over Low-power Wireless Personal Area Networks layer provides adaptation between IP layer and 802.15.4 MAC layer.

- **Mesh Link Establishment (MLE)**

  The MLE (Mesh Link Establishment) protocol allows a node to periodically multicast the estimate of the quality of the links to neighboring nodes. Thereby, the MLE messages are transported using single-hop link local unicasts and multicasts between routers, and it defines link configuration and neighbor detection in Thread network.

- **Network Layer**

  Devices use IP routing to forward packets. A device routing table is populated with a compressed form of a mesh local address for routers and the appropriate next hop.

- **Transport Layer**

  Thread devices use user datagram protocol as defined in RFC 768 [22] for messaging between devices for mesh establishment and maintenance.

3.3.4 Thread Devices

There are different types of devices in a thread network. Figure 3.10 shows these components. They are as follows [21]:

- **Border Router**: A border router is a gateway, which provide services for devices within the 802.15.4 network, and it supports connectivity for Thread network to access other adjacent networks, such as Wi-Fi, Ethernet. A Thread network typically contains one or more border routers.

- **Router**: Routers provide routing services to network devices, and they are not designed to sleep.Also, they provide joining and security services for devices trying to join the Thread network.

- **Host**:The host devices are sleepy end devices.They only communicate through their parent router and cannot forward messages to other devices.

3.3.5 Thread roles

There are various roles of devices participating in a Thread network, depending upon their type and configuration of the Thread network join process.

- **Commissioner**: This is a special role that is essential for adding new devices to the Thread network. There may only be one active commissioner in the Thread network.

- **Joiner**: The device to be added to a commissioned Thread network

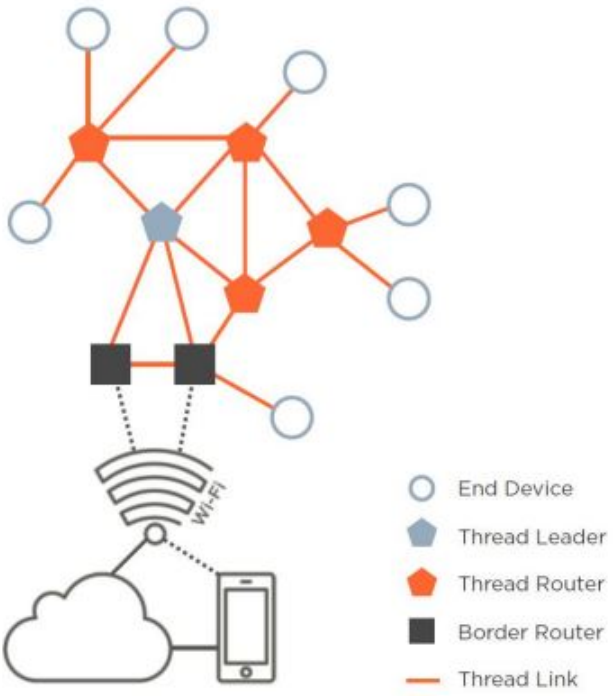

Figure 3.6: Thread Device Roles [21]

- **Leader**: The Thread network consists of only one router acting as the leader. It is responsible for maintaining network-specific data, such as the services available in the network, the number of routers in the network, and so on.

3.3.6 Identifiers

1. IPv6 Prefix: The network-specific prefix will be used to form the IPv6 addresses of all nodes in the network.

2. PAN ID: The personal area network ID is a 2-byte random value, which is unique to the Thread network.

3. Joiner ID: The unique identifier of joiner is pre-installed and will not be changed. It is based on the global IEEE EUI-64 identifier assigned at the factory and the MAC extended address (link layer address) assigned to the Thread interface.

4. RLOC16: The 16-byte unique value assigned to each node in the mesh network.

Leader generates RLOC16 and distributes it to all routers. Parents produce and Distribute RLOC16 to all terminal device [21].

5. XPAN ID: The extended personal area network ID is a random value of 8 bytes long and is unique to the Thread network.

3.3.7 Security

The Thread uses symmetric key encryption technology to encrypt messages in the network. The safety mechanism used in the Thread grid consists of the following safety materials.

- Master Key: This is the secret of the whole network of Thread mesh network. It produced when did the network form. This key was provided during the network joining.

- Network Credentials: It refers to a collection of network specific data, such as the IPv6 prefix of the network, Master Key, Network Name ...

- Key Encryption Key (KEK): It is a derived key of the device passphrase and is used to securely distribute network credentials to Joiner devices.

- MAC key: A symmetric key used to encrypt messages at the MAC layer of the Thread network.

- MLE key: A symmetric key used to protect the message of the mesh link establishment protocol.It is derived from the master key using a hash algorithm.

3.3.8 Constrained Application Protocol

Some key network features for IoT devices are low power and processing consumption for better battery life and response time.The recommended and compatible network protocol for Thread is Constrained Application Protocol (CoAP) [23].

CoAP is a lightweight machine-to-machine web application networking protocol. The format has similarities to Hypertext Transfer Protocol (HTTP), but offers lower power and processing consumption, combined with smaller packet sizes to better accommodate IoT devices. CoAP runs over UDP instead of TCP for greater simplification as well as compatibility with 6LowPAN

Figure 3.7 shows the application, transport and networking layer protocols.

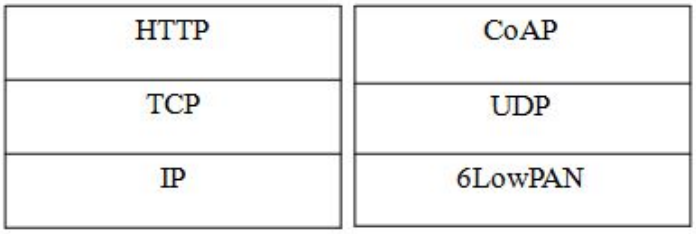

Figure 3.7: HTTP and CoAP protocol stacks

3.4 Comparison

Comparing ZigBee, Thread and Bluetooth in common characteristics, they look similar to each other. They are both from the same standard group IEEE 802.15 they use the same unlicensed 2.4 GHz operational frequency band to perform their low power connectivity and low cost. There are still some differences between two competitive technologies in their specifications. Bluetooth's specification is based on IEEE 802.15.1 group and ZigBee and Thread implements IEEE 802.15.4. They are two different technologies in their design. At very beginning Zigbee more focused on controlling, monitoring and automation and Bluetooth belongs to be more focused on creating a network with laptops, mobile phones. Otherwise, Thread is fine for local IoT systems, and it is not intended for long-distance data transmission.

Thread is the youngest technology compared to Zigbee and BLE mesh. It is a Internet Protocol version 6 (IPv6) based mesh networking protocol. This protocol has a limited number of routers (32 per network) and it is not appropriate for large-scale networks. However, can connect up to 511 devices per router. So, this networking standard works well for high-density meshnets. This standard has a sufficiently high data rate compared to similar technologies. Thread is fine for local IoT systems, and it is not intended for long-distance data transmission.

Bluetooth is designed to be used for self-organizing wireless ad-hoc networks.Ad-hoc networks do not have a permanent structure and provide decentralized wireless networks between devices. Devices are connected dynamically and forming a network. The bandwidth of Bluetooth is 1 Mbps; ZigBee and Thread are one fourth of this value. The strength of Bluetooth lies in its ability to allow interoperability and replacement of cables, ZigBee and Thread, of course, are low costs and long battery life.

Zigbee is much older than its competitors. Zigbee is a low-power and low-bandwidth wireless network.It is intended for low data rate networks. It has a fixed maximum rate of 250 Kbits/sec. Zigbee RF4CE [17] is a standard developed for radio frequency remote controls.

The main distinction of BLE mesh networking is that it uses Bluetooth instead of the Internet Protocol and managed flood messaging instead of routing, so, devices don't need much Random Access Memory (RAM). When the routing mechanism applied in Zigbee and Thread assists to direct messages to the destination nodes. Moreover, compared to flood messaging, routing has a more complicated implementation. But at the same time, it provides faster data transfer with lower overhead.

Each of the three standards provides wireless mesh communication and is widely used in IoT and smart home systems. The technologies are suitable for networks with low bandwidth and low power consumption. Security is a significant feature of all the protocols.

The three protocols presented are summarized in a Table 3.3, along with the criteria used to compare them and other features such as bandwidth, maximum number of nodes per network or security.

Specifications	**BLE**	**ZigBee**	**Thread**
IEEE Standard	802.15.1	802.15.4	802.15.4
Frequency Band	2.4GHz	2.4GHz, 868 MHz, 915MHz,	2.4GHz
Range	10m	1-100m	30m
BandWidth	1mbps	250kbps	250kbps
Node/Network	7	65000	300
Routing	Managed Flooding	Full Routing	Full Routing
IP support	No	No	yes
Security support	AES 128	AES 128	AES 128,ECC
Scalability	Yes	Yes	Yes
Cloud connectivity	Gatways	Gatways	Border Router / Gatways
Protocol layering	Network and Application	Network and Application	Network
Battery Type and life	can operate on a coin cell for several years	can operate on a coin cell for several years	can operate on a coin cell for several years

Table 3.3: Physical layer parameters of BLE,Zigbee,and Thread

Chapter 4

Development tools

During this thesis, a wide variety of software, integrated development environments (IDE) and hardware were used to perform the implementations.

4.1 Comparative Study of Development Kits

The most relevant characteristic when choosing a module are power consumption, processing capacity and storage.For this reason, six developments kits(Dks) that support all 3 technologies meet the hardware requirements currently on the market.These modules are presented in table 4.1.

As shown in the table, some modules have similar characteristic that more important in terms of performance, while others stand out for their relativity low power consumption. In terms of processing, the highlight was Nordic Semiconductor's nRF52840 module, which has a higher clock speed, up to about 2-3 times faster than Texas Instruments' cc2538 and Qorvo's K32W061/41.

When analyzing the RAM storage capacity, the nrf52 module was also superior to the others.Another aspect analyzed was Flash memory and here again, the nrf52840 was the most powerful, about 32 times more than the Texas Instruments. Finally, the last characteristic compared was the power consumption of the different modules during transmission and reception operations, and the Nordic Semiconductor nRF52840 and

nRF5340 are the most economical.

	Processors	Ram	Flash	Energy Consumption
Nordicsemiconductor nRF5340	Cortex-M4 64MHZ	64kB	256 kB	3.4mA TX/ 3.1mA Rx
NordicSemiconductor nRF52840	Cortex-M4 64MHZ	256kB	1MB	4.8mA TX/ 4.6mA Rx
NXP K32W061/41	Cortex-M4 48MHZ	152kB	640KB	7.4mA Tx/ 4.3mA RX
Qorvo QPG6095	Cortex-M4 64MHZ	64kB	512kB	11.2mA Tx/ 7.4mA Rx
Silicon Labs EFR32MG21	Cortex-M33 80MHZ	96kB	1024kB	9.3mA Tx/ 9.4mA Rx
Texas Instruments cc2538	Cortex-M3 32MHZ	32kB	512kB	24mA Tx/ 20mA Rx

Table 4.1: Comparison between some modules that operate with Bluetooth Low Energy, Zigbee and Thread [24] [25] [26] [27] [28].

4.2 Hardware

After conducting this analysis of the different modules, Nordic Semiconductor was chosen, as it is the best in terms of low power consumption and performance. Moreover, this module allows working in a mesh network, which can connect several devices and communicate at the same time. Thus, in this work, different boards are chosen such as Arduino Nano 33 BLE, Arduino Nano 33 BLE Sense and nRF52840 Dongle which are described below.

4.2.1 Arduino Nano 33 BLE

The Arduino Nano 33 BLE featuring a lot more powerful processor from Nordic Semiconductors nRF52840, a 32-bit ARM Cortex M4 CPU running at 64 MHz. It has 1 MB

of program memory, and with a lot more variables.The main processor includes other amazing features like Bluetooth pairing via NFC and ultra low power consumption modes [29].

Figure 4.1: Arduino Nano 33 BLE

Features	**specification**
Processor	32-bit ARM Cortex M4 CPU
Flash memory	1 MB
Radio	2.4GHz
Clock Frequency	64 MHz
Protocols	Bluetooth mesh, Thread, Zigbee,

Table 4.2: Arduino Nano 33 BLE specification

4.2.2 Arduino Nano 33 BLE Sense

Arduino Nano 33 BLE Sense board has been designed to offer a power savvy and cost-effective solution. It is based on a NINA B306 module, that hosts a Nordic nRF52480 that contains a Cortex M4F microcontroller [29]. The Nano 33 BLE Sense is the same as the Arduino Nano 33 BLE with the addition of a set of sensors such as temperature, pressure, humidity.

Figure 4.2: Arduino Nano 33 BLE sense

Features	specification
Processor	32-bit ARM Cortex M4 CPU
Flash memory	1 MB
Radio	2.4GHz
ClockFrequency	64 MHz
Protocols	Bluetooth mesh, Thread, Zigbee,

Table 4.3: Arduino Nano 33 BLE sense specification

4.2.3 nRF52840 USB Dongle

The nRF52840 USB Dongle (board number PCA10059) is a device from Nordic Semiconductor that has the advantages of being small and low-cost while being able to use most of the short-range wireless standards including Bluetooth Low Energy. It can be used for product and software development with the help of the nRF52 SDK and the application nRF Connect for Desktop, which is why it is a great tool to develop wireless applications. The Dongle features a green LED (LD1), a multicolor RGB LED (LD2), a user configurable button (SW1), and a reset button (SW2). It also has a USB-A-type connector printed on the circuit board and 15 GPIOs in addition to the ground, power and SWD connections along the castellated edges [30]. It does not feature an onboard debugger, but instead it can be programmed easily with nRF Connect software provided by Nordic Semiconductor.

This requires the software for the dongle to be developed and debugged on a nRF52840 development kit which has an onboard debugger.

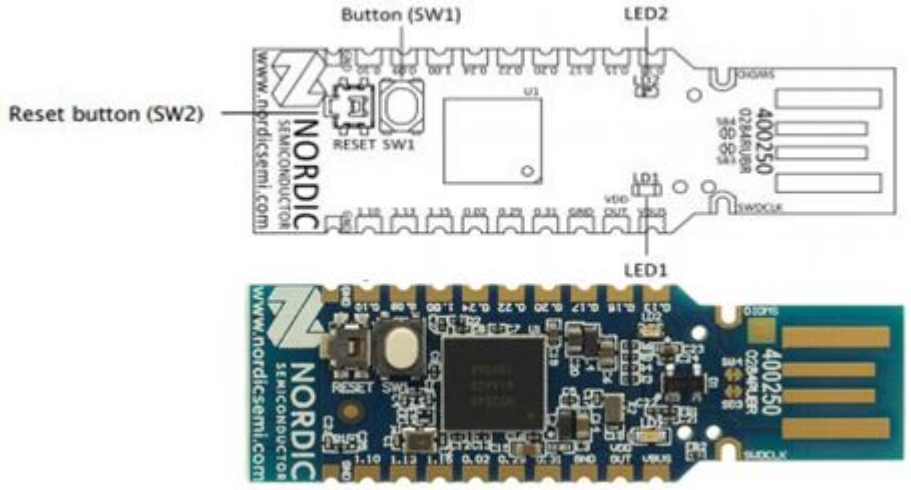

Figure 4.3: nRF52840 Dongle's buttons and LEDs drawing [30]

Features	**specification**
Processor	32-bit ARM Cortex M4 CPU
RAM	256 kB
Flash memory	1 MB
Radio	2.4GHz
Clock Frequency	64 MHz
Protocols	Bluetooth mesh, Thread, Zigbee,

Table 4.4: nRF52840 Dongle Specifications

4.2.4 An External Hardware Debugger

nRF52 development kit is provided with Bluetooth Low Energy, Zigbee and Thread protocol stack from Nordic Semiconductors. To compile and run software on the nRF52840 devices, Segger Embedded Studio IDE and J-link to debug and run code on development kits were used.

There are a variety of external hardware debugging devices available, and in this project the SEGGER J-Link debugger "J-Link EDU Mini" was used as shown in figure below.

Figure 4.4: J-Link EDU Mini

The connections between the two devices, J-Link EDU Mini and Arduino Nano 33 BLE as shown in figure below 4.5.

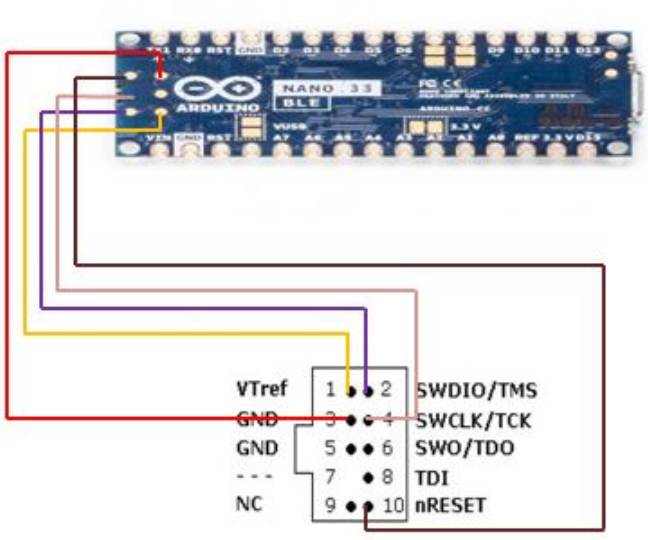

Figure 4.5: Wiring diagram of Arduino Nano 33 BLE

4.2.5 SEGGER Embedded Studios

Segger Embedded Studio (SES) is a powerful IDE with different tools for project editing and debugging for Advance RISC Machine (ARM) Cortex devices. It is free for all users of Nordic Semiconductor's Software Development Kit (SDK). Different compilers for embedded development are also included and there is support for cross-platform development for both Windows, macOS and Linux.

4.2.6 Segger J-Link

This is a debugging tool that has been used for flashing the nRF52 DK. This even involves a J-Link Real Time Terminal (RTT) viewer [31] which was used for debugging the C code in real time once the board using the Segger Embedded Studio Integrated Development Environment (IDE).

Nordic Semiconductor provides tools that are useful when doing developing, programming, and debugging.

4.2.7 nRF Connect

nRF Connect is an app for getting familiar with, developing, and testing devices. It allows setting up a local device, connect it to others devices and discover their services, maintain the connection and the connection parameters, pair the devices, and change the server setup for the local device.

4.2.8 Putty

Terminal used for Command-line interface (CLI), Universal Asynchronous Receiver Transmitter (UART) and USB debugging.

4.3 Zigbee and Thread implementation

For the Zigbee and Thread protocols, Segger's Embbeded Studio for ARM was used to provide a complete set of functions to meet the needs of this project. As SDK, the "nRF_SDK_For Zigbee and Thread" version was used as it is the version available from Nordic Semiconductor for these protocols.

For the implementation tests, the nRF connect for desktop software and the Putty command line interface were used.

Figure 4.6 below shows the technologies that used.

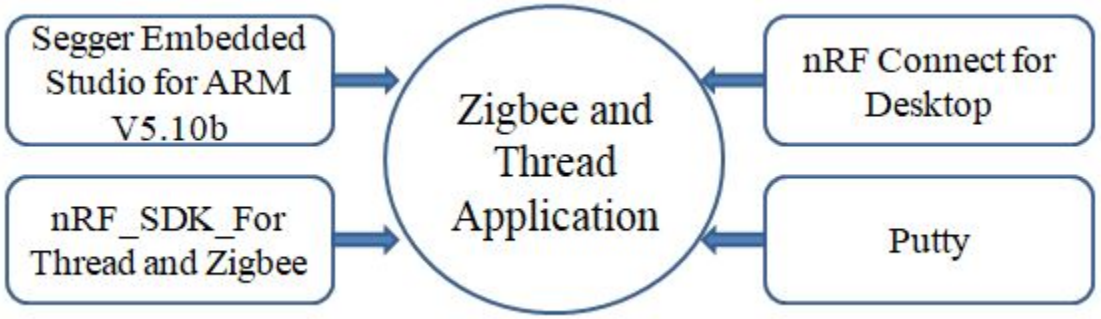

Figure 4.6: Technologies used in the Zigbee and Thread implementation

4.4 BLE implementation

As an integrated development environment (IDE), Segger Embbedded Studio for ARM 5.10b was used to provide, free of charge, a complete set of functions to meet the needs of this project. As an SDK, the nRF5.SDK.17.0.2.d674dde version was used, as it is the latest version available from Nordic Semiconductor. Regarding the implementation of the BLE protocol stack, it was chosen to use the S140 Softdevice, as it is supported by the PCA10059 and PCA10056 boards and allows the device to be implemented both as a peripheral and a central device. Thus, before running, the program that uses Bluetooth must first program the SoftDevice on the board with nRFgo Studio and nRF Connect desktop applications used to implement device tests.

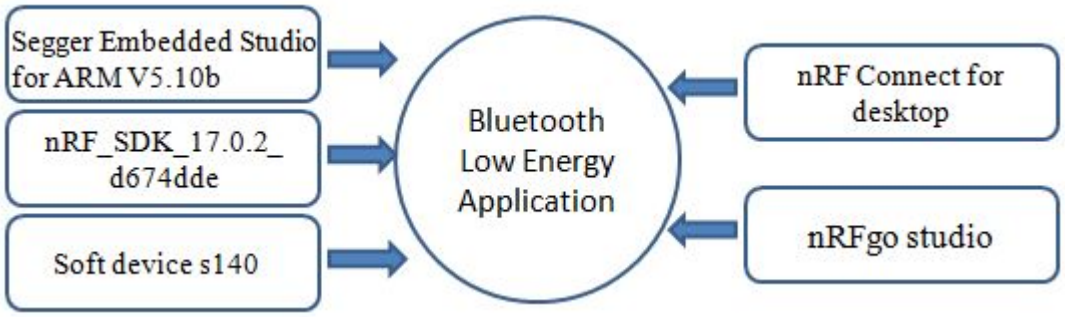

Figure 4.7: Technologies used in the BLE implementation

4.5 Main function and global variables

4.5.1 Thread Network

The main function of the Thread protocol contains many other functions, as shown in the figure 4.8 below. First, it initiates the required functions such as logger, scheduler, timer, leds. Then it checks if any tasks are scheduled to be executed. If not, the sensing device goes into sleep mode and receives an interrupt when a task needs to be executed.

The log_init function initiates the logging module. *The NRF_LOG_INFO* command outputs a debugging string to the J-Link CLI. The scheduler_init function initiates a scheduler that places upcoming tasks to be executed. The timer_init function initiates the timer [24].

```
int main(int argc, char *argv[])
{
    log_init();
150 scheduler_init();
    timer_init();
    leds_init();
    uint32_t err_code = bsp_init(BSP_INIT_LEDS, NULL);
    APP_ERROR_CHECK(err_code);
    while (true)
    {
157     thread_instance_init();
        while (!thread_soft_reset_was_requested())
        {
160         thread_process();
            app_sched_execute();

            if (NRF_LOG_PROCESS() == false)
            {
                thread_sleep();
            }
        }
```

Figure 4.8: Technologies used in the Thread implementation

The *thread_instance_init* function starts the thread and configures the parameters of the detection device, as shown in Figure 4.9. It starts by disabling the sleep mode, so that the detection device is always on, and then automatically starts the device if it is already accepted in a network. The *thread_init* function initiates the Thread with the PAN ID, channel, network name and other selected configurations.

The *thread_cli_init* function initiates a command line interface that is explained in the next chapter. This connection can be used to set the PAN ID, channel name, network name, and other parameters when the detection device is operational [24]. Function *thread_state_changed_callback_set* defines the functions that are executed when the state of the node changes, for example, from a child device to a router device.

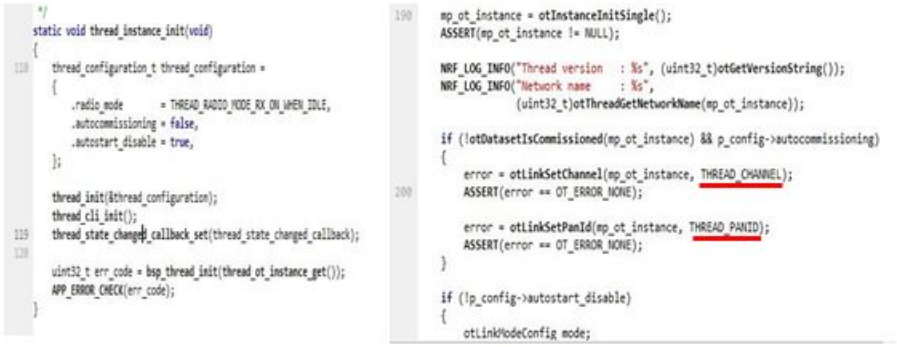

Figure 4.9: Different functions of Thread

4.5.2 Zigbee Network

The figure 4.10 shows the code of the main function and the different functions it contains. So, first the *log function_init* initiate the log module and then the *zb_cli_start* function to define the start of the stack and the *NRF_LOG_INFO* command produces a debugging string on the J-Link CLI.

Function *Zb_nrf52_general_init* is used to initialize the general SoC which contains the function that initializes the timer, the tracer, the random generator and the BCE AES.

In addition, the *Zboss_signal_handler* function allows the application to control a broader set of basic functionality, including assembly commissioning and network formation.

```
   SoC general initialization
*/
void zb_nrf52_general_init(void)
{
  /* Initialise system timer */
  zb_osif_timer_init();
#if defined ZB_TRACE_OVER_USART && defined ZB_TRACE_LEVEL
  /* Initialise serial trace */
  zb_osif_serial_init();
#endif
  /* Initialise random generator */
  zb_osif_rng_init();

  /* Initialise AES ECB */
  zb_osif_aes_init();
```

Figure 4.10: SoC initialization

First, Zigbee stack is ready to work after initializing the BDB, then configuring the network stored in NVRAM with the *ZB_BDB_SIGNAL_DEVICE_FIRST_START* function; then the device will try to rejoin with the *SIGNAL_DEVICE_REBOOT* function.

Then, the Zigbee stack has completed the network orientation procedure and the device may have joined the network, which is indicated by the signal status code with the *ZB_BDB_SIGNAL_STEERING* function. This signal informs the Zigbee trust center application with the *ZB_ZDO_SIGNAL_DEVICE_AUTHORIZED* function of the authorization of a new device in the network, and is typically implemented on the coordinator node. Finally, it signals to the router and coordinator that a PAN ID has been detected.

This figure 4.12 shows the different profile identifiers of the Zigbee for example ZDO profile ID and handler profile ID.

```
case ZB_BDB_SIGNAL_DEVICE_FIRST_START:
    NRF_LOG_INFO("Device started for the first time");
    if (status == RET_OK)
    {
        if (role != ZB_NWK_DEVICE_TYPE_COORDINATOR)
        {
            NRF_LOG_INFO("Start network steering");

case ZB_BDB_SIGNAL_STEERING:
    if (status == RET_OK)
    {
        zb_ext_pan_id_t extended_pan_id;
        char ieee_addr_buf[17] = {0};
        int  addr_len;
#ifndef ZB_ED_ROLE
    case ZB_ZDO_SIGNAL_DEVICE_AUTHORIZED:
        {
            zb_zdo_signal_device_authorized_params_t
            char ieee_addr_buf[17] = {0};
            int  addr_len;

case ZB_NWK_SIGNAL_PANID_CONFLICT_DETECTED:
        {
            NRF_LOG_INFO("PAN ID conflict detected, trying to resolve... ");
```

Figure 4.11: Different functions of Zigbee

```
 */
/*! Profile identifiers */
enum zb_af_profile_id_e
{
  /** ZDO profile id */
  ZB_AF_ZDO_PROFILE_ID   = 0x0000,
  /** Legacy profile */
  ZB_AF_LEGACY_PROFILE1_ID   = 0x0101,
  /** Legacy profile */
  ZB_AF_LEGACY_PROFILE2_ID   = 0x0102,
  /** Legacy profile */
  ZB_AF_LEGACY_PROFILE3_ID   = 0x0103,
  /** HA profile id */
  ZB_AF_HA_PROFILE_ID   = 0x0104,
  /** Legacy profile */
  ZB_AF_LEGACY_PROFILE4_ID   = 0x0105,
  /** Legacy profile */
  ZB_AF_LEGACY_PROFILE5_ID   = 0x0106,
  /** Legacy profile */
  ZB_AF_LEGACY_PROFILE6_ID   = 0x0107,
```

Figure 4.12: Different Profile ID

4.5.3 Bluetooth Low Energy Network

The main function of the Bluetooth Low Energy protocol contains many other functions, as shown in the figure 4.13 below. The *NRF_LOG_INFO* command outputs a debugging string to the J-Link CLI. First, start and connect the BLE device, then advertising a function that simulates data such as *pm_evt_handler*, *battery_level_update* and

gap_params_init and *gatt_init*.

```
778 int main(void)
    {
780     bool erase_bonds;

        core_init();

        buttons_leds_init(&erase_bonds);

        ble_init();

        flashlog_init();

790     battery_measurement_init();

        APP_ERROR_CHECK(nrf_cli_ble_uart_service_init());

        NRF_LOG_INFO("BLE CLI example started.");

        task_manager_start(idle_task, (void *)erase_bonds);
    }
```

Figure 4.13: Technologies used in the BLE implementation

The figure 4.14 below shows an example of the different code functions that used in BLE.

```
static ble_uuid_t m_adv_uuids[] =
{
    {BLE_UUID_HEART_RATE_SERVICE,           BLE_UUID_TYPE_BLE},
    {BLE_UUID_BATTERY_SERVICE,              BLE_UUID_TYPE_BLE},
    {BLE_UUID_DEVICE_INFORMATION_SERVICE,   BLE_UUID_TYPE_BLE}
};

void advertising_start(bool erase_bonds)
{
    if (erase_bonds == true)
    {
static void pm_evt_handler(pm_evt_t const * p_evt)
{
    pm_handler_on_pm_evt(p_evt);
    pm_handler_flash_clean(p_evt);

static void gap_params_init(void)
{
    ret_code_t              err_code;
    ble_gap_conn_params_t   gap_conn_params;
    ble_gap_conn_sec_mode_t sec_mode;
static void gatt_init(void)
{
    ret_code_t err_code = nrf_ble_gatt_init(&m_gatt, gatt_evt_handler);
    APP_ERROR_CHECK(err_code);
}
```

Figure 4.14: Different functions of BLE

Chapter 5

Test and Simulation

This chapter contains 3 parts of simulation of the 3 technologies, starting with the Thread network, then ZigBee and BLE.

5.1 Mesh Network

5.1.1 Network layout

The nRF52 SoC comes with the Thread, Zigbee and Bluetooth Low Energy protocol stack. Thus, to compile and run software on nRF52840 devices, Segger's Embedded Studio IDE, nRF connect and J-link are used to debug and run code on the devices. On the host device PUTTY, a serial terminal, is used for Thread and Zigbee protocols to control the device running the mesh network and for Bluetooth Low Energy the RFC is used. The application opens the COM port and activates an interactive command line interface (CLI) to get information from the three networks using security and reliability.

Figure 5.2 shows an overview of the Thread, Zigbee and BLE network layout.

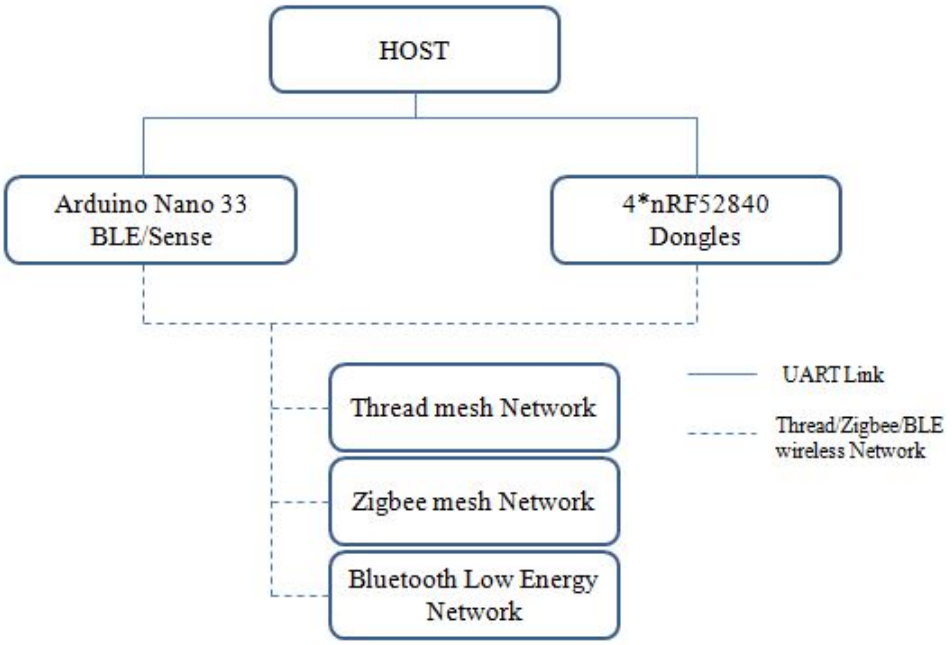

Figure 5.1: Block Diagram for creating and controlling three networks from host devices

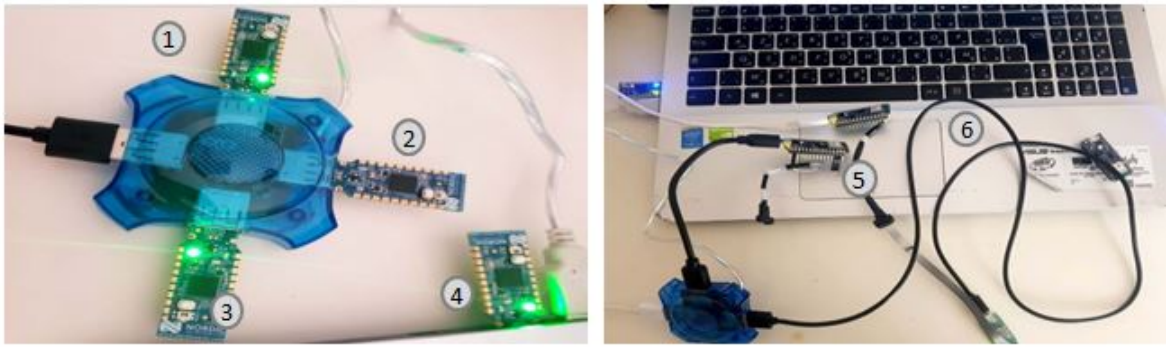

Figure 5.2: Setting up of devices

5.2 Thread Network

This diagram describes the steps of creating a network with boards like Arduino Nano 33 BLE soldered with j-Link, and nRF52840 Dongles that are used as Leader, Routers and child to make a Thread network.

5.2.1 Network configuration

In order for the devices to join and communicate with the Thread network, certain network configurations are required. All network configurations are done from a separate Windows laptop with remote control via Putty.

5.2.2 Network co-processor

To flashing the Network co-processor to nRF52840, first, add HEX file and select the DFU bootloader in nRFConnect app for desktop and add the devices to the network. The channel determines what frequency to be used within the network. All channels are close to the chosen frequency band, differentiating with a small range. For communication in the Thread network, devices need to use the same channel. The XPAN ID together with the network name is a unique identifier for the network and is used in case of name collision. The PAN ID is a unique identifier for communication within the Thread network and used in case of multiple Thread networks within radius running on the same channel. As shown in the table 5.1 of the Leader configuration.

User name	Value
Panid	0x1234
ifconfig up	Done
State	Leader
ipaddr	fdde:ad00:beef:0:0:ff:fe00:fc00 fdde:ad00:beef:0:0:ff:fe00:dc00 fdde:ad00:beef:0:f633:979d:cf70:c041 fe80:0:0:0:b458:45c4:7753:8182
Network name	OpenThread
Master Key	00112233445566778899aabbccddeeff
Extended PAN	dead00beef00cafe
Channel	11
Extended Mac	b65845c477538182
Thread Version	2

Table 5.1: Thread network configuration

There are four IP addresses such as ALOC, RLOC, ML-EID and LLA. First of all, the anycast leader address is used to route traffic to a Thread interface and identifies the location of multiple interfaces in a Thread partition. While, the Routing Locator (RLOC) is created when the device is attached to the network and the first four bytes are the mesh prefix, this address changes if the topology changes.

The Mesh-local EID address is used to communicate with the other interface in the same wired network and the Link-Local address is created with the MAC address. It is not used to communicate between nodes, but can still be used between two nodes if there is only one link and one radio transmission.

- Leader Anycast Locator(ALOC): fdde:ad00:beef:0:0:ff:fe00:fc00
- Routing Locator(RLOC): fdde:ad00:beef:0:0:ff:fe00:dc00
- Mesh-Local EID(ML-EID): fdde:ad00:beef:0:f633:979d:cf70:c041
- Link-Local Address(LLA): fe80:0:0:0:b458:45c4:7753:8182

5.2.3 Join devices

Figure 12 shows the commands for joining a network with two devices. First, the device gets its PAN ID which is used by the network co-processor as shown in Table 5.2.

When the network coprocessor has added the devices, it must use the correct configuration and settings: the PAN ID and channel are set identically to the network it wants to join. Use the ifconfig up command to activate the network, and the thread start command to activate communication within the Thread network.

The state of the device will depend on the requirements of the network and the specifications of the device. This device is a router. To confirm that the child device's communication is properly routed. In addition, the child device is routed to its parent a router or leader device before entering the border router, which is how information should travel from a child device.

User name	**COM11**	**COM12**	**COM14**
PAN ID	0x1234	0x1234	0x1234
ifconfig up	Done	Done	Done
State	Router	Router	Router
Network Name	OpenThread	OpenThread	OpenThread
Ext Mac	3ef4c90dbe2be0c0	36a7aeca27b83e6f	46c5dde91469ddf4
Master Key	c00112233445566778899aabbccddeeff		
Extended PAN	dead00beef00cafe		
Channel	11		

Table 5.2: Characteristics of three joining devices

The table 5.2 shows the three joining devices and their characteristics.

I) IPaddr of 3 Routers

Router1 (COM11) :

- fdde:ad00:beef:0:0:ff:fe00:3c00

- fdde:ad00:beef:0:d78b:clde:8cc6:cf5e

- fe80:0:0:0:b458:dde9:1469:ddf4

Router2 (COM12) :

- fdde:ad00:beef:0:0:ff:fe00:2c00

- fdde:ad00:beef:0:1331:a7fe:b2f9:90b0

- fe80:0:0:0:3cf4:c90d:be2b:e0c0

Router3 (COM14) :

- fdde:ad00:beef:0:0:ff:fe00:6800

- fdde:ad00:beef:0:9e4f:6e44:6439:b5aa

- fe80:0:0:0:34a7:aeca:27b8:3e6f

All routers maintain router table.Whenever a device connects to a network, its information is shared with its parent and all other routers also get updated. The routers exchange the cost of routing to other routers in the Thread network with other routers in a compressed format using Mesh Link Establishment(MLE) which are used to establish and configure secure radio links, detect neighboring devices and maintain the costs of routing between devices in the Thread network. As shown in the figure 5.4 below, the routing table for the various devices is populated with a compressed form of a local mesh address for each router and the appropriate next hop. It also contains the path age and cost, which define the number of nodes to traverse when sending data from the current node to a destination node.

✓ **COM13 : LEADER**

ID	RLOC	Next hop	Path Cost	LQ In	LQ Out	Age	Extended Mac
11	0x2c00	15	1	3	3	15	3ef4c90dbe2be0c0
15	0x3c00	11	1	3	3	4	46c5dde91469ddf4
26	0x6800	11	1	3	3	13	36a7aeca27b83e6f

✓ **COM11 :Router 1**

ID	RLOC	Next hop	Path Cost	LQ In	LQ Out	Age	Extended Mac
11	0x2c00	55	1	3	3	13	3ef4c90dbe2be0c0
15	0x3c00	63	0	0	0	0	46c5dde91469ddf4
26	0x6800	55	1	3	3	4	36a7aeca27b83e6f
55	0xdc00	11	1	3	3	15	b65845c477538182

✓ **COM12 :Router 2**

ID	RLOC	Next hop	Path Cost	LQ In	LQ Out	Age	Extended Mac
11	0x2c00	63	0	0	0	0	3ef4c90dbe2be0c0
15	0x3c00	55	1	3	3	28	46c5dde91469ddf4
26	0x6800	55	1	3	3	4	36a7aeca27b83e6f
55	0xdc00	11	1	3	3	10	b65845c477538182

✓ **COM14 :Router 3**

ID	RLOC	Next hop	Path Cost	LQ In	LQ Out	Age	Extended Mac
11	0x2c00	55	1	3	3	29	3ef4c90dbe2be0c0
15	0x3c00	55	1	3	3	20	46c5dde91469ddf4
26	0x6800	63	0	0	0	0	36a7aeca27b83e6f
55	0xdc00	11	1	3	3	3	b65845c477538182

Figure 5.3: Routing table

Figure 5.3 shows the Leader's neighbor table that contains the Rloc address and extended address of each device, as well as the received signal strength indicator RSSI and the age that defines the time that elapses between device binding.

Role	RLOC16	Age	Avg RSSI	Last RSSI	R	S	D	N	Extended Mac
R	0x2c00	25	-37	-37	1	0	1	1	3ef4c90dbe2be0c0
R	0x3c00	16	-45	-42	1	0	1	1	46c5dde91469ddf4
R	0x6800	24	-27	-27	1	0	1	1	36a7aeca27b83e6f

Figure 5.4: Leader routing table

The network contains 1 Leader and three routers. From any node, can check the routing table or its neighbors. Now add a new device to the first router in COM 11 and this device has the role of a child. The figure 5.5 below shows the neighbor table of the first router, as seen, it has added the new child device with its parameters.

Role	RLOC16	Age	Avg RSSI	Last RSSI	R	S	D	N	Extended Mac
C	0x3c01	6	-23	-23	1	0	1	1	06dae04756c56f6
R	0x2c00	7	-24	-24	1	0	1	1	3ef4c90dbe2be0c0
R	0x6800	5	-25	-25	1	0	1	1	36a7aeca27b83e6f
R	0xdc00	5	-53	-50	1	0	1	1	b65845c477538182

Figure 5.5: Routing table of the router 1

5.2.4 Send messages with UDP and CoAP

1. UDP Command

One of the application services provided by OpenThread is the User Datagram Protocol (UDP), a transport layer protocol. An application built on top of OpenThread can use the UDP API to transmit messages between nodes on a Thread network or to other devices on an external network (such as the Internet, if the Thread network has a border router).

Figure 5.6: UDP command

As shown in the figure 5.6 , first start UDP and bind it to a socket for any IPv6 address, then switch to the other device and start UDP, and connect to the socket that is configured on the first device, then send a message from the first device and the UDP message was received in the other router. The UDP connection should be active between the two nodes.

To get the time between sending a packet and receiving a response, use the ping command with the UDP command. The graphs represented on the Figure 5.7 shows the time in (ms) between 3 devices when they use different bytes.

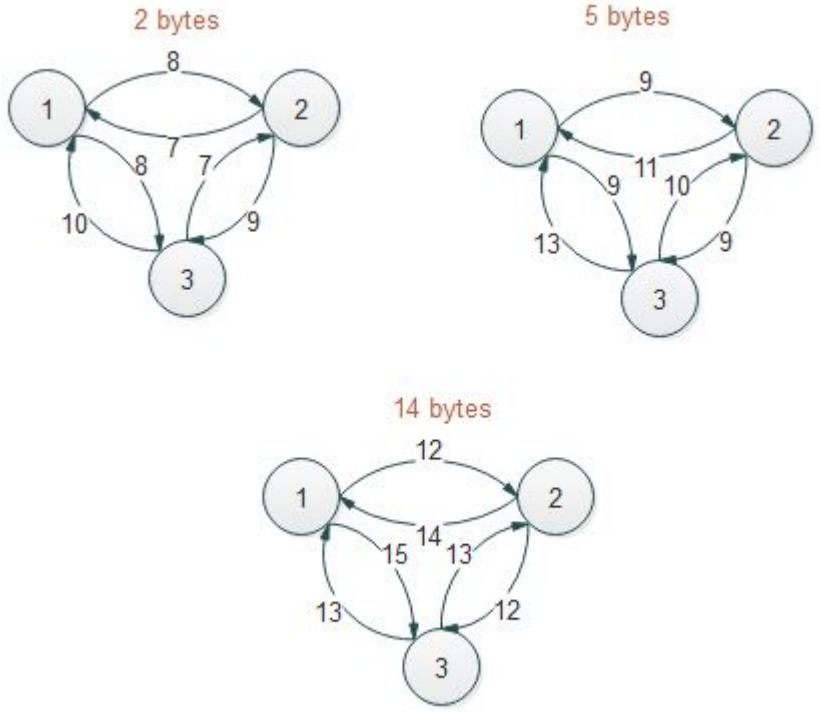

Figure 5.7: Graph of transmission of different data

2. CoAP Command

CoAP is designed for use with constrained nodes. CoAP has four methods: GET, POST, PUT, and DELETE. CoAP defines request and response messages as shown in figure 5.8 below method to send message with "GET"[32].

Distinct from HTTP, CoAP messages are encoded in a simple binary format. The feature enables CoAP to choose the transmission reliability from the application layer. Unlike HTTP over TCP, CoAP messaging is bound to User Datagram Protocol (UDP), which does not provide a retransmission mechanism. The CoAP request can be sent on the default CON mode so that an exponential retransmission mechanism can ensure the reliability of data transmission.

Figure 5.8: CoAP configuration

5.3 Zigbee

As described in the diagram 5.2 above, the steps of creating a Zigbee network. Then, some network configurations must be used to join and communicate in the Zigbee mesh network. In this section, the configuration is done using the PUTTY serial terminal with the following settings:

1) Baud rate: 115200

2) 8 data bits

3) 1 stop bit

4) No parity

5) HW flow control: None

The network was creating with boards like Arduino Nano 33 BLE soldered with J-Link, and nRF52840 Dongles that are used as coordinator, Routers and End devices.

5.3.1 Create zigbee network

In nRFConnect app for desktop, add HEX file and select the DFU bootloader and add the devices to the network. First, in serial terminal, configure the devices to be in the same network. One device should be the coordinator, and the others can be routers or end devices in a Zigbee network. In this case, will have three routers configured to send data to the coordinator. First, in the serial terminal, configure the devices to be in the same network. One device should be the coordinator, and the others can be routers or end devices in a network. In this case, three routers are configured to send data to the coordinator.

The following table explain the operations and commands performed by coordinators, routers to form or join a network. The coordinator is the only device that can start a network, so every Zigbee network must have a coordinator. It is responsible for selecting an unused operating channel, PAN ID, security and stack profile for a network that, routers must discover and join a valid Zigbee network. When a router joins a network, it receives a 16-bit address randomly chosen by the device that authorized the connection.

Once a router joins a Zigbee network, it remains connected to the network on the same channel and PAN ID until it is used. The table 5.3 shows the different characteristics of the different joining devices:

User name	**COM17**	**COM16**	**COM18**	**COM19**
bdb start	Done	Done	Done	Done
PAN ID	9943	9943	9943	9943
Role	Coordinator	Router1	Router2	Router3
Short address	0000	7059	5E85	5Dc4
eui64 address	f4ce36e27cf78d03	4ce3679f2448239	f4ce36b34448da7b	f4ce3614e2d676ab
Channel	16	16	16	16
Nwk key	00112233445566778899aabbccddeeff			

Table 5.3: Characteristics of three joining devices

5.3.2 Ping Command

First, started to use the ping command to show the time between sending and receiving a packet in 3 different devices: a coordinator, a router and an end device with 4 different sizes:

Type	**Node1: Coordinator**	**Node2: Router 1**
eui64	f4ce36e27cf78d03	f4ce3679f2448239
Short address	0000	7059

Table 5.4: Address of the 2 devices

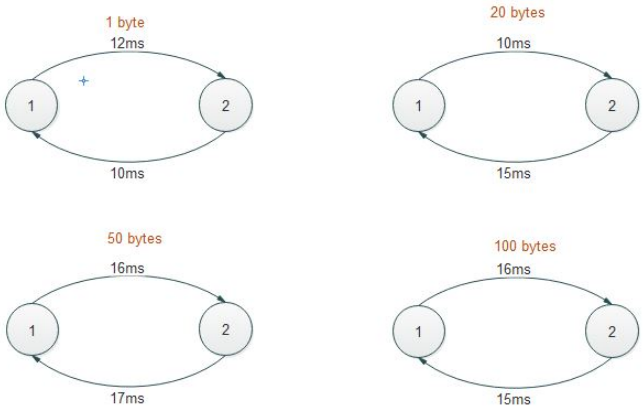

Figure 5.9: Ping times between coordinator and router

Now, used 3 devices with the same previous size of packet. As shown in the figure below

Type	**Node1: Coordinator**	**Node2: Router 1**	**Node3: Router 2**
eui64	f4ce36e27cf78d03	f4ce3679f2448239	f4ce36b34448da7b
Short address	0000	7059	5E85

Table 5.5: The address of 3 devices

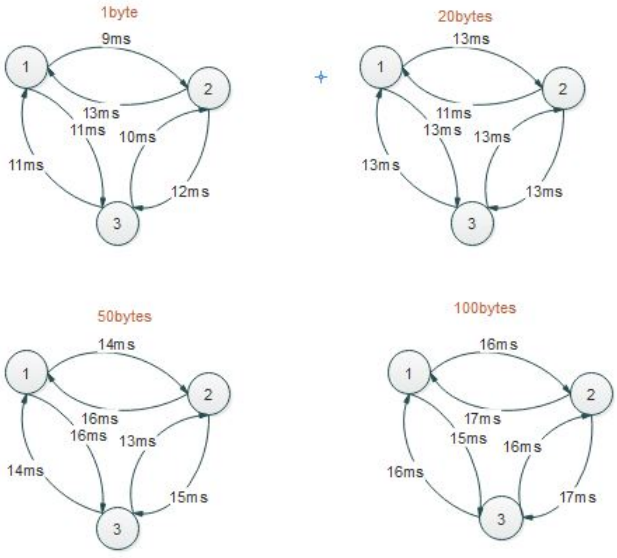

Figure 5.10: Ping times between coordinator and 2 routers

The following part is when used 4 devices that one coordinator and three routers and also with the same packet size.

Type	**Node1: Coordinator**	**Node2: Router 1**	**Node3: Router 2**	**node3: Router 3**
eui64	f4ce36e27cf78d03	f4ce3679f2448239	f4ce36b34448da7b	f4ce3614e2d676ab
Short address	0000	7059	5E85	5Dc4

Table 5.6: The address of 3 devices

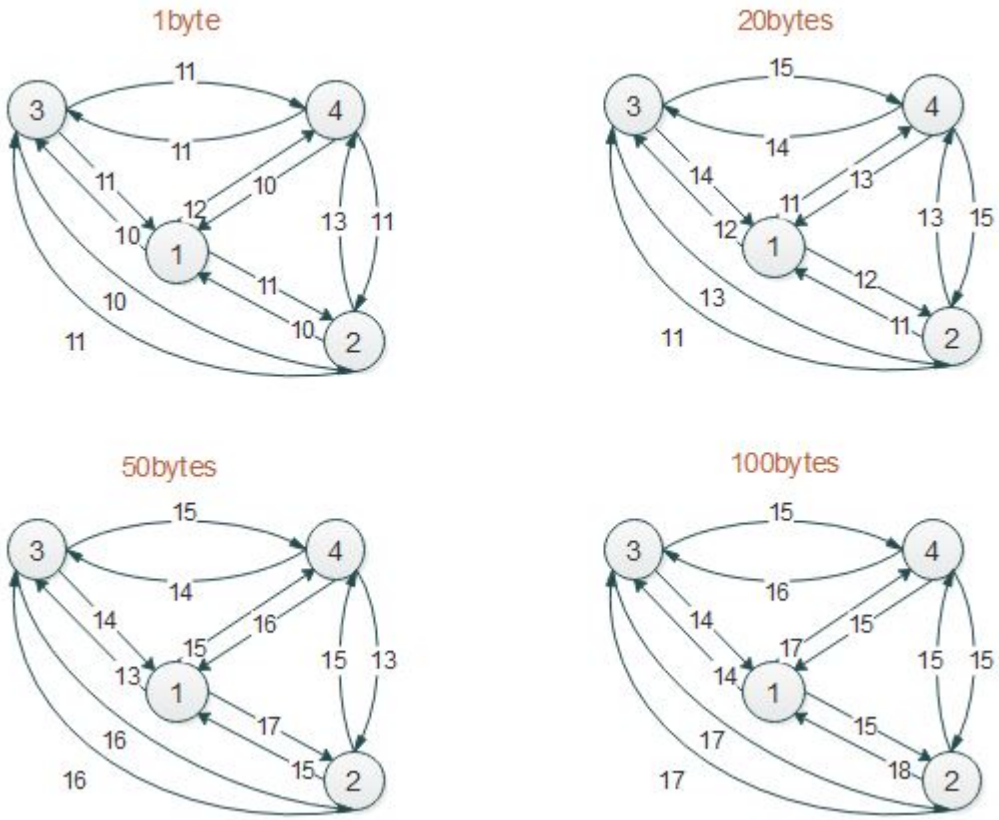

Figure 5.11: Ping times between coordinator and 3 routers

5.3.3 Bind Table

Binding is an action that defines the relationships between two devices at the application layer, specific endpoints, and a cluster identifier. It provides a mechanism for attaching an endpoint on one node to one or more endpoints on another node, and can even be intended for groups of nodes. So, as shown in the figure 5.12 the bind table between two devices that implements the following mapping such as Source Address, Source Endpoint, Cluster ID Destinations address and Destination Endpoint.

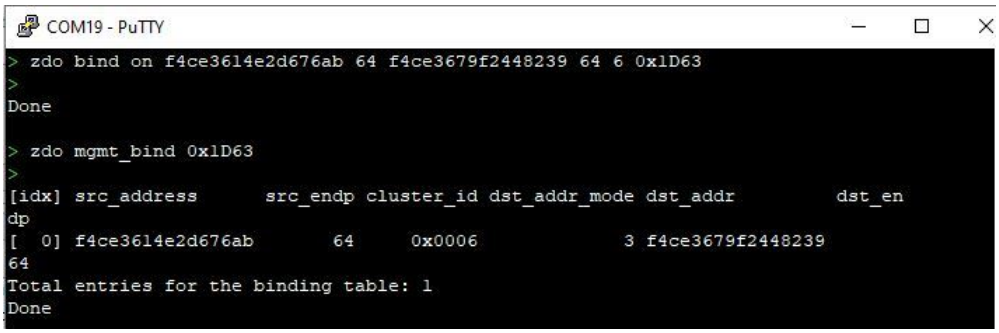

Figure 5.12: Zigbee bind table

Example of command to control lighting devices

The CLI example is used to control Zigbee lighting devices.It has different steps, first it has to discover the devices that have joined the network using the zdo match_desc command.It can use the Zigbee Light Link (0xc05e) or Home Automation (0x0104)profile, depending on the type of devices that have in the network [24] after searching the eui64 address of the device using its network address.

The zdo simple_desc_req command to discover the different clusters as shown in figure 5.13 below the structure command of lighting devices

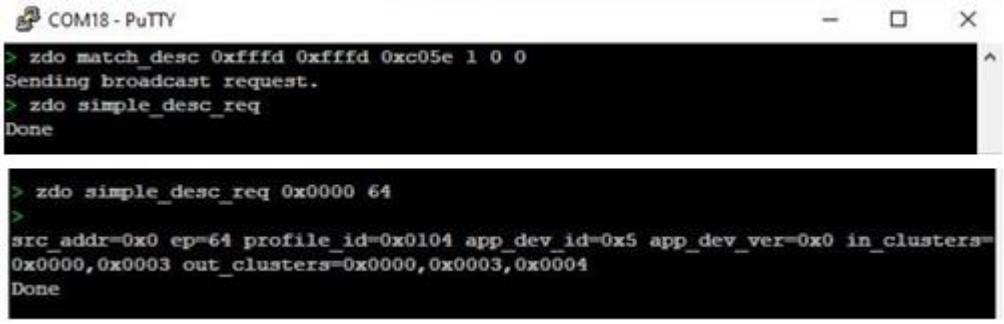

Figure 5.13: Different Command

5.4 BLE Network

5.4.1 Network layout

The Arduino Nano 33 BLE featuring a lot more powerful processor nRF52840, and it comes with the BLE mesh protocol stack. Thus, to compile and run software on nRF52840 devices, Segger's Embedded Studio IDE and the J-link are used to debug and run code on the development kits. On the host device is a nRFconnect application for desktop.

5.4.2 Creating a Bluetooth Low Energy Network

The nRF Connect application on the desktop supports a few applications. It started out as a simple application, but is now becoming the single hub for all desktop-based nRF applications. As of the writing of this thesis, the applications included in the desktop nRF Connect app are:

* Bluetooth Low Energy application
* nRF Cloud Gateway application
* Power Profiler
* Programmer (experimental)

This work focuses on using the Bluetooth Low Energy application and Programmer. First select the nRF52840 program, return to SES and load the SoftDevice and the

application from there, as shown in the figure 5.14.

Bluetooth Low Energy supports two modes of operation: as a central BLE device that can discover advertising devices (broadcasters and devices) and as a BLE device that advertises, allows connections and exposes a GATT server. The GATT can be customized with services and features adopted by SIG. Next, perform a search for advertising BLE devices and connect to a device by clicking the connect button next to a device in the search results. Once connected to a device, browse and navigate through the various services and features exposed by the GATT server on the device.

The Bluetooth Low Energy application of nRF Connect allows connecting to several devices at the same time, and they will all be displayed in the same window.

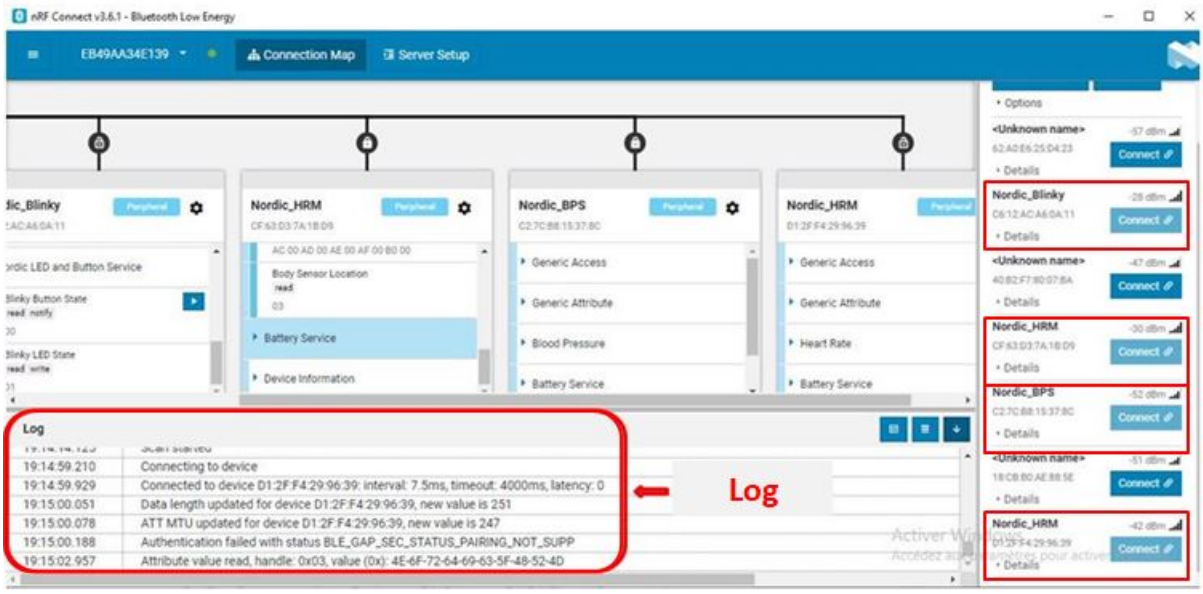

Figure 5.14: Connected to multiple peripherals

In the figure 5.14 above, five devices are connected so that each device can send data to any other, and these are received in the "nrf connect" application.

For example, after connecting the devices to the mesh network, to change the status of the LED of one device, all other devices are notified in the same way in the USB Bootloader of nrf connect application. Also, access to the characteristic value of a device

to all other devices, and each device can get the characteristic value in BLE.

Another feature available is the ability to modify connection properties such as connection settings, pairing and disconnection.

Once connected to a device, the dongle can interact with the device's GATT server. For example, it can read, write and enable/disable notifications and directions.

Each device have MAC address as shown in the table below:

Application	**Mac Address**
Heart Rate service	CF:63:D3:7A:1B:D9
Blood pressure service	C2:7C:B8:15:37:BC
Blinky	C6:12:AC:A6:DA:11

Table 5.7: The MAC address

The next section shows the examples that used such as Heart Rate service, Blink and Blood pressure application examples.

1. Heart Rate service application

Run the Heart Rate Service example from the SDK in nRF connect and after the device is connected, the nRF Connect application displays the full table of attributes. Each row is an attribute, and they are grouped into services and with characteristics as subgroups. The following figure 5.15 shows the list of services displayed in the nRF Connect BLE application when connected to an nRF52 dongle running the HRS example.

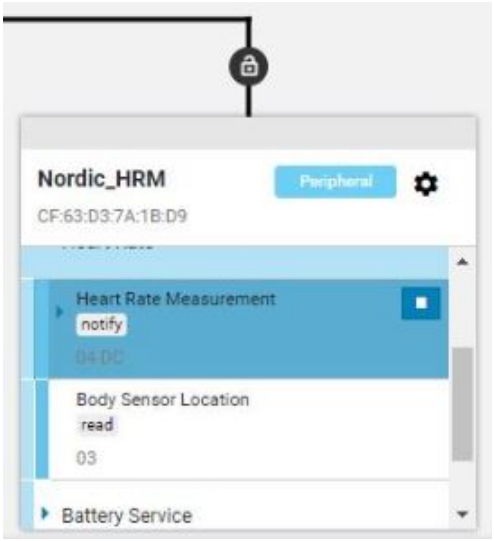

Figure 5.15: Heart Rate peripheral

Thus, the example uses two services that are typically used in a heart rate sports watch for example, a heart rate service and a battery service. The device also has a device information service. The attribute type (UUID) for the heart rate service is 0x180D.

The heart rate service also has two features:

- A Heart Rate Measurement feature, which contains the heart rate value and is used to send a heart rate measurement.

- An optional body sensor location feature, which is used to describe the intended location of the heart rate measurement for the device. The attribute type (UUID) of the heart rate measurement feature is 0x2A37, and the "notify" property is also specified with the value 0x04DC. This means that notification is currently enabled. Finally, the attribute type of the client feature configuration is 0x2902.

2. Blood pressure application

The blood pressure application, which includes the two blood pressure profile services: Blood Pressure Service and Device Information Service. In addition, the use of the battery service is also demonstrated.

When the application starts, a timer to generate battery measurements is started. Observe that the status BSP_INDICATE_ADVERTISING is indicated. Connect to the

device from nRF Connect, then select "Pair" to link to the device.

After linking, observe that the BSP_INDICATE_CONNECTED status is indicated and the services are displayed in the connected device, then start receiving values for the blood pressure service and battery service when click the "Play" button. An indication of the blood pressure measurement is received. In addition, battery level notifications are received every two seconds as shown in the figure 5.16 below in the BPS application:

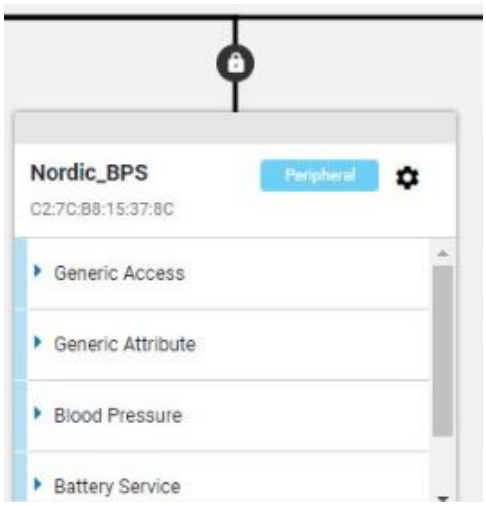

Figure 5.16: Blood pressure peripheral

The application will stop the advertisement after 3 minutes and will go into system shutdown mode.

3.Blinky

The BLE Blink application uses the LED button service server.This is a small custom service that is used to toggle LEDs and receive button statuses from the nRF52 development board. First, observe that LED 1 is On this indicates that the application is advertising, then connect to the device from nRF Connect app and observe that LED 2 is On and LED 1 is off, this indicates that the connections are established. Notifications are received on the Feature button (0x1324). Also, this example has 'written' features can write '01' to observe that the LED is on in the dongle and '00' the LED is off.

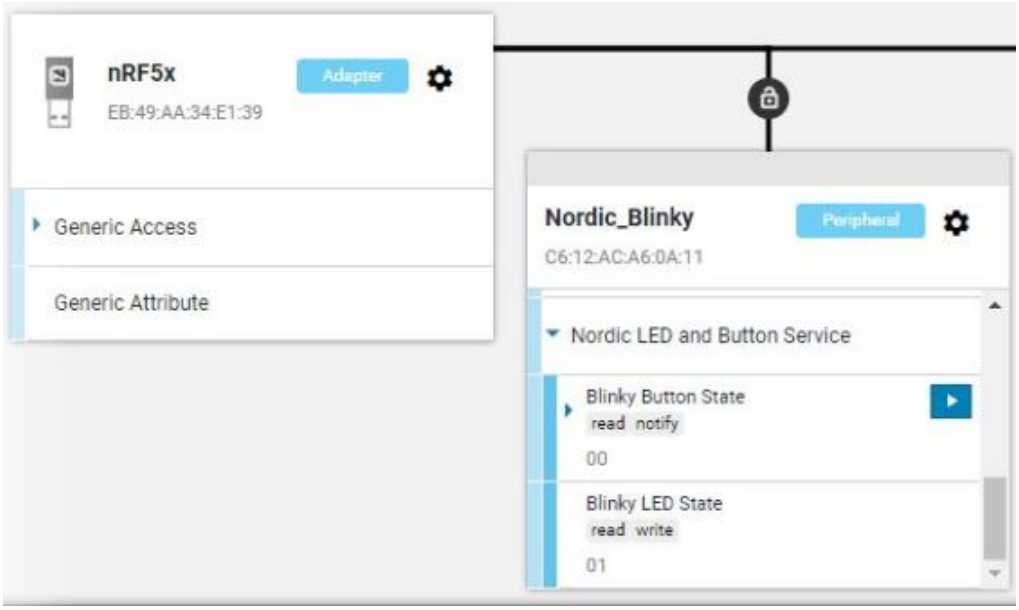

Figure 5.17: Blink peripheral

5.4.3 Different parametres in BLE

This section presents the parameters and properties of the connection between the central controller and the peripheral, such as connection interval, slave latency, supervision delay, link status, pairing status.

Parametres	**Blink**	**Heart Rate service**	**Blood Pressure**
Connection interval	100ms	400ms	500ms
slave latency	0ms	0ms	0ms
Time out	4000ms	4000ms	4000ms

Table 5.8: Different parameters

Chapter 6

Conclusion and Future Work

This thesis analyzed and implemented a mesh network with a good overview of three low-power wireless protocols: Bluetooth Low Energy, Thread and Zigbee, including the comparison of their main features and behaviors in terms of various metrics, network topology, security, quality of service and power consumption. Also test hardware and software compatibility with all three technologies using the nrf52840 dongle and the Arduino Nano 33 BLE.

To choose the best technology is no simple answer because there are fundamental architectural differences between Zigbee, Thread and Bluetooth mesh. Zigbee and Thread can use flooding, but generally use a routing mesh to minimize network overhead that can interfere with messaging. Bluetooth mesh uses flooding mesh, but allows devices to be configured to act as routers to reduce the impact of flooding. Zigbee and Thread networks include routing nodes and end nodes. The routing table is established when the mesh network is created. In this way, a node can communicate efficiently with another node by sending messages along a specific route through the mesh. This has a positive effect on mesh throughput and can reduce latency as the mesh grows. So, a routing mesh is better than a flooding mesh because it provides more efficient communications and predictable performance. On the other hand, routing is more difficult to implement.

In addition, the choice of the best technology with the mesh network depends on the end application or ecosystem. There are many established ecosystems.If the ecosystem

has not been specified for the application, many protocol choices are available.

In conclusion, the Thread network is an optimal choice because it is easier to control IoT products and systems from personal devices such as mobile phones or tablets. In addition, Thread technology is expected to grow significantly in the next few years. It has great potential and has the added benefits of Wi-Fi like functionality.

Some hardware is relatively new, there was a lack of documentation, this mean that some issues were difficult to resolve. It was also difficult to differentiate between commercial statements and objective information, but in the end, the implementation of a mesh network was achieved after a lot of work.

There are many interesting ideas that can be implemented in the future, such as evaluating the three networks in a more general setting with a variety of end devices, a multi-hop configuration, and using a set of border routers to determine the effect of distance variation and adding real sensors: temperature or light sensors and connecting them to the user interface.

Bibliography

[1] satista, *satista*, https://www.statista.com/.

[2] *Internet of ThingsWireless Networks*, https://www.ccontrols.net.

[3] A. Behr, *Best Uses of Wireless IoT Communication Technology*, https://industrytoday.com.

[4] A. Patel and T. A. Champaneria, "Study and comparison of various internet of things protocols", 2015.

[5] M.Andersson, "Short-range Low Power Wireless Devices and Internet of Things (IoT)", *1st ed. connectBlue*, 2014.

[6] P. S. Neelakanta and H. Dighe, "Robust factory wireless communications: a performance appraisal of the Bluetooth/spl trade/and the ZigBee/spl trade/colocated on an industrial floor", in *IECON'03. 29th Annual Conference of the IEEE Industrial Electronics Society (IEEE Cat. No. 03CH37468)*, IEEE, vol. 3, 2003, pp. 2381–2386.

[7] S. Chakkor, E. A. Cheikh, M. Baghouri, and A. Hajraoui, "Comparative performance analysis of wireless communication protocols for intelligent sensors and their applications", *arXiv preprint arXiv:1409.6884*, 2014.

[8] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4", in *2012 IEEE wireless communications and networking conference workshops (WCNCW)*, IEEE, 2012, pp. 232–237.

[9] S. Shams, *Top IoT Communication Protocols Updated 2021 [ZigBee, NFC, And More*, https://hashstudioz.com, 25 August 2020.

[10] A. G. Ata Elahi, *ZigBee Wireless Sensor and Control Network*, https://www.informit.com.

[11] pulkitagarwal03pulkit, *Advantage and Disadvantage of Mesh Topology*, https://www.geeksforgeeks.org.

[12] A. Dessales, *Conception d'un réseau de capteurs sans fil*, 1st ed. fRANCE, 2011.

[13] *" BLE System Architecture"*, https://embeddedcentric.com/introduction-to-bluetooth-low-energy-bluetooth-5/.

[14] *"Bluetooth low energy"*, https://embeddedcentric.com/introduction-to-bluetooth-low-energy-bluetooth-5/.

[15] "Mesh Profile", in *Bluetooth SIG*, Jan. 2019, pp. 1–333.

[16] *What is Zigbee Technology? Architecture, Topologies and Applications*, https://www.electronicshub.org, November 17, 2017.

[17] "Zigbee Alliance", in *Zigbee Specification*, Aug. 2015, pp. 1–542.

[18] S. Shams, *MESH NETWORKS FOR LIGHTING CONTROL APPLICATIONS*, https://adlt.com.sg, June/July 2016.

[19] A. K. Ramachandran, "Multi-Protocol Device Commissioning Framework for IoT Mesh Networks",

[20] V. Rudresh, *Mesh network for lighting control applications*, https://research.kudelskisecurity.com, November 8, 2017.

[21] "Thread Network Fundamentals", in *thread group*, May 2020, pp. 1–22.

[22] J. Postel, *Internet Standard "User Datagram Protocol"*, 28 August 1980.

[23] *""IETF, CoAP"*, Available:https://tools.ietf.org/html/rfc7252..

[24] *" Nordic Semi conductor"*, https://infocenter.nordicsemi.com/.

[25] *“ Qorvo ″*, https://www.qorvo.com/products/p/QPG6095.

[26] *“ silicon-labs ″*, https://ar.mouser.com/new/silicon-labs/silabs-efr32mg21-socs/.

[27] *“ Texas Instruments ″*, https://www.ti.com/product/CC2538/.

[28] *“ NXP ″*, https://www.nxp.com/products/wireless/thread/k32w061-41.

[29] https://store.arduino.cc.

[30] “nRF52840 Dongle”, in *Nordic semiconductor*, Jun. 2018, pp. 1–15.

[31] *“J-Link RTT Viewer″*, https://www.segger.com/products/debug-probes/j-link/technology/real-time-transfer/rttviewer/.

[32] *“ Open Thread″*, https://github.com/openthread/openthread/blob/main/src/cli/README_COAP.md.

Appendix A

One of the most used concepts in the development of applications using BLE, is the concept of the concept of characteristics, which, in turn, contains some properties that are discussed below:

proprieties	**Value**	**Description**
Broadcast	0x01	Permits broadcasts of the characteristics value
Read	0x02	Permits read of the characteristic value
Write without response	0x04	Permits write of the characteristic value without response
Write	0x08	Permits write of the characteristic value with response
Notify	0x10	Permits notifications of the characteristic value without acknowledgment
Indicate	0x20	Permits indications of the characteristic value with acknowledgment
Authenticated signed writes	0x40	Permits signed writes to the characteristic value
Extended Proprieties	0x80	Additional characteristics proprieties are defined in the characteristics Extended proprities Descriptor

Table A.1: the bit field of the characteristic properties