VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Mohamed Mahmud Ali

DEVELOPMENT AND OPTIMIZATION OF A BLUETOOTH LOW ENERGY (BLE) MESH NETWORK FOR ENVIRONMENTAL MONITORING AND ALERTING

Technology and Communications

2024

ACKNOWLEDGEMENTS

I would like to express my gratitude to Gao Chao for his invaluable guidance and assistance throughout the completion of this thesis. Additionally, I extend my heartfelt thanks to all my teachers at Vaasa University of Applied Sciences for imparting the knowledge and skills that have been crucial in this journey.

Mohamed Mahmud Ali

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Mohamed Mahmud Ali
Title	Development and Optimization of a Bluetooth Low Energy (BLE) Mesh Network for Environmental Monitoring and Alerting
Year	2024
Language	English
Pages	52
Name of Supervisor	Gao Chao

This thesis presents the design, implementation, and optimization of a Bluetooth Low Energy (BLE) Mesh network using Nordic Semiconductor nRF52-DK boards for real-time environmental monitoring. The research focuses on developing firmware and integrating communication protocols to ensure efficient and reliable data transfer across the network. The key objectives include optimizing the network's scalability and reliability, evaluated through practical experiments.

The study systematically assesses the Packet Loss Ratio (PLR) at each network node, identifying potential bottlenecks and inefficiencies that could hinder large-scale deployments. To this end, four scenarios were tested: (1) a single link performance with one broadcaster and 3 listeners, (2) a chain topology with one broadcaster, three relays, and one listener, (3) a mesh network with two relays, two listeners and one broadcaster, and (4) a hybrid topology with multi-hop configuration with redundant communication pathways. These scenarios were designed to analyze the impact of relay effectiveness, multipath delivery, and multi-hop communication on network performance in both outdoor and indoor environments.

The results indicate that the optimized BLE Mesh network architecture significantly enhances data accuracy, reliability, and scalability, making it well-suited for environmental monitoring applications. The practical experiments validate the network's real-world functionality and demonstrate its potential for integration into larger IoT systems.

In conclusion, this thesis advances BLE Mesh networking technology by providing scalable, reliable solutions for environmental monitoring, with broader implications for industries such as smart buildings and industrial facilities.

Keywords	Bluetooth low energy (BLE), Internet of Things (IoT), nRF52-DK, Packet loss ratio (PLR)

CONTENTS

LIST OF FIGURES AND TABLES

ABBREVIATIONS

ARM	Advanced RISC Machines
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
ECDH	Elliptic-curve Diffie-Hellman
FHSS	Frequency Hopping Spread Spectrum
GATT	Generic Attribute Profile
GAP	Generic Access Profile
GPIO	General Purpose Input/Output
IoT	Internet of Things
ISM	Industrial, Scientific, and Medical
I2C	Inter-Integrated Circuit
LED	Light-Emitting Diode
LPN	Low Power Nodes
NFC	Near Field Communication
OOB	Out Of Band
PLR	Packet Loss Ratio
PDU	Protocol Data Unit
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
SDK	Software Development Kit
SoC	System on Chip
SEQ	Sequence Number
SIG	Bluetooth Special Interest Group
SRC	Source
TDMA	Time Division Multiple Access
TTL	Time To Live
UUID	Universally Unique Identifier
UART	Universal Asynchronous Receiver-Transmitter
WIFI	Wireless Fidelity

1 INTRODUCTION

The rapid advancement of IoT technologies has transformed various sectors, including environmental monitoring. Bluetooth Low Energy Mesh (BLE Mesh) stands out for its low power consumption, cost-effectiveness, and widespread use in IoT applications. This thesis aims to leverage BLE Mesh to create an efficient mesh network for real-time environmental monitoring.

The primary goal is to design, implement, test and optimize a BLE Mesh network using Nordic Semiconductor nRF52-DK boards, focusing on real-time detection and transmission of environmental signals. The study focuses testing the performance of the BLE Mesh network in various configurations, such as single link performance, chain, mesh, and hybrid topologies in both indoor and outdoor environment, to optimize network performance and ensure reliable data transmission.

The thesis includes a literature review on BLE Mesh networking, environmental monitoring systems, sensor technologies, and communication protocols. The system design phase outlines the network architecture, including node placement, communication protocols and sensor integration. Hardware and software development will involve firmware development for the nRF52-DK boards to enable BLE Mesh communication.

Environmental sensors will measure parameters like luminous intensity, noise levels, temperature or humidity. The optimization phase will focus on enhancing scalability and data reliability through iterative testing. Practical experiments will validate the network's functionality in real-world environments.

By addressing network range and signal accuracy challenges, this research aims to provide a robust framework for deploying BLE Mesh networks in diverse IoT applications, promoting the broader adoption of IoT technologies in environmental monitoring.

1.1 IoT Challenges

For over two decades, the Internet of Things (IoT) has transformed industries by enabling remote monitoring, analysis, and control of devices. However, the proliferation of IoT applications has brought significant challenges for developers, manufacturers, and users. Despite the ease of adding connectivity, each new IoT application faces recurring issues, often unknown to many manufacturers.

IoT devices face several significant challenges. Security is a major concern, as these devices are vulnerable to cyberattacks due to limited power for robust security measures and outdated firmware that accumulates vulnerabilities. Coverage is another issue, as IoT devices need a network connection, but many solutions, like Wi-Fi, have limited range, restricting deployment locations. Scalability also presents challenges, as managing many IoT devices is complex, requiring different connectivity solutions and management platforms across regions, creating logistical challenges.

Interoperability issues arise from diverse IoT configurations and a lack of universal standards, complicating technology integration and deployment. Bandwidth availability is limited, and RF bandwidth can become congested, leading to signal interference and reduced device performance. Additionally, IoT devices often rely on small batteries designed for long life, but high-power consumption tasks, like continuous data transmission, reduce their longevity. Remote access is another challenge, as connectivity types affect remote access, with some requiring on-site presence for troubleshooting or updates, which is costly and impractical at scale.

These challenges are becoming more pronounced as IoT becomes more widespread and accessible, highlighting the need for continuous innovation and adaptation in IoT technologies [1].

1.2 Objectives

The primary objective of this thesis is to design, implement, test, and optimize a Bluetooth Low Energy (BLE) Mesh network performance tailored for real-time environmental monitoring and alerting. This objective is divided into several detailed goals. The first goal is to design the BLE Mesh network by developing topologies for single link, multi-hop, and many-to-many communication. The second goal involves firmware development for nRF52-DK boards, where the nodes will be configured as listeners, relays, and broadcasters. Additionally, communication protocols will be implemented on the nRF52-DK boards to facilitate data exchange and provisioning within the BLE Mesh network.

The third goal focuses on testing the network performance through practical experiments to assess the functionality and feasibility of the BLE Mesh network in the designed topologies. The fourth goal involves evaluating the network performance, including a range evaluation to assess communication over different distances of the BLE Mesh network under varying conditions, and a data accuracy evaluation by calculating the Packet Loss Ratio (PLR) over each node under different traffic loads.

Finally, the fifth goal is to apply the BLE Mesh network in case studies and applications, specifically by demonstrating real-time environmental monitoring applications through the integration of a temperature sensor into the network.

2 BLUETOOTH LOW ENERGY MESH (BLE MESH)

This chapter delves into the intricacies of BLE Mesh communication and its relationship with the Bluetooth Low Energy (BLE). It introduces the BLE Mesh architecture, explaining the key concepts and terminologies associated with this technology. Furthermore, the chapter explores the distinctive features of BLE Mesh networks, highlighting their applications within the Internet of Things (IoT).

2.1 Overview of BLE Mesh

The goal of the July 2017 introduction of the Bluetooth LE Mesh protocol was to improve the capabilities and range of Bluetooth networks, especially for industrial applications that make use of Bluetooth Low Energy (BLE).

Two main topologies were supported by earlier Bluetooth versions:

- One-to-One: Straightforward connections made by two BLE gadgets.
- One-to-Many: Beacons and other devices that run-in broadcast mode.

With Bluetooth LE Mesh, a new many-to-many topology was supported, enabling multiple mesh devices to connect with each other through mesh nodes inside the same network using multi-hop communication. This makes it possible to build expansive networks of devices that can support hundreds or even thousands of units, making them perfect for a wide range of Internet of Things (IoT) uses [2].

BLE Mesh communication occurs over the three designated BLE advertisement channels: 37, 38, and 39. Nodes in the mesh network continuously scan these channels using a 100% duty cycle, meaning they listen for incoming packets unless they are transmitting. Communication relies on a managed flooding mechanism as illustrated in Figure 1, where each node repeats incoming messages, propagating them throughout the network until they reach the intended destination.

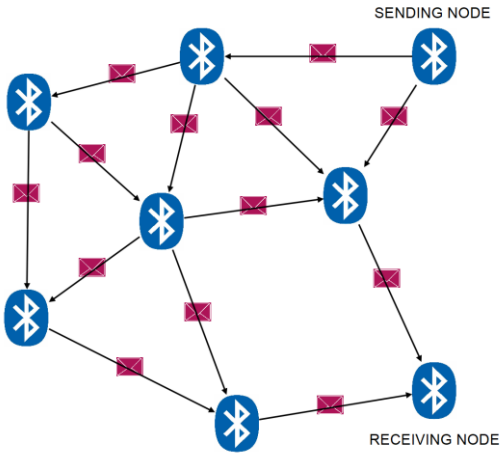


Figure 1. Flooding Mechanism

The total communication time includes the time for transmission across these channels, the channel switching time, and processing at both ends. The receiving node captures the packet, moves the message to the application layer, and may trigger actions. An acknowledgment is then sent back through the same three-channel process, completing the communication cycle when the original sender processes the acknowledgment. The duration of this cycle depends on the channels used for both the initial message and the acknowledgment [3].

The Bluetooth Low Energy (BLE) stack is divided into two main components as illustrated in Figure 2, the host and the controller. The host typically runs on the operating system, while the controller is often a separate chip. These components communicate through the Host Controller Interface (HCI). BLE Mesh technology specifically utilizes the link layer and physical layer of the controller [4].

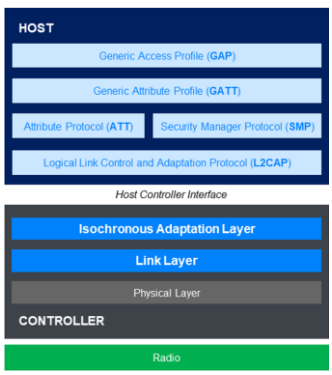


Figure 2. The Bluetooth LE stack

BLE operates in the 2.4 GHz ISM band, spanning 2402 to 2480 MHz, with 40 channels spaced 2 MHz apart as illustrated in Figure 3. These include 3 advertising channels (37,38 and 39) placed specifically in the lower, upper, and middle of the band to avoid interference from Wi-Fi and other sources and 37 data channels (Ch. 0-36) for connection-oriented communication.

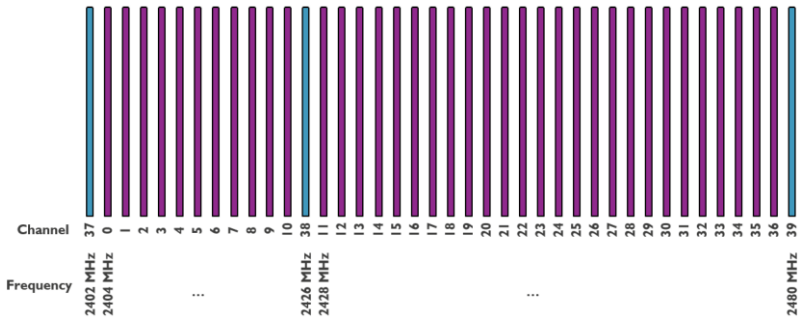


Figure 3. BLE Frequencies range

BLE employs two primary communication modes: Advertising Mode, where devices broadcast information on advertising channels and Connection-oriented Mode, where a master device manages multiple peripherals using TDMA and FHSS [3].

The BLE Physical Layer uses Gaussian Frequency Shift Keying (GFSK) for modulation, encoding digital data by shifting the frequency of the carrier signal—upwards for a binary '1' and downwards for a binary '0'. The frequency deviation, which varies with the PHY variant, defines three key variants. LE 1M PHY operates at 1 Msym/s with a minimum frequency deviation of 185 kHz and is mandatory for all BLE devices. LE 2M PHY, which is optional, doubles the symbol rate to 2 Msym/s with a 370 kHz deviation. LE Coded PHY, also optional, uses Forward Error Correction (FEC) at 1 Msym/s to extend range but reduces the data rate.

BLE operates in half-duplex mode, utilizing Time-Division Duplexing (TDD) to simulate full-duplex communication. Advanced features include Angle of Arrival (AoA) and Angle of Departure (AoD) for direction finding, which employ antenna arrays and switching techniques during signal transmission or reception [4].

2.2 Bluetooth Low Energy Mesh Architecture

The mesh stack consists of several layers as it illustrated in Figure 4, each with specific roles that work together to create a mesh network.

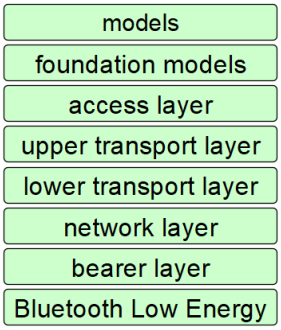


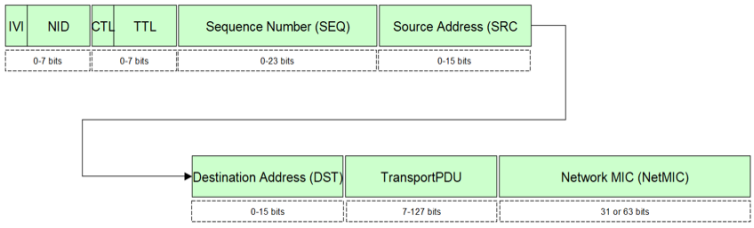
Figure 4. BLE Mesh layers

The Models Layer implements behaviors, messages, states, and state bindings as per model specifications, while the Foundation Models Layer handles models for mesh network configuration and management. The Access Layer enables applications to use the upper transport layer, defining application data format and managing encryption/decryption processes.

The Upper Transport Layer handles encryption, decryption, and authentication of application data, as well as transport control messages, while the Lower Transport Layer manages PDU segmentation and reassembly when necessary [5].

The Network Layer plays a crucial role in transporting Lower Transport PDUs via the bearer layer. It defines the Network PDU format and performs essential functions, including decrypting and authenticating incoming messages, encrypting and authenticating outgoing messages, and managing message relay decisions [6]. Figure 5 shows PDU structure and its descriptions, Understanding the packet frame at this layer is vital, as it provides the foundation for tracing PDU packets in Wireshark, particularly focusing on TTL, SEQ, DST, IVI, and NID, which will give us more information about the traffic in BLE Mesh networks. This will be discussed in Chapter 4.3.

The Bearer Layer manages mesh Protocol Data Units using two types of bearers: the Advertising Bearer, which utilizes Bluetooth LE's GAP advertising and scanning capabilities, and the GATT Bearer, which enables devices that don't support such as (smart phones, tablets, etc) the Advertising Bearer to communicate indirectly with mesh network nodes using the Proxy Protocol. A mesh Proxy node can support both bearer types, allowing for conversion and relay of messages between Advertising and GATT Bearers, thus ensuring a wide range of devices can interact with the mesh network. This layered architecture facilitates efficient and flexible communication within the BLE Mesh network, with each layer handling specific aspects of data processing and transmission [5].



Field Name	Bits	Notes
IVI	1	Least significant bit of IV Index
NID	7	Value derived from the NetKey used to identify the encryption Key and Privacy Key used to secure this PDU
CTL	1	Network Control
TTL	7	Time To Live
SEQ	24	Sequence Number
SRC	16	Source Address
DST	16	Destination Address
TransportPDU	8 to 128	Transport Protocol Data Unit
NetMIC	32 or 64	Message Integrity Check for Network

Figure 5. PDU packet frame with description

2.3 Concepts and Terminology

Grasping the topology of Bluetooth Mesh networking involves learning a series of technical terms and concepts that are unique to it. This section will introduce and explain the most essential of these terms and concepts.

Mesh node is a device that has been added to a Bluetooth mesh network. When devices are not yet integrated into the network are termed unprovisioned devices. There are four types of nodes. Relay nodes retransmit messages from other nodes to extend their reach across the network. Proxy nodes act as intermediaries, facil-

itating communication between non-mesh BLE devices and the mesh network using the GATT protocol. Friend nodes play a crucial role in energy conservation by caching messages for Low Power Nodes (LPNs), allowing these nodes to sleep longer and thus reduce their energy consumption. LPNs are specifically designed for minimal energy use, relying on friend nodes to receive messages and maintain efficient power consumption [5].

A node may consist of multiple elements, each element is independently controllable components. For instance, a light fixture could include several light bulbs, each of which can be turned on or off separately. These individually controllable parts within a single node.

In Bluetooth mesh networks, models are used to define the functionality of nodes. Each model encapsulates a set of states and behaviors specifying the messages required to interact with these states. In Figure 6 shows that element 1 contains 3 models. A key example of these models is the mandatory Configuration Model, which manages a node's configuration through various states and messages. Beyond the standard models specified by Bluetooth, the nRF5 SDK for Mesh includes unique models for specialized functions, and custom models can be created for specific tasks [7], such as broadcasting reboot commands among nodes, as implemented in this project. (Vendor Model4.2.1).

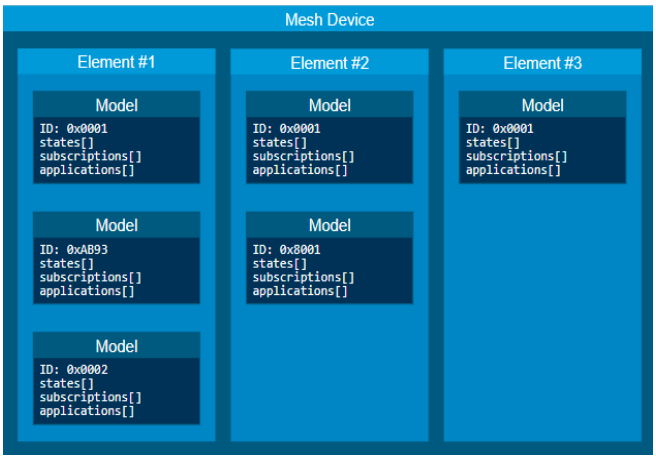


Figure 6. Structure of a Mesh Device with Multiple Elements and Models

Elements within a node can exist in different conditions, indicated by state values., like a light bulb being "on" or "off." Changing from one state to another. Additionally, properties provide context to state values, such as specifying whether a temperature reading is for an outdoor or indoor environment. There are two types of properties, a manufacturer property, which provides read-only access and admin property which provides read-write access.

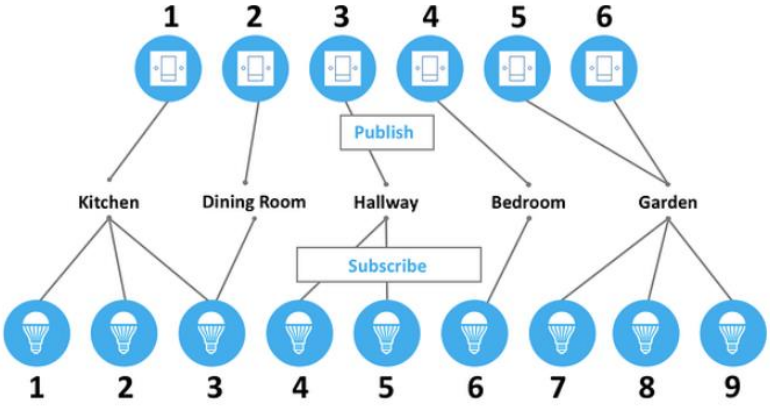
Scenes in a mesh network are collection of states values identified by a unique 16-bit identifier, allowing simultaneous adjustment of multiple states across different nodes, activated on-demand or at a scheduled time.

Communication in mesh network is message-oriented, with nodes sending messages to query, control, or report status. Each message has a unique operation code (opcode) that defines its type. Messages are categorized as either acknowledged or unacknowledged. Acknowledged messages require a response to confirm receipt and provide data, and they must be idempotent, meaning their effect remains the same even if received multiple times. If a response is not received, the sender may resend the message [5].

Operation codes, or opcodes, are defined by the Bluetooth Special Interest Group (Bluetooth SIG). They are specific identifiers within Bluetooth Mesh networks. These opcodes differentiate the types of messages that can be sent and received by models within the mesh network. Each opcode corresponds to a specific operation, ensuring that messages are accurately interpreted by receiving nodes. In Bluetooth communication, an opcode is a sequence of bytes (octets) that instructs a device on what action to perform, with opcodes consisting of 1, 2, or 3 octets depending on the structure of the first octet [6].

Bluetooth mesh uses three types of addresses. a unicast (unique to each device), group (addressing multiple devices simultaneously), and virtual (UUID-based with a large address space). Devices are assigned unicast addresses during provisioning,

group addresses during network configuration, and virtual addresses can be generated on-the-fly for dynamic addressing needs. In Figure 7, "Switch 1" publishes to the group address "Kitchen." The nodes "Light 1," "Light 2," and "Light 3" are subscribed to the Kitchen address, so they receive, and process messages sent to this address. Consequently, "Light 1," "Light 2," and "Light 3" can be turned on or off using "Switch 1." [2].



Publish-subscribe in Bluetooth mesh lighting control system (Source: "Bluetooth Mesh – An Introduction for Developers")

Figure 7. Publish & subscribe to group addresses

The provisioning process is crucial in Bluetooth Mesh networks, enabling devices to join and become nodes within the network. the device responsible for adding nodes is called the provisioner, typically a tablet or smartphone. Provisioning consists multiple steps, in the first step called beaconing, the unprovisioned device signals its availability by broadcasting mesh beacon advertisements, which include information like UUID and OOB data as illustrated in Figure 8, devices 1,2 and 3 are sending their beacons to the provisioner. The second step known as Invitation. Upon detecting these beacons, the provisioner sends a provisioning invite to the selected device. The unprovisioned device responds with its capabilities, such as the number of supported elements and security algorithms. In Figure 8, provisioner chooses to provision with device #2).

The third step is exchanging public keys, the provisioner and device exchange public keys using the Elliptic-curve Diffie-Hellman (ECDH) algorithm, either directly or through an OOB channel. In the fourth step, the Authentication starts verifying the device, potentially requiring user interaction. Authentication methods may include displaying or entering random numbers or using static OOB methods.

The last step is provisioning data distribution, a session key is generated to securely exchange provisioning data, Including the network key (NetKey), device key (DevKey), initialization vector index (IV index), and a unicast address. The device then becomes an active node in the network [2].

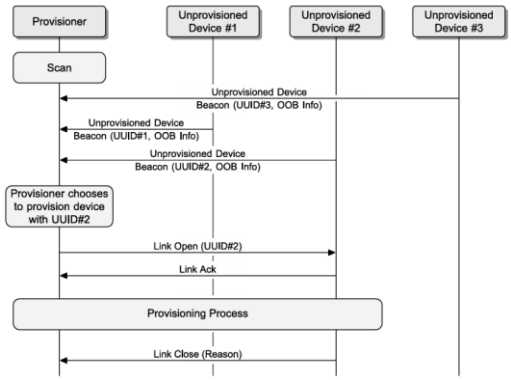


Figure 8. Establishment of Link by ID between a Provisioner and unprovisioned device

Bluetooth LE mesh networks use a method called “managed flooding,” this method ensuring messages reach their destinations via multiple paths. There are measures to optimize the way of flooding works in BLE Mesh network. Heartbeats measures, they are periodic messages sent by nodes to indicate activity and provide data on the number of hops needed to reach them, which helps optimize the Time To Live (TTL) field. The TTL measure controls the maximum number of hops for message relaying, conserving energy by preventing unnecessary relays, and heartbeat data helps determine the optimal TTL for each message. Additionally, a message cache measure, to store recently seen messages, discarding duplicates immediately to prevent redundancy. Friendship measure is another key feature, were Friend nodes store and forward messages for Low Power nodes [5].

Bluetooth Mesh uses strong security, including provisioning authentication, dual-layer AES-CCM encryption, and replay protection. Network encryption Integrates a device into the mesh network by assigning a network key (NetKey), allowing it to decrypt and authenticate network-layer messages and relay them, but not decrypt application-level data. While transport encryption protects application data with AppKey that shared among specific nodes within the same application in a mesh network, this key is used to decrypt and authenticate application-level messages, but it only works within a single network. The DevKey is a unique key for each device, securing communication during provisioning [7].

BLE Mesh network is protected against security threats. One such threat is the Trash Can Attack, which Involves unauthorized access via discarded devices with stored keys. To prevent this, BLE Mesh securely removes nodes by blacklisting them and refreshing network and application keys, ensuring updated keys are distributed to all remaining nodes. The other threat is the Replay Attack**,** which involves a malicious device retransmitting stored messages. BLE Mesh prevents this by using a sequence number (SEQ) and an incrementing initialization vector index (IV index). If a message's SEQ is less than or equal to the last valid one, it's discarded to prevent replay attacks. The IV index further enhances protection by incrementing with each message [2].

2.4 BLE Mesh Network Data Flow

The data flow in a BLE Mesh network involves multiple layers within the stack structure, as illustrated in Figure 9. When a message is sent from the source node, the application initiates the process by calling the appropriate model's publish function. The model includes the message content in a publishing packet with an operation code (opcode), which is then passed to the access layer. The access layer retrieves essential parameters such as the destination address, encryption keys, and the time to live (TTL) value, before forwarding the packet to the transport layer. The transport layer encrypts the message using the designated application key, segments the message if necessary, and sends each segment to the network

layer. The network layer adds a network header with a sequence number, encrypts the packet with the network key, and transfers it to the bearer layer. The bearer layer includes the network message in an advertisement packet and schedules it for broadcast. All mesh nodes within range receive the broadcast, and the packet is passed from their bearer layers to their network layers. If a node is not the intended destination, the network layer decreases the TTL value by one, re-encrypts the packet with the same network key, and relays it. Once the packet reaches the destination node, the network layer decrypts it and sends it to the transport layer, where all message segments are reassembled, decrypted with the application key, and passed to the access layer. The access layer verifies the opcode, application key and destination address, and delivers the message to the relevant models. The receiving model processes the message and may instruct the application to execute the corresponding action. Additionally, the receiving model can send an acknowledgment back to the source node, confirming receipt of the message, following the same data flow in reverse [8].

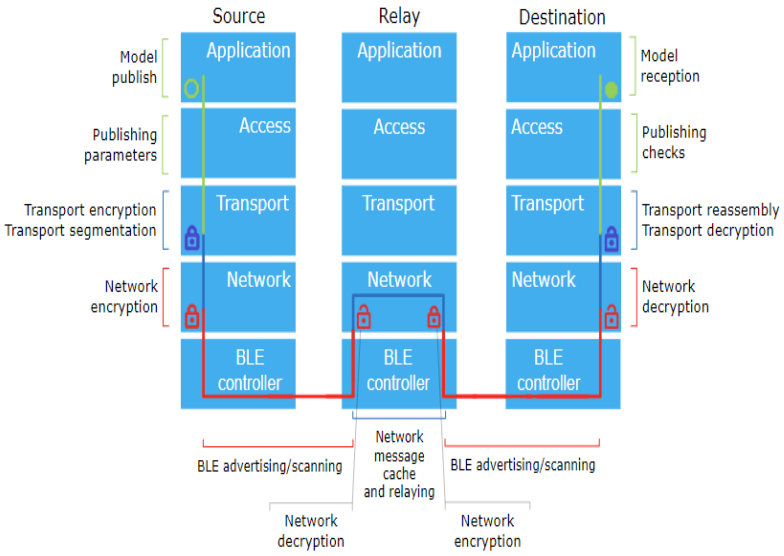


Figure 9. Data flow in BLE Mesh

2.5 Features of BLE Mesh Networks

Bluetooth LE Mesh offers many benefits in IoT applications. It is built on top of BLE, using the same protocol stack, which means that smart devices with Bluetooth 4.0 or later, such as many modern smartphones, can directly utilize Bluetooth mesh technology. Additionally, Bluetooth mesh ensures global interoperability, guaranteeing that devices and products from different manufacturers will work together seamlessly as long as they are certified according to the standard. Scalability is another advantage, with Bluetooth mesh designed to support a large number of nodes within a single network. It can accommodate up to 32,767 elements. Reliability is also one of the features that focuses on the development of the standard, ensuring it can effectively handle lost or corrupt messages. Finally, security is a mandatory feature in BLE Mesh, unlike BLE, where security is optional and left to the developer's discretion [2].

2.6 Applications of BLE Mesh Networks

Bluetooth Low Energy (BLE) Mesh networks have become a cornerstone in the development of modern, interconnected systems. Their ability to support scalable and reliable communication across numerous devices makes them ideal for a wide range of applications.

Wireless smart lighting is predicted to be a disruptive technology in homes and commercial spaces. Systems range from simple ON/OFF control to complex setups with brightness, color, and temperature control.

BLE Mesh has been used for data transmission from sensors. It is suitable for low data rate, sporadic, or event-driven monitoring rather than continuous high-frequency data transmission. Example BLE Mesh functional systems used in office areas, with hybrid nodes for specific alerts like water leaks.

BLE Mesh can be used for emergency communication such as earthquakes and it is known as “Bluemergency” for two reasons. Firstly, the BLE Mesh network supports many-to-many communications. Secondly, most BLE Mesh devices can be configured to operate on battery power.

BLE Mesh can be used in smart factories. Proposed systems included a mesh sensor network for data collection and secure downstream communication, implemented using Raspberry Pis and nRF52840 DK with ARM Cortex-M4F CPUs.

BLE Mesh improves parking IoT systems by gathering RSSI values for vehicle localization, achieving around 70% accuracy in occupancy detection using machine learning [12]. BLE Mesh enhances smart home functionality by enabling seamless control of various devices, improving convenience, security, and energy management [13].

3 SOLUTION OVERVIEW

This chapter provides an overview of the nRF52 Development Kit series as a solution for IoT application, focusing on both the hardware and software aspects that make it a powerful platform for Bluetooth Low Energy (BLE) mesh networking applications. The chapter begins with an examination of the nRF52832 System on Chip (SoC). This hardware overview is followed by an exploration of the nRF5 Software Development Kit (SDK) for Mesh, which offers a robust set of tools and libraries designed to simplify the development of mesh-capable applications on the nRF52 platform. Additionally, an overview of nRF52840 USB dongle is provided.

3.1 nRF52 Hardware:

The nRF52832 is a versatile SoC by Nordic Semiconductor, featuring a 32-bit Arm Cortex-M4 processor, 512 kB flash memory, and 64 kB RAM. It supports BLE with data rates of 1 Mbps and 2 Mbps in the 2.4 GHz band. The key features include flexible power management, multiple communication interfaces (SPI, I2C, UART), NFC capabilities, and strong security measures.

Available in QFN48 and WLCSP packages, it provides 32 GPIO pins configurable for digital I/O, analog inputs, and peripherals like SPI, I2C, UART, and PWM. GPIOs support internal pull-up/down resistors and can trigger interrupts. The nRF52832 operates on 1.7 V to 3.6 V supply voltage and offers ultra-low power modes, including System OFF (0.3 µA) and System ON (1.9 µA). It has an automatic LDO and DC/DC regulator and a 64 MHz internal oscillator for fast wake-up.

Development kits include 4 physical buttons as illustrated in Figure 10 connected to GPIO pins, with SDK routines for debouncing, capable of waking the device, initiating functions, or triggering interrupts. The nRF52832's efficient power management and robust GPIO capabilities make it suitable for various applications, from simple input devices to complex IoT systems [10].

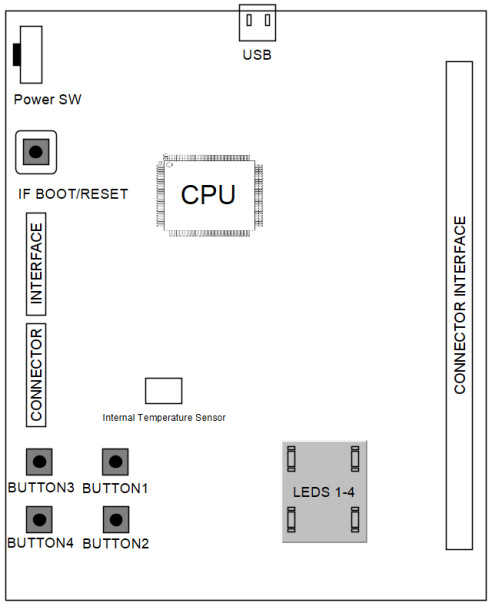


Figure 10. nRF52DK board

3.2 nRF52 Software Development Kit for Mesh

Nordic Semiconductor offers a comprehensive development ecosystem for the nRF52 series, including the nRF5 SDK for Mesh, which provides frameworks, APIs, libraries, and development tools like Segger Embedded Studio. This SDK supports BLE Mesh implementation and integrates seamlessly with nRF52 hardware known for its performance and power efficiency.

The nRF5 SDK includes precompiled binaries and source code to facilitate mesh-capable application development, handling complex communications and state management while optimizing for low power consumption. The provided APIs and libraries simplify managing a BLE Mesh network, allowing developers to focus on higher-level application logic by handling tasks such as message encryption, decryption, and routing within the mesh.

Nordic Semiconductor's development tools, including Segger Embedded Studio, offer an efficient environment for coding, compiling, and debugging, tailored to embedded system development. Additionally, the SDK offers a rich set of example applications for IoT developers working with nRF5 Series devices, demonstrating

functionalities such as Bluetooth Low Energy (BLE) profiles, cryptographic features for secure communications, device firmware updates (DFU), Near Field Communication (NFC), and proprietary protocols like Gazell and Enhanced ShockBurst (ESB) [9].

3.3 nRF52840 USB dongle

The nRF52840 Dongle (PCA10059) as illustrated in Figure 11 is a versatile and cost-effective USB development tool designed for wireless applications using the nRF52840 SoC, including Bluetooth® Low Energy, ANT™, 802.15.4, and custom 2.4 GHz protocols. One of the key uses of this dongle is during the testing and evaluation phase of BLE Mesh networks, where it can be used as a packet sniffer in conjunction with Wireshark. This dongle will capture and analyse BLE Mesh traffic, providing detailed insights into network behavior and performance. The dongle's ability to support all short-range wireless standards available on the nRF52 family, combined with its built-in USB interface for high data throughput, makes it an ideal tool for developing and testing nRF-based wireless solutions [11].

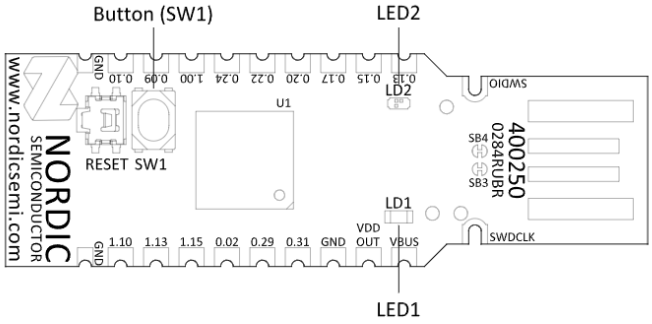


Figure 11. nRF52840 Dongle

4 SOLUTION IMPLEMENTATION

This section will detail the configuration and implementation of the BLE Mesh network, emphasizing the setup of nodes, development of firmware, testing and integration of environmental sensors. It will include practical code samples, hardware setup descriptions, and specialized techniques used to meet the network's requirements. The network will feature five nRF52 nodes positioned at different distances, with each node assigned a specific role—broadcaster, listener, or relay—based on the testing scenarios. Additionally, an nRF52840 USB dongle will be employed to monitor and analyse the signal strength, packet frames, and quality.

4.1 System Installations

This section outlines procedures required for installing and configuring the essential software tools to support the development and analysis tasks in this project. The focus is on setting up the nRF Connect for Visual Studio Code, installing the nRF Sniffer, and configuring additional software such as nRF Connect for Desktop and nRF Mesh for mobile devices.

4.1.1 nRF Connect for VS Code

The nRF Connect for Visual Studio Code provides a comprehensive environment for development, allowing integration with the Nordic Semiconductor tools and SDKs. The steps for installation are as follows [14]:

- **nRF-Command-line-Tools-10.23:** These tools provide command-line interfaces for various nRF devices and functionalities.
- **install j-link driver:** The J-Link driver is essential for interfacing with J-Link debug probes.
- **install VS code extension – nRF Connect for VS Code:** Install this extension to integrate nRF tools with the development environment.

- **install toolchain:** The nRF Connect extension provides guidance for installing the necessary toolchain, which includes the GNU Arm Embedded Toolchain and other dependencies required for building and debugging applications.
- **install nRF connect SDK**: Using the nRF Connect extension, download and install the nRF Connect Software Development Kit (SDK). The SDK offers essential libraries and tools for developing applications on nRF devices.

4.1.2 nRF Mesh Application

The nRF Mesh app is available on IOS and Android. It is a network controller. It allows to provision, control, configure and managed nodes in the network using nRF Mesh open-source libraries. It allows users to authenticate devices, manage nodes, handle network and application keys, control OnOff, Level, and Scenes functions, and send messages to vendor-specific models. The app also supports sharing network configurations across multiple phones for collaborative management. More features are available from Nordic Semiconductor [15].

4.2 Firmware Development:

nRF connect SDK and Zephyr SDK providing many samples of applications. Those applications can be found from reference list [16 & 17]. A Bluetooth Mesh template will be used for developing a firmware for Bluetooth mesh networks involves creating and configuring various types of nodes, each serving distinct roles within the network. These nodes enable communication, data collection, and relay of information across the network. The key types of nodes discussed in this section include the Broadcaster Node, Relay Node, Listener Node, and later Collector Node. Each of these nodes has unique functionalities and is essential for the overall operation and efficiency of the mesh network. Figure 12 shows the implemented nodes in the network.

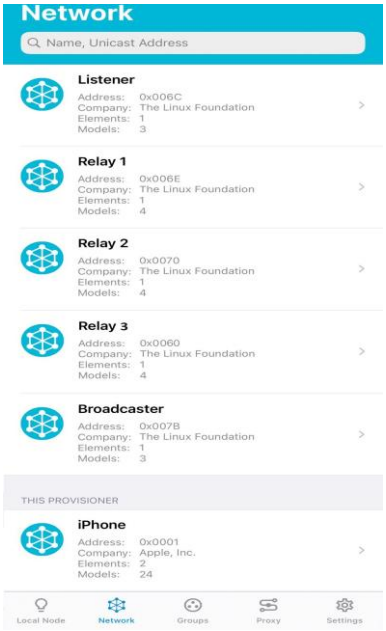


Figure 12. Nodes Creation

Before introducing nodes roles, binding application key, define operation codes (Opcodes) and provisioning need to be configured for all nodes. For Binding Application Key, Nodes that share the same Application Key can decrypt and process the data sent by a broadcaster using that key. (App Key 1) will be assigned to all models in the nodes within this network as it illustrated in Figure 15.

Define operation codes (Opcodes), in Figure 13, the opcode OP_ONOFF_SET_UNACK is used to send an "On/Off Set" command without expecting an acknowledgment, reducing network traffic. Similarly, MY_VENDOR_MODEL_ID is a custom vendor models and will be used to perform specific actions that not implemented by BLE Mesh standards like rebooting a device. When a node receives a message with this opcode and model ID, it knows to execute the corresponding command.

```
/* Define Bluetooth Mesh operation codes (opcodes)*/
#define OP_ONOFF_GET        BT_MESH_MODEL_OP_2(0x82, 0x01)
#define OP_ONOFF_SET        BT_MESH_MODEL_OP_2(0x82, 0x02)
#define OP_ONOFF_SET_UNACK BT_MESH_MODEL_OP_2(0x82, 0x03)
#define OP_ONOFF_STATUS     BT_MESH_MODEL_OP_2(0x82, 0x04)
#define OP_SEQ_NUMBER       BT_MESH_MODEL_OP_2(0x82, 0x05)
#define OP_REBOOT BT_MESH_MODEL_OP_2(0x82, 0xFF)
#define MY_VENDOR_MODEL_ID 0x8001
```

Figure 13. Opcodes Identifier

For provisioning, all nodes in the network follow the same provisioning process, managed by functions and a provisioning structure provided by the Zephyr Bluetooth Mesh Provisioning API (Figure 14). This process defines how each node interacts with the provisioner. The output_number function displays an Out-Of-Band (OOB) number during provisioning, which may be used for user authentication. Once provisioning is complete, the prov_complete callback is triggered, indicating the node's full integration into the mesh network, often by actions like lighting an LED. If the node's provisioning data is reset, the prov_reset function re-enables provisioning. The bt_mesh_prov structure encapsulates these behaviors, including the node's unique identifier (UUID), OOB output size, and the relevant callback functions. This structure ensures integration of the node into the Bluetooth Mesh network, allowing it to communicate with other nodes using the assigned credentials.

```c
/* Provisioning */

static int output_number(bt_mesh_output_action_t action, uint32_t number)
{
    printk("OOB Number: %u\n", number);

    board_output_number(action, number);

    return 0;
}

static void prov_complete(uint16_t net_idx, uint16_t addr)
{
    board_prov_complete();
}

static void prov_reset(void)
{
    bt_mesh_prov_enable(BT_MESH_PROV_ADV | BT_MESH_PROV_GATT);
}

static const struct bt_mesh_prov prov =
{
    .uuid = dev_uuid,
    .output_size = 4,
    .output_actions = BT_MESH_DISPLAY_NUMBER,
    .output_number = output_number,
    .complete = prov_complete,
    .reset = prov_reset,
};
```

Figure 14. Provisioning configurations

4.2.1 Broadcaster Node

Assigning the role of a Broadcaster to a node in a BLE Mesh network involves a series of configurations that enable the node to transmit messages to other nodes across the network. This process begins with the assignment of specific models, context configuration and message transmission. One element and three models depicted in Figure 15 will be created to do different actions.

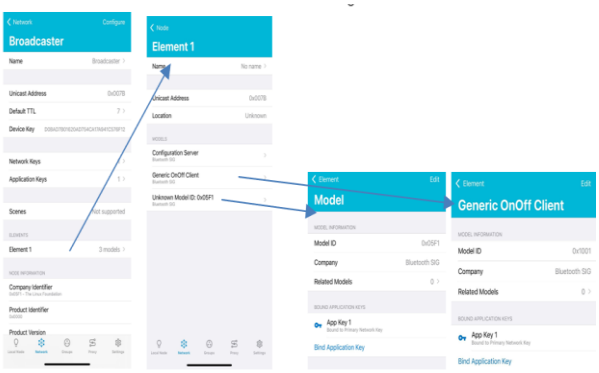


Figure 15. Models Creation in Broadcaster node

Models in the broadcaster node are (Figure 16):

- **Configuration server (BT_MESH_MODEL_CFG_SRV):** The Configuration Server model is mandatory for all nodes, and it plays a crucial role in ensuring that the node can be properly managed and controlled within the mesh network.
- **Generic On/Off Client Model (BT_MESH_MODEL_ID_GEN_ONOFF_CLI):** Allows the node to send On/Off commands, including sequence numbers, to test network performance.
- **Vendor Model (BT_MESH_MODEL_VND):** Allow the node to send reboot commands across the network.

```
/* This application only needs one element to contain its models */
    struct bt_mesh_model models[] = {
    BT_MESH_MODEL_CFG_SRV,
    BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_ONOFF_CLI, NULL, NULL,
                  NULL),
    BT_MESH_MODEL_VND(BT_COMP_ID_LF, MY_VENDOR_MODEL_ID, NULL, NULL, NULL)

};

static const struct bt_mesh_elem elements[] = {
    BT_MESH_ELEM(0, models, BT_MESH_MODEL_NONE),
};
```

Figure 16. Elements and Models Creation in broadcaster

Context Configurations in the broadcaster are made by bt_mesh_msg_ctx structure and it is used to define the context for sending a Bluetooth Mesh message. It specifies several parameters that control how the message is transmitted across the network. In Figure 17 the app_idx field is set to the first application key (keys[0]) associated with the model at index models[1] which is representing the

Generic On/Off Client (BT_MESH_MODEL_ID_GEN_ONOFF_CLI), ensuring that the message is encrypted using the first key in this array. The addr field is assigned the value BT_MESH_ADDR_ALL_NODES, which indicates that the message will be broadcast to all nodes in the mesh network. The Addr field can be changed to transmit a message to a specific destination by using the node unicast address such as (0x006C). Finally, the send_ttl field is set to BT_MESH_TTL_DEFAULT, which determines the maximum number of hops the message can make through the network, typically set to 127, allowing the message to traverse a large network before being discarded. This value can also change by user input.

After models are created and context configuration are made, a buffer need to be created before message transmission starts. 4 bytes of buffer is created for this message opcode (OP_ONOFF_SET_UNACK). The state value of the led (val), sequence number (seq) and transition ID (tid) are added to this buffer. Then bt_mesh_model_send() is ready to transmit the packet within the Bluetooth Mesh network. This function handles the transmission using the defined parameters (model index, context configuration, buffer, Optional callback and callback data). Then the client model and the configured context (ctx) and 4 bytes of buffer with NULL of call back data will be passed and transmission begins. A custom Vendor Model, using model [2] (BT_MESH_MODEL_VND), follows a similar process for message transmission.

```c
/** Send an OnOff Set message from the Generic OnOff Client to all nodes. */
static int gen_onoff_send_with_seq(bool val)
{
    if (models[1].keys[0] == BT_MESH_KEY_UNUSED) {
        printk("The Generic OnOff Client must be bound to a key before sending.\n");
        return -ENOENT;
    }
    static uint16_t seq_num = 1;
    struct bt_mesh_msg_ctx ctx = {
        .app_idx = models[1].keys[0],
        .addr = BT_MESH_ADDR_ALL_NODES,
        .send_ttl = BT_MESH_TTL_DEFAULT,
    };
    static uint8_t tid =0;
    printk("tid: %d\n", tid);
    printk("src: %d\n", onoff.src);
    printk("Sending OnOff: %d, Seq Num: %u\n", val,seq_num);
    printk("..........................\n");

    BT_MESH_MODEL_BUF_DEFINE(buf, OP_ONOFF_SET_UNACK, 4);
    bt_mesh_model_msg_init(&buf, OP_ONOFF_SET_UNACK);
    net_buf_simple_add_u8(&buf, val);
    net_buf_simple_add_le16(&buf, seq_num++);  // Use a custom sequence number
    net_buf_simple_add_u8(&buf, tid++);

    return bt_mesh_model_send(&models[1], &ctx, &buf, NULL, NULL);
}
```

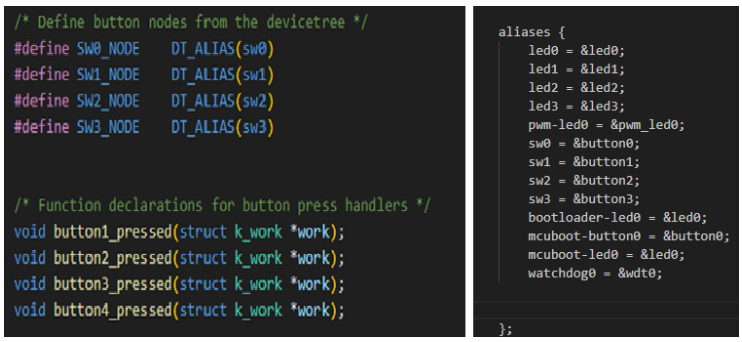
Figure 17. Message transmission

Buttons configurations are configured for use in the testing phase, where each button triggers a different data rate (number of messages per minute) to be sent over BLE Mesh. As shown in Figure 18, macros are used to define four button nodes (SW0_NODE, SW1_NODE, SW2_NODE, SW3_NODE), which are linked to corresponding aliases (sw0, sw1, sw2, sw3) in the device tree which is a hierarchical data structure that describes hardware in the system [18]. This ensures that each button is correctly mapped to its hardware configuration.

Four handler functions (button1_pressed,…., button4_pressed) are declared, with each function associated with a specific button:

- Button 1: Configured to send data at a rate of 60 and 90 packets per minute.
- Button 2: Configured to send data at a rate of 120 packets per minute.
- Button 3: Configured to send data at a rate of 600 packets per minute.
- Button 4: Configured to send a reboot command to all nodes, allowing remote control of all nodes during the testing phase to collect data simultaneously without manually reading each test.

This configuration facilitates testing by enabling different data rates and remote reboot control via the buttons, as illustrated in Figure 18.



```
/* Define button nodes from the devicetree */
#define SW0_NODE    DT_ALIAS(sw0)
#define SW1_NODE    DT_ALIAS(sw1)
#define SW2_NODE    DT_ALIAS(sw2)
#define SW3_NODE    DT_ALIAS(sw3)


/* Function declarations for button press handlers */
void button1_pressed(struct k_work *work);
void button2_pressed(struct k_work *work);
void button3_pressed(struct k_work *work);
void button4_pressed(struct k_work *work);
```

```
aliases {
    led0 = &led0;
    led1 = &led1;
    led2 = &led2;
    led3 = &led3;
    pwm-led0 = &pwm_led0;
    sw0 = &button0;
    sw1 = &button1;
    sw2 = &button2;
    sw3 = &button3;
    bootloader-led0 = &led0;
    mcuboot-button0 = &button0;
    mcuboot-led0 = &led0;
    watchdog0 = &wdt0;

};
```

Figure 18. Buttons configurations

4.2.2 Listener Node

The listener node illustrated in Figure 19 is implemented by defining specific Bluetooth Mesh models that enable it to receive and process messages. This is done

using the Generic On/Off Server model, along with a custom model to handle additional functionality. One element and 4 models are created in this node.

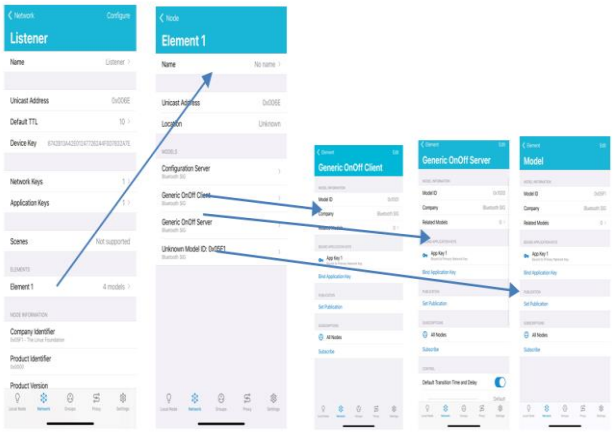


Figure 19. Listener Node

The created models (Figure 20) in Listener Node are:

- **Configuration server (BT_MESH_MODEL_CFG_SRV):** The Configuration Server model is mandatory for all Bluetooth Mesh nodes, and it plays a crucial role in ensuring that the node can be properly managed and controlled within the mesh network.
- **BT_MESH_MODEL_ID_GEN_ONOFF_CLI:** This model will be used later to send temperature data to collector node (This is not part of testing phase).
- **BT_MESH_MODEL_ID_GEN_ONOFF_SRV:** This is the OnOff Server model. It listens to gen_onoff_set_unack messages coming from the broadcaster.
- **BT_MESH_MODEL_VND:** This model used for receiving rebooting command from the broadcaster.

```
// Mesh Model Declaration
static struct bt_mesh_model models[] = {
    BT_MESH_MODEL_CFG_SRV,
    BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_ONOFF_CLI, gen_onoff_cli_op, NULL, NULL),
    BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_ONOFF_SRV, gen_onoff_srv_op, NULL, NULL),
    BT_MESH_MODEL_VND(BT_COMP_ID_LF, MY_VENDOR_MODEL_ID, reboot_model_ops, NULL, NULL)
};

// Mesh Element Declaration
static struct bt_mesh_elem elements[] = {
    BT_MESH_ELEM(0, models, BT_MESH_MODEL_NONE),

};
```

Figure 20. Elements and model creation in listener

After creating the models, the Listener node receives incoming messages through the Generic On/Off Server operation model. The broadcaster transmits messages using the OP_ONOFF_SET_UNACK opcode. The Listener node is configured to listen for this opcode to receive messages, as illustrated in Figure 21. Upon receiving a message, the gen_onoff_set_unack() function handles the reception of On/Off commands, extracts relevant data such as sequence numbers (seq) and (val) value to triggers the LED to blink accordingly.

```
static const struct bt_mesh_model_op gen_onoff_srv_op[] = {
    { OP_ONOFF_SET_UNACK, BT_MESH_LEN_MIN(2),   gen_onoff_set_unack },
    BT_MESH_MODEL_OP_END,
};
```

Figure 21. Receiving Messages using OP_ONOFF_SET_UNACK

After extracting the sequence number from the packet, it is passed to the PLR function to calculate the Packet Loss Ratio (PLR). The PLR function processes the sequence numbers to compute packet loss statistics, including the total number of missed packets and the packet loss ratio. Additionally, it manages out-of-order and duplicate packets as shown in Figure 22.

```
// Handler for receiving messages with sequence numbers
static int PLR(uint16_t seq_num) {
    // Static variables to keep track of sequence and statistics
    static uint16_t last_seq_num = 0;
    static uint16_t first_seq_seq = 0;
    static bool is_first_packet = true;
    static uint16_t total_missed_packets = 0;
    static uint16_t total_received_packets = 0;

    // Handling the first packet received
    if (is_first_packet) {
        first_seq_seq = seq_num;
        last_seq_num = seq_num ;  // Initialize the last sequence number with the first received sequence
        is_first_packet = false; // Mark that the first packet has now been received
        total_received_packets++; // Start counting packets with the first one received
    } else {
        // Normal handling for all subsequent packets
        if (seq_num > last_seq_num) {
            // Calculating missed packets if there's a gap
            total_missed_packets += (seq_num - last_seq_num - 1);
            last_seq_num = seq_num; // Update the last received sequence number
            total_received_packets++; // Increment the total received packets count
        } else if (seq_num == last_seq_num) {
            // Handle duplicate packet
            printf("Duplicate packet received, seq_num: %u\n", seq_num);
        } else {
            // Handle out of order packet
            printf("Out of order packet received, seq_num: %u, last_seq_num: %u\n", seq_num, last_seq_num);
        }
    }

    // Calculate Packet Loss Ratio (PLR)
    uint32_t total_packets_considered = total_received_packets + total_missed_packets;
    uint32_t plr = (total_missed_packets * 10000) / total_packets_considered;

    printk("First received packet: %u\n", first_seq_seq);
    printk("Total missed packets: %u\n", total_missed_packets);
    printk("Total received packets: %u\n", total_received_packets);
    printk("Received seq_num %u, PLR: %u.%02u%%\n", seq_num, plr / 100, plr % 100);
    printk("......................\n");

    return 0;
}
```

Figure 22. PLR Calculation

4.2.3 Relay Node

In Zephyr, the relay functionality within the Bluetooth Mesh subsystem is controlled through specific Kconfig options that enable and fine-tune the relay behaviour. In Figure 23 The **CONFIG_BT_MESH_RELAY** option enables a node to relay messages received from other nodes across the mesh network, which is crucial for creating a robust network where messages need to traverse multiple hops. The **CONFIG_BT_MESH_RELAY_RETRANSMIT** option further refines this functionality by defining the retransmission count and interval for relayed messages, ensuring multiple opportunities for message delivery, particularly in environments prone to interference or packet loss. This is complemented by the **CONFIG_BT_MESH_RELAY_RETRANSMIT_COUNT** sub-option, which sets the exact number of retransmissions, and the **CONFIG_BT_MESH_RELAY_RETRANSMIT_INTERVAL** sub-option, which controls the time interval between each retransmission. These configurations are essential for optimizing network performance, balancing reliability, and managing network traffic effectively [20].

```
CONFIG_BT_MESH_LOOPBACK_BUFS=3
CONFIG_BT_MESH_NETWORK_TRANSMIT_COUNT=2
CONFIG_BT_MESH_NETWORK_TRANSMIT_INTERVAL=20
CONFIG_BT_MESH_RELAY=y
CONFIG_BT_MESH_RELAY_ENABLED=y
CONFIG_BT_MESH_RELAY_RETRANSMIT_INTERVAL=20
CONFIG_BT_MESH_TX_SEG_MSG_COUNT=1
CONFIG_BT_MESH_RX_SEG_MSG_COUNT=1
CONFIG_BT_MESH_SEG_BUFS=64
CONFIG_BT_MESH_RX_SEG_MAX=3
CONFIG_BT_MESH_TX_SEG_MAX=3
CONFIG_BT_MESH_SAR_TX_SEG_INT_STEP=0x05
```

Figure 23. Enable Relay

4.3 Testing and Evaluation

With the BLE Mesh network design established, the next phase involved rigorous testing and evaluation to assess its performance under various conditions. In this section, the system will log every packet received and lost to evaluate traffic efficiency by calculating the packet loss ratio at each node and comparing the results

across different nodes and scenarios. The testing phase will include four distinct scenarios.

4.3.1 Scenario 1 single link performance

Scenario 1 is focus on a single link in a BLE Mesh network. Figure 24 depict a series of tests measuring the Packet Loss Ratio (PLR) at various distances (5m, 15m, 30m, and 40m) between a broadcaster (Br) and observers (R1, R2, R3), with the support of a sniffer device. These tests were conducted in an indoor environment with obstructing obstacles.

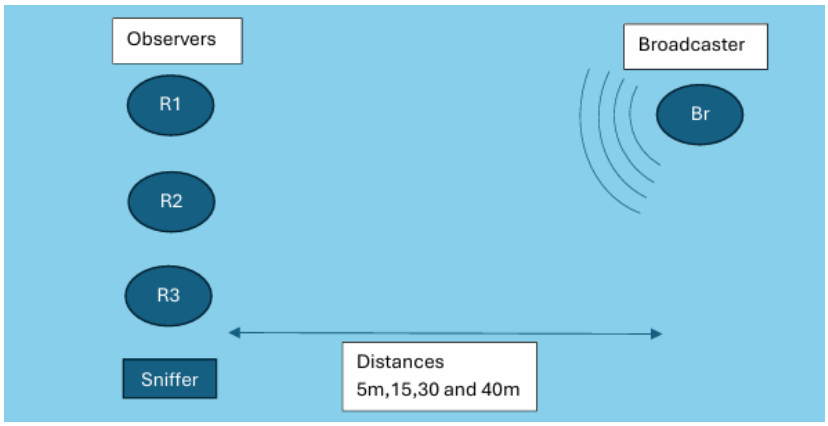


Figure 24. single link performance

The results presented in Table 1. The data reveals a clear relationship between distance, packet generation rates (data rate), and RSSI (Received Signal Strength Indication). As both link distance and packet generation rate increase, the Packet Loss Ratio (PLR) also rises. At the shortest distance tested (5 meters), the Packet Loss Ratio (PLR) remains low across most data rates. However, at 1200pkt/min, there is a noticeable increase in PLR values, likely due to the increased load on network traffic.

At 15 meters, while some packet loss is observed, especially at higher data rates, the signal remains strong enough (-65 dBm to -66 dBm) to maintain communication with manageable Packet Loss Ratios (PLR). However, beyond this distance, the

signal deteriorates significantly, as shown by the results at longer distances where no signal is detected at the lower data rates and extremely high PLR values (up to 91%) are recorded. This suggests that the network coverage reaches it edge by 30 meters.

Distance	Data-Rate	PLR /R1	PLR /R2	PLR /R3	RSSI
5m	60	0.00%	0.00%	0.00%	~-51 dBm
	120	0.00%	0.00%	0.00%	~-51 dBm
	600	0.52%	0.34%	0.17%	~-52 dBm
	1200	2.86%	2.95%	2.95%	~-50 dBm
15m	60	0.00%	6.90%	1.44%	~-65 dBm
	120	2.65%	6.03%	0.00%	~-66 dBm
	600	5.42%	7.35%	3.36%	~-66 dBm
	1200	12.04%	36.58%	21.24%	~-66 dBm
30m	60	no signal	no signal	89.06%	~-80 dBm
	120	no signal	no signal	87.91%	~-80 dBm
	600	no signal	no signal	90%	~-82 dBm
	1200	no signal	no signal	91%	~-86 dBm
40m	60	no signal	no signal	no signal	---
	120	no signal	no signal	no signal	---
	600	no signal	no signal	no signal	---
	1200	no signal	no signal	no signal	---

Table 1. single link performance test

4.3.2 Scenario 2 (Chain Topology)

After determining that nodes can directly transmit over distances of 15-30 meters in a closed environment where signal tracing is difficult, the second scenario of testing took place in an open environment free of obstacles to enhance signals. In Scenario 2 as illustrated in Figure 25, the nodes were arranged in a chain topology with one broadcaster, three relays, and one listener, positioned at distances of 15-meters and 30-meters apart.

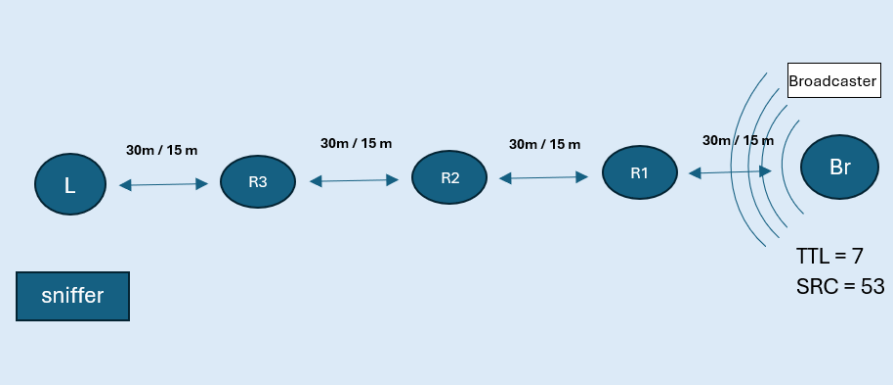


Figure 25. Scenario 2

In this scenario the broadcaster-initiated a different data rate transmission cycle over a 60-second period. Relay nodes forwarded these messages to adjacent nodes, decrementing the Time To Live (TTL) from 7 to 4 by the time they reached the listener(L) as illustrated in Figure 26. Therefore a 3-hops end-to-end network is formed. A sniffer monitored traffic from R3 to show message relaying by analyzing TTL and source identifiers (SRC).

Figure 26. Broadcaster packet over R3

The results from this stage, presented in Table 2, indicate changes in Packet Loss Ratio (PLR) at varying distances and data rate. At around 15 meters, the overall Packet Loss Ratio (PLR) at the listener was about 1.44%, with almost no loss at relays R1, R2, and R3, and a signal strength of approximately -60 dBm. At a higher data rate of 600 pkt/min, PLR remained 0% at R1, slightly increased to 0.52% at R2, and 0.34% at R3. Interestingly, the listener had an even lower PLR of 0.17%, better than R1 and R2. This improved performance at the listener is likely due to

the short distances between nodes, allowing other nodes to successfully deliver packets even if one node experiences a loss.

As the distance between nodes increases and the data rate rises, the Packet Loss Ratio (PLR) also significantly increases. At lower data rates, the PLR rises to 17%, but as the data rate increases, the listener and more distant nodes, such as R3, experience a sharp rise in packet loss. This pattern underscores the impact of both distance and data rate on network performance, with higher rates and greater distances leading to more substantial packet loss and decreased signal strength (RSSI).

Distance	data-rate	PLR /R1	PLR /R2	PLR /R3	PLR /L	RSSI
~15m	60	0%	0%	0%	1.44%	~-60dBm
	90	0%	0%	0%	1.00%	~-60dBm
	120	0%	0%	0%	0.83%	~-52dBm
	600	0.00%	0.52%	0.34%	0.17%	~-53dBm
~30m	60	3.27%	3.27%	11.70%	16.30%	~-68dBm
	90	2.87%	3.00%	9.03%	17.00%	~-68dBm
	120	0%	0%	4.68%	26%	~-81dBm
	600	1.06%	5.16%	5.16%	14.74%	~-77dBm

Table 2. Scenario 2 tests

Figure 27 illustrates the average Packet Loss Ratio (PLR) across all data rates for each hop, with measurements taken at intervals of 15 meters (indicated in blue) and 30 meters (indicated in red). Starting with R1, which exhibited a PLR of 0%, followed by R2 at 0.13%, R3 at 0.08%, and concluding with the listener node at 0.86%. This demonstrates that data transmission can be maintained over a 60-meter range with a packet loss of 0.86% at the listener node.

Additionally, the average PLR for all data rates over each hop in 30-meter intervals. shows that the total distance between the broadcaster and the listener, spanning 120 meters, results in a data loss of 18.5% at the listener node.

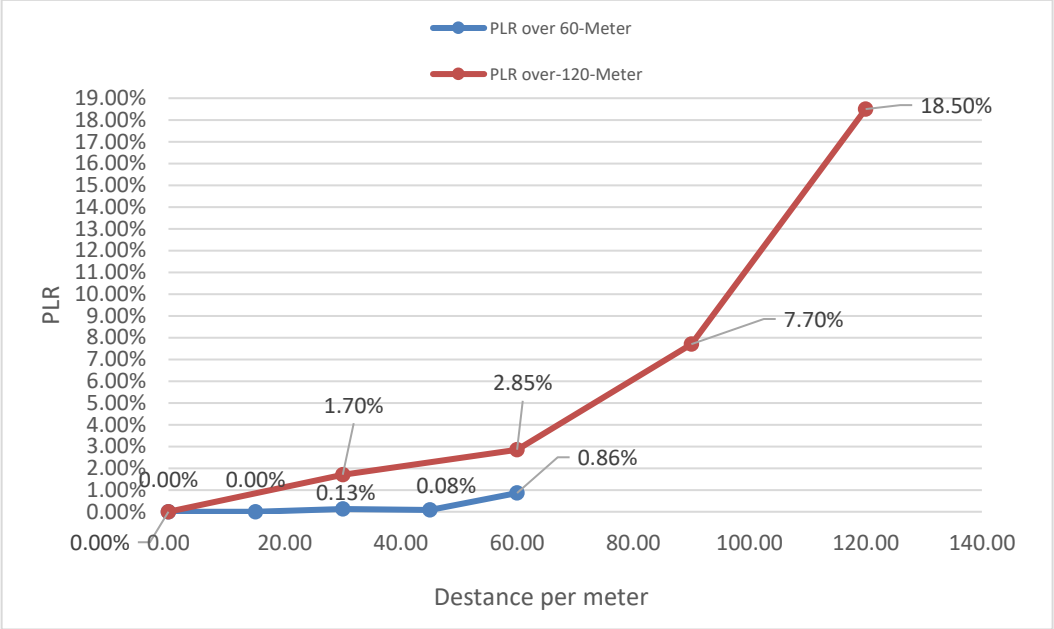


Figure 27. PLR over 60-Meters

These findings highlight the relay nodes' efficiency in extending the network's range and improving message delivery over greater distances. The relays still significantly reduced packet loss compared to single link transmission, especially at longer distances. The data shows that while the PLR increases with distance and data rate size, the presence of relays helps maintain lower PLR values, indicating effective message relay and improved network reliability.

4.3.3 Scenario 3 (Mesh topology)

In Scenario 3, the setup will be tested in an open area (outdoors) with nodes placed 15 meters and 30 meters apart (Figure 28). The configuration included two relay nodes (R1 and R2), two listener nodes (L1 and L2), This setup aimed to improve the Packet Loss Ratio (PLR) by leveraging the redundancy provided by multiple relay nodes.

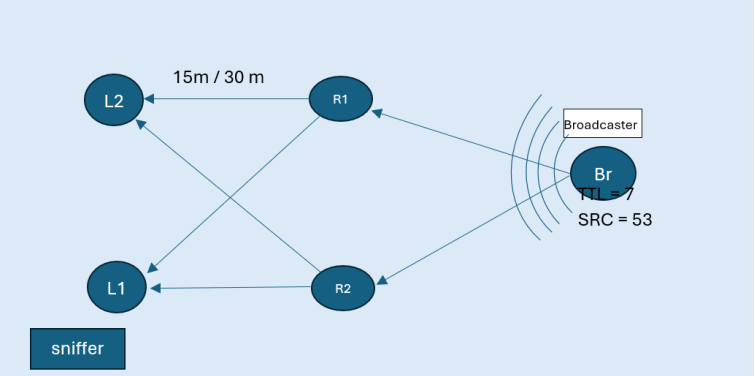


Figure 28. Scenario 3

As in the previous test, the broadcaster transmitted data at varying data rates with a source ID of 53 over a 60-second cycle, forming a 1-hop network. The relay nodes received these messages and forwarded them to adjacent nodes, decrementing the TTL by one with each hop. By the time the packets reached their destination, the sniffer monitored the packets from the broadcaster (SRC 53) and observed that the TTL value had been decremented by R1 and R2, reducing it to 6.

Data from Table 3 showed that at 15-meters, the PLR at listeners were minimal. For instance, for 120pkt/min, the PLR at L1 was 1.3% and 0.78% at L2 and 3.9% at R1 and 6.3% at R2, with an RSSI of approximately -57 dBm. At 30 meters, the PLR increased but was still better managed compared to single link transmission and chain topology. For example, for 120pkt/min, the PLR at L1 was 8.13%, while at R1 0% and R2 it was 0%, with an RSSI around -60dBm.

Distance	Data rate	PLR /L1	PLR /L2	PLR /R2	PLR /R1	RSSI
15m	60	0%	0%	0%	0%	~-51dBm
	90	0%	0%	0%	0%	~-51dBm
	120	1.30%	0.78%	6.30%	3.90%	~-57dBm
	600	0%	0%	0.16%	0.16%	~-54dBm
30m	60	0%	0%	4.76%	0%	~-57dBm
	90	0%	0%	0%	0.40%	~-57dBm
	120	8.13%	0.80%	0%	0%	~-60dBm
	600	1.98%	1.32%	1.48%	0.49%	~-64dBm

Table 3. Scenario 3 test

The analysis indicated that introducing multiple relay nodes improved the overall network reliability by reducing the PLR at the listener nodes. Such a setup will create two paths for data delivery to L1 and L2. Path through R1 (B-R1-L) and path through R2 (B-R2-L) showed that if a packet was lost at R1, it could still be delivered by R2. Figures 29&30 shows that despite PLR over R1 and R2 was 1.01% and 1.60%, PLR over L1 was 0.32% and L2 was 0.19%. This demonstrates that packets loss at R1 and R2 did not influence on L1 and L2, however it improves the deliverances.

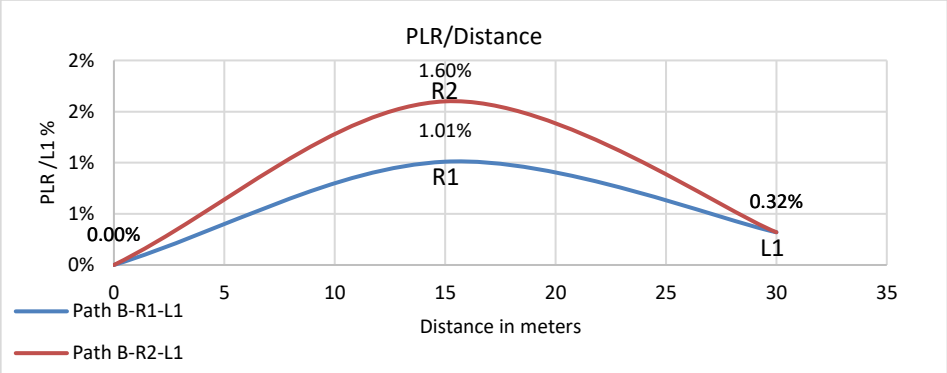


Figure 29. L1 paths over 30 meters

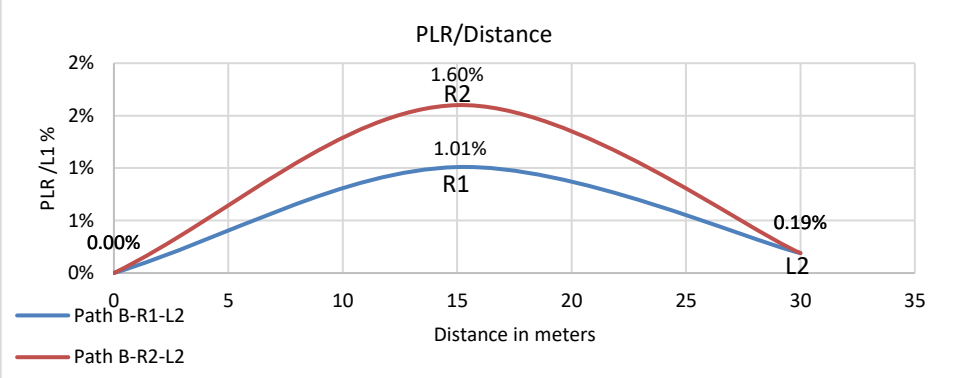


Figure 30. L2 paths over 30 meters

On other hand, Figure 31 shows that nodes placed at 30-meters distance. PLR over L1 is 2.52% while PLR at R1 was 0.22% and 1.56% was at R2. While PLR on L2 (Figure 32) was enhanced compared to R2. This shows as that on larger distances between adjacent nodes still can affect the network efficiency.

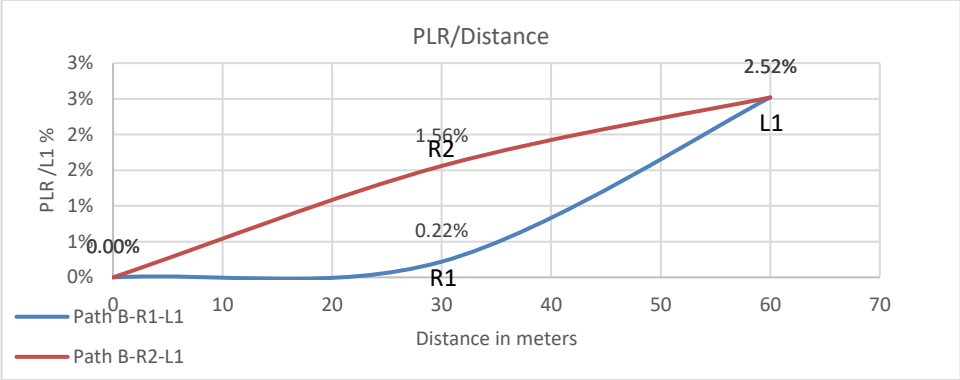


Figure 31. L1 paths over 60 meters

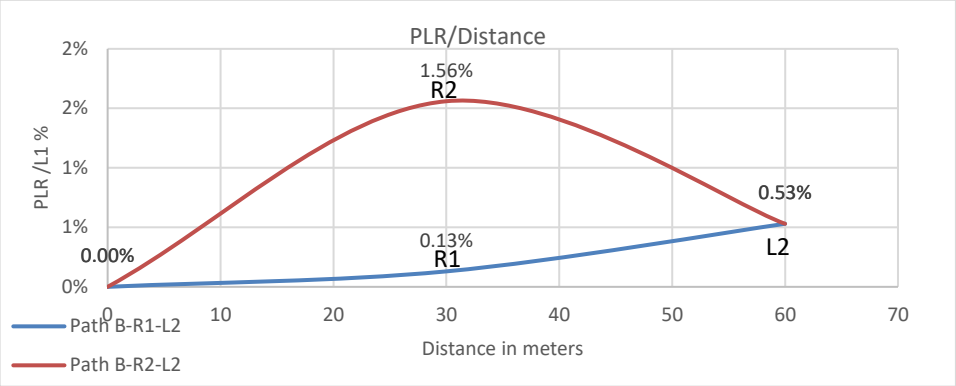


Figure 32. L2 paths over 60 meters

4.3.4 Scenario 4 (Hybrid Topology)

In this Scenario, depicted in Figure (33), same environment setup has been implemented as previous scenarios with two relays (R1 and R2), two listeners (L1 and L2), and one broadcaster.

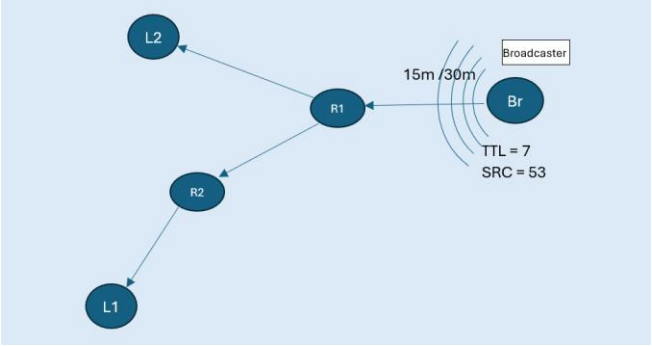


Figure 33. Scenario 4

As before the broadcaster transmits various data rate over 60-second. Broadcaster using a source ID of 53 and Time To Live value is 7. The messages are relayed by R1, which forwards them to adjacent nodes R2 and L2. TTL value will be reduced to 5 by the time it reaches L1 and to 6 by the time it reaches L2.

The results from Table 4 indicate that at a 15-meter distance, L1 exhibited higher Packet Loss Ratio (PLR) compared to L2. Specifically, L1 had PLR of 0.2%, 0.49%, 2.15% and 2.43% for data rate of 60, 90, 120, and 600, respectively, while L2 maintained a 0% PLR across all data rate. This difference demonstrates that L1 is more prone to packet loss because it relies on R2 for data, which in turn depends on R1, increasing the risk of cumulative packet loss. The RSSI values were approximately -63dBm, indicating strong signal strength but still some data loss at L1. In contrast, L2 has a lower PLR as it receives data directly from R1 and additionally from R2, providing redundancy. Therefore, if one relay loses some packets, L2 still has a chance to receive the data from the other relay, resulting in better data integrity. The analysis underscores the effectiveness of using multiple relays to ensure data delivery, particularly over extended distances and in environments prone to signal degradation.

Distance	Data rate	PLR /L1	PLR /L2	PLR /R2	PLR /R1	RSSI
15m	60	0.20%	0%	0%	0%	~-63dBm
	90	0.49%	0%	0.49%	0%	~-61dBm
	120	2.15%	0%	0%	0%	~-57dBm
	600	2.43%	0%	0%	0%	~-58dBm
30m	60	6.40%	0%	3.27%	0%	~-58dBm
	90	4.54%	0%	0%	0%	~-58dBm
	120	0.78%	0%	0.78%	0%	~-55dBm
	600	7.18%	2.20%	5.03%	0.50%	~-63dBm

Table 4. Scenario 4 tests

These findings highlight that L1, which requires two hops (45 meters) to receive data, experiences higher packet loss due to the compounded risk of packet loss at each hop (R1 to R2, then R2 to L1). Figure 34 shows the average of packet loss at L1 is 1.04% despite the PLR at R1 were 0 and R2 0.12%. This shows the more hops in the network more risk of losing data.

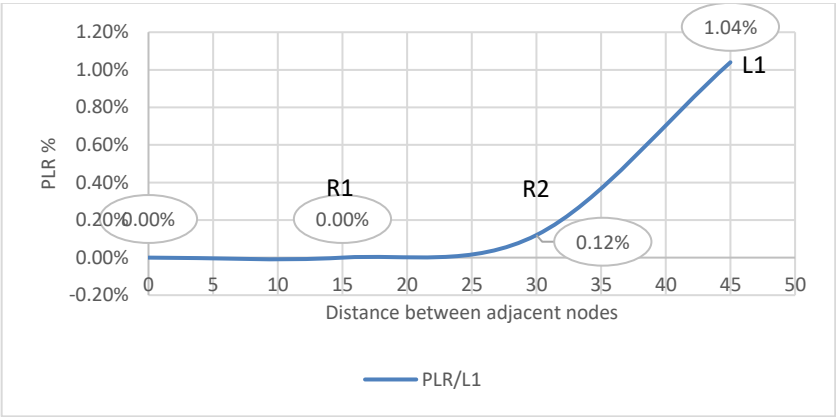


Figure 34. PLR over L1

In contrast, L2 benefits from redundancy, receiving data from both R1 and R2, which reduces its overall PLR. This scenario demonstrates the advantage of multiple relays in ensuring data delivery and the importance of relay placement in enhancing network reliability, especially over longer distances (90-meters). The observed PLRs and RSSI values support the conclusion that relay positioning can significantly mitigate packet loss and improve communication efficiency in mesh networks. Figure 35 demonstrates the nodes placed 30 meters apart and shows that PLR at node L2 was 0.55% at 90 meters of link distance, while R1 was 0.12% and R2 was 2.27%.

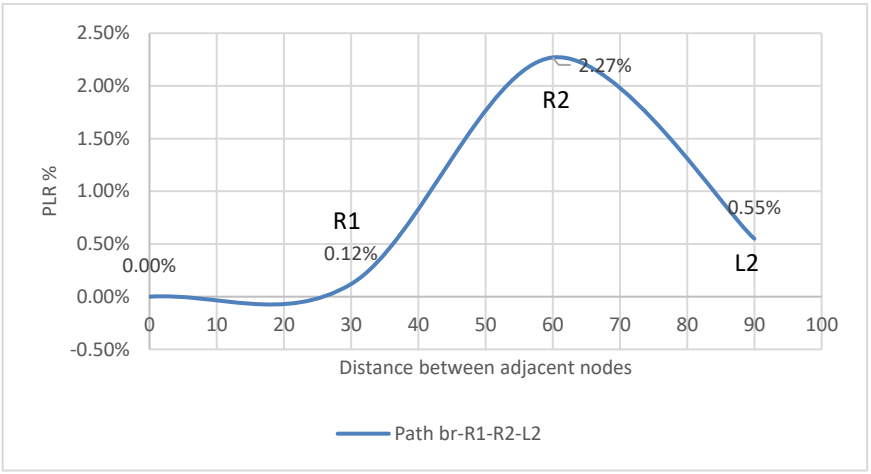


Figure 35. PLR over L2

4.4 Sensor Integration

After evaluating the reliability of mesh networks in large areas, the subsequent step involves integrating sensors into the network. This integration aims to assess the effectiveness of mesh networks as a solution for IoT systems, particularly in comparison to other technologies. By doing so, we can demonstrate the robustness and efficiency of mesh networks in managing IoT applications, highlighting their potential advantages in terms of scalability, coverage, and reliability.

Embedding sensors within the mesh network allows for the assessment of real-world performance metrics, such as data transmission consistency, latency, and power consumption.

Due to resource limitations, a small-scale network will be implemented to integrate sensor data and collect it from a collector node, as illustrated in Figure 36. This network will demonstrate how transmission can be directed to a specific destination address (destination ID 108, or 0x006c).

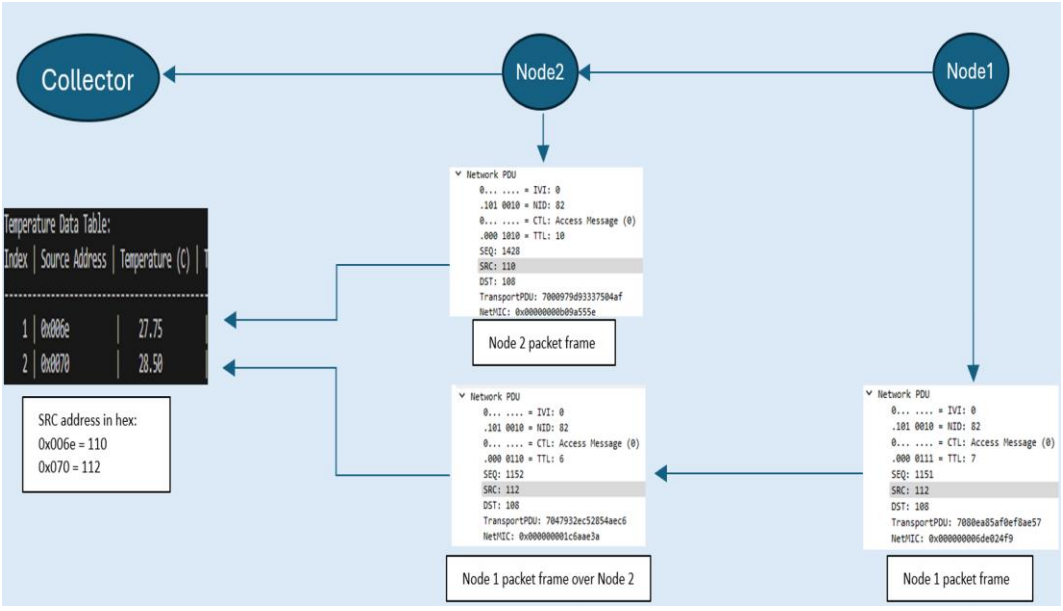


Figure 36. Network diagram

The diagram illustrates the process of temperature data transmission within a Bluetooth Low Energy (BLE) mesh network involving Node1, Node2, and a Collector node. Node1 and Node2 each measure their respective temperatures, with Node1 recording 28.59°C and Node2 recording 27.75°C. Each node then creates a Network Protocol Data Unit (PDU) that encapsulates the temperature data along with the Source Address (SRC), Destination Address (DST), Time To Live (TTL), Network Key (NetKey) and Application key (AppKey1) necessary for transmission within the mesh network. Node1 sends its PDU directly to Node2, which, in turn, relays both its own temperature data and Node1's data to the Collector. The collector node will listen to channels (37,38 and 39) and receives these PDUs, extracts the temperature data and stores it in a temperature data table (Table 5), displaying the source addresses and corresponding temperature values.

Temperature Data Table:

Index	Source Address	Temperature (C)	Timestamp (ms)
1	0x006e	26.50	4789
2	0x0070	34.25	4947
3	0x006e	26.75	4969
4	0x006e	26.75	5951
5	0x0070	34.25	6047
6	0x006e	26.75	6945
7	0x006e	26.75	7967
8	0x0070	34.25	8952
9	0x006e	26.50	8956
10	0x0070	34.25	9977
11	0x006e	26.75	9980
12	0x006e	26.50	10992
13	0x0070	33.75	11987
14	0x006e	26.50	11989
15	0x0070	33.75	12982
16	0x006e	26.50	13016
17	0x0070	33.75	13992
18	0x006e	26.25	14043

Table 5. Temperature data

5 CONCLUSIONS

This thesis has successfully explored the design, implementation, testing and optimization of a Bluetooth Low Energy (BLE) mesh network using Nordic Semiconductor nRF52-DK boards, specifically tailored for real-time environmental monitoring and alerting. Through detailed experiments and practical implementations, it was demonstrated that BLE Mesh networks effectively minimize packet loss and maintain strong signal strength at shorter distances, while the use of relay nodes significantly mitigates packet loss at longer distances. Different network topologies were tested, with the double relay setup showing superior performance in maintaining data integrity over extended ranges. Multi-hop configurations highlighted the potential for increased packet loss due to cumulative effects at each hop, but relay placement and redundancy significantly reduced these losses. The integration of environmental sensors validated the network's suitability for real-time monitoring applications, demonstrating efficient data transmission across nodes. These findings have significant implications for various industries, including industrial facilities, smart buildings, and environmental agencies, by providing scalable, cost-effective solutions for environmental monitoring. Future work could further optimize the network for greater distances and larger data rate, explore advanced error correction algorithms, and integrate more sophisticated sensors. In conclusion, this thesis contributes to the advancement of BLE Mesh networking technology, offering a robust framework for deploying scalable, reliable, and energy-efficient networks for environmental monitoring and beyond.

REFERENCES

1- Schalffer, N. (2023, January 10). 7 IoT Challenges in 2023 and How to Solve Them. Retrieved August 17, 2024 from, 7 IoT Challenges 2023 and How to Solve Them | emnify Blog

2- Afaneh, M. (2022, October 19). Bluetooth Mesh Networking: The ultimate guide. Novel Bits. Retrieved August 23, 2024 from https://novelbits.io/bluetooth-mesh-networking-the-ultimate-guide/

3- Baert, M., Rossey, J., Shahid, A., & Hoebeke, J. (25 July 2018). The bluetooth mesh standard: An overview and experimental evaluation. *Sensors*, *18*(8), 2409. https://doi.org/10.3390/s18082409

4- The bluetooth® low energy primer. Retrieved August 25, 2024 from (15th March 2024). the-bluetooth-le-primer-v1.2.0.pdf

5- Woolley, M. (2 December 2020). Bluetooth mesh networking. Bluetooth.com Retrieved August 17, 2024 from https://www.bluetooth.com/wp-content/uploads/2019/03/Mesh-Technology-Overview.pdf

6- Mesh Working Group. (2019, January 21). *Mesh profile*. Bluetooth® Technology Website. Retrieved August 17, 2024 from https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/

7- Technical documentation. NRF5 SDK for mesh V5.0.0: Bluetooth Mesh Concepts. (2020-11-18) Retrieved August 17, 2024 from.https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v5.0.0/md_doc_user_guide_mesh_basic_concepts.html?cp=9_2_2_0

8- NRF5 SDK for mesh V5.0.0: Bluetooth mesh stack architecture. (2020-11-18). Retrieved August 17, 2024 from https://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v5.0.0/md_doc_user_guide_mesh_basic_architecture.html?cp=9_2_2_1

9- Nordic. (2020-04-08). *Working with nRF52 Series*. Introduction. Retrieved August 17, 2024 from https://infocenter.nordicsemi.com/topic/ug_nrf52832_pdk/UG/nrf52_PDK/required_tools.html?cp=5_2_5_2

10- Nordic Semiconductor infocenter. (2020-04-08). Retrieved August 17, 2024 from https://infocenter.nordicsemi.com/topic/ug_nrf52832_pdk/UG/nrf52_PDK/hw_desc_intro.html?cp=5_2_5_5

11- nRF52840 Dongle User Guide v2.1.1 (2023-05-15). Retrieved August 17, 2024 from https://infocenter.nordicsemi.com/pdf/nRF52840_Dongle_User_Guide_v2.1.1.pdf

12- Natgunanathan, I. (2023, February 6). Bluetooth Low Energy Mesh: Applications, considerations and current state-of-the-art. MDPI. Retrieved August 25, 2024 from https://www.mdpi.com/1424-8220/23/4/1826

13- *Bench talk*. Bluetooth Mesh Networks and Home Automation | Bench Talk. (May 10, 2019). Retrieved August 25, 2024 from https://www.mouser.com/blog/bluetooth-mesh-home-automation

14- Getting started with NRF Connect SDK (NRF52 series). (2021-02-11). Retrieved August 25, 2024 from https://infocenter.nordicsemi.com/pdf/getting_started_NCS_nRF52_20210211.pdf

15- NordicSemiconductor. Nordicsemiconductor/IOS-NRF-Mesh-library: Provision, configure and control Bluetooth mesh devices with NRF mesh library. GitHub. Retrieved August 25, 2024 from https://github.com/NordicSemiconductor/IOS-nRF-Mesh-Library

16- Nrfconnect. SDK-zephyr/samples/Bluetooth/mesh at V3.5.99-NCS1 · nrfconnect/SDK-zephyr. GitHub. Retrieved August 25, 2024 from

https://github.com/nrfconnect/sdk-zephyr/tree/v3.5.99-ncs1/samples/bluetooth/mesh/

17- Nrfconnect. SDK-NRF/samples at main · nrfconnect/SDK-NRF. GitHub. Retrieved August 25, 2024 from https://github.com/nrfconnect/sdk-nrf/tree/main/samples

18- Introduction to devicetree - Zephyr Project Documentation. (2023, June 1). Retrieved August 25, 2024 from https://docs.zephyrproject.org/2.7.5/guides/dts/intro.html

19- GPIO — General purpose input/output. Technical documentation. (2021-11-08). Retrieved August 25, 2024 from https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Fgpio.html&anchor=concept_zyt_tcb_lr

20- All Configuration Options - Zephyr Project Documentation. (2023, June 1). Retrieved August 25, 2024 from https://docs.zephyrproject.org/2.7.5/reference/kconfig/index-all.html