Zigbee control design of pixelated light strips

Konetampet Gopalakrishnan, C.

Award date:
2019

Department of Mathematics and Computer Science
System Architecture and Networking Group

Zigbee Control Design of Pixelated Light Strips

Master Thesis

Chandana Gopalakrishnan
Student Id: 1377434
Email Id: c.konetampet.gopalakrishnan@student.tue.nl

University Supervisors:
Dr. ir. Reinder. J. Bril (SAN Group, TU/e)
Dr. Erik J. Luit (SAN Group, TU/e)

Company Supervisors:
Dr. ir. Gerhard Mekenkamp (New IoT System Architecture)
Ms. Sonia Kataria (IoT Application Creation)
Signify

Eindhoven, Friday 2nd August, 2019

Abstract

This paper discusses an end-to-end system design to have efficient control of LED strips by addressing the LEDs individually. These light strips are usually called individually addressable light strips on the market. In Signify research, such light strips are called pixelated light strips. Since the paper discusses an end-to-end system, as a first step, some of the use cases that could be interesting for pixelated light strips have been defined. The use cases for pixelated light strips are classified into two categories as static and dynamic use cases. This paper provides a definition for both the static and dynamic use cases and provides examples for each of these types. Looking at the rapid development of communication technologies, a large part of the internet-connected devices can be contributed to lighting. Wireless control of lights can be done through wireless radio networks. In this thesis work, a Zigbee network is used for lighting control. This thesis gives an insight into the Zigbee protocol and the possible Zigbee design extensions to control individual LEDs on pixelated light strips. Various experiments have been done to do performance evaluation of these designed Zigbee extensions with respect to metrics such as packet loss, scalability, latency, memory, processing power etc. The above mentioned performance metrics are defined in this paper and different experiments are carried out with respect to them. Based on the results generated from these experiments, a trade-off is made between different design extensions to decide on the best-fit Zigbee design extensions for both the static and dynamic use cases. For use cases, which don't fit into the current design, an alternative design is suggested to control the individual LEDs on the light strips and possible future work in the direction of this alternative design is suggested in this paper.

At the end of this work, a user interface using Google's UI framework, Flutter is developed to control the pixelated light strips through a mobile application.

Acknowledgement

I would like to thank my company supervisor Dr. ir. Gerhard Mekenkamp from Signify (Philips Lighting). I am so happy to have worked under you. The work that you have contributed to my thesis is equally important as mine. You have never failed to answer my questions, help me whenever I got stuck up in work, brainstorm many innovative ideas along with me. You have taught me various aspects in a research work and improved my analytical thinking. The feedback that you have given for my presentation skills and thesis writing were so valuable and I would like to thank you for going an extra mile to shape me professionally. I would like to thank Ms. Sonia Kataria from Signify, for being an energetic supervisor and guiding me throughout the thesis work. You were such an inspiration and have always motivated me. You taught me a lot about application development and thanks for being such a cool supervisor. I would like to thank both my supervisors for investing so much time in guiding me with my work.

I would like to thank Dr. Erik J. Luit and Dr.ir. Reinder. J. Bril from Eindhoven University of Technology for guiding me throughout the thesis work. Right from the beginning of my thesis you both have been so supportive. I would like to thank for the time and effort that you both have invested in my thesis work. Your inputs have been constructive and improved my work a lot. I have always been intrigued by your questions and it made me think a lot more in research perspective which has helped to improve my work. I thank you again for your immense support and motivation.

I would like to thank my fellow interns at Signify for being cool and supportive friends and for making me feel at home. I would also like to thank my fellow intern Dhyaneswaran for giving peer feedback and helping me with my thesis.

Finally, I would like to thank my family & friends back in India. Thanks appa, amma, Jyostna and Jyoshita for all your love and unconditional support. I would like to thank my well-wisher Povendhan for being there through my thick and thin.

Contents

List of Figures

List of Tables

Acronyms

ACK Acknowledgement.

APS Application Sublayer.

BLE Bluetooth Low Energy.

CCA Clear Channel Assessment.

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance.

IoT Internet of Things.

MAC Medium Access Control.

PHY Physical Layer.

RTT Round Trip Time.

SoC System on Chip.

WSN Wireless Sensor Networks.

WSTK Web Services Tool Kit.

ZC/ZR Zigbee Coordinator/Router.

ZCL Zigbee Cluster Library.

ZDO Zigbee Device Object.

ZED Zigbee End Device.

ZLL Zigbee Light Link.

Chapter 1

Introduction

This thesis aims to extend the Zigbee protocol standards and validate the scalability and performance of pixelated light strips for efficient lighting control. Strips which allow controlling individual leds are some times referred to as individually addressable LED strips. Within Signify the term pixelated is mostly used. Therefore, also in this thesis pixelated strips refer to led strips which have individually addressable LEDs on them. This thesis aims to control the individual pixels of this LED strip by extending Zigbee standard. Philips Hue is a smart home lighting system developed by Philips which not only controls switching lights on and off but can also be instructed to produce colors that can be controlled via a website or a smartphone. For instance, it can completely change the atmosphere in a room by setting different scenes according to the user's mood. Philips Hue light strips work similar to the Hue bulbs. These are flexible fixtures which have LEDs integrated on it. These light strips are available on the market and can change color and brightness of the LEDs dynamically. However, it is not possible to address individual LEDs on these light strips.

Currently, the Zigbee protocol does not support addressing of individual LEDs in a light strip. This project resulted in an end-to-end system design which includes various design options to control the individual pixels of light strips which still fits within the Zigbee standards. These design options can handle various pixelated use cases while still being cost effective. The design options should be able to set a nice static color pattern on the light strip as well as dynamically change the color pattern on the light strip, e.g, for streaming applications like reacting to sounds and music. Based on this, the pixelated use cases are broadly classified into static and dynamic uses cases. To implement these use cases, three design options are suggested in this paper, as discussed in chapter 5. To evaluate the Zigbee design extensions various performance metrics are defined. The metrics used are packet loss, latency, memory and CPU load. Various experiments were done with respect to the defined use cases to do a performance evaluation of the designed Zigbee extensions. A trade-off was made between various design options and the best suited designs for both the static and dynamic use cases were decided. Finally, an implementation was made using an (ARM-based) Zigbee chip from SiLabs at Signify on hardware nodes using the SiLabs chip. Through this, performance, scalability and reliability analyses of wireless LED light strips in the context of commercial lighting was done. To control the pixelated light strips from mobile apps, a user-interface was implemented using Flutter.

Advances in the wireless networking technologies have lead to development of several home automation technologies. Internet of Things (IoT) has been a part of home automation and has become an evolving technology that is expected to become mainstream in a few decades. According to a survey by IHS [1], the commercial and industrial sector, controlled by smart building automation, smart industrial automation, and efficient lighting, will account for nearly 50 percent of new connected devices between 2018 and 2030. Smart lighting plays a pivotal role, unlocking the power of the IoT. Signify (Philips Lighting), the world leader in lighting, has been ranked as the leading Influencer on Internet of Things (IoT) lighting by industry market research firm Navigant Research.

In the traditional home lighting, the conventional light bulbs work by flipping the switch on and off. Smart lighting, on the other hand, gives far more control over the lights. The lights are still connected to home's power, but each smart bulb and LED-integrated fixture allows to controlled wirelessly with phone, tablet, or smart assistant, such as Google Assistant or Amazon Alexa. The advantage of using Wireless Sensor Networks (WSN) for lighting systems over wired networks is multi-fold: it provides the ability to use the existing network infrastructure, ease of installation and maintenance. However, wireless networks brings in challenges such as reliability of communication, security, scalability etc. Based on the way in which the nodes are interconnected, there are different network topologies such as star, clustered tree and mesh which each have their own pros and cons. Among these network topologies, mesh networks provide a promising research direction in lighting systems as they have several advantages such as scalability and reliability.

Zigbee is more widely used in the integrated lighting domains mainly because of it's mesh topology. Mesh networks can be reliable because they are self-healing, such that a network can still operate even with a broken or faulty connection. In a lighting system, this means that the range is extended since lights can forward messages to other lights, and when some lights are powered off, new routes to remaining lights are automatically discovered. Additionally, Zigbee is low cost and provides low power consumption which is best suited for various embedded systems which are widely deployed for controlling and monitoring applications. For these reasons, the Hue ecosystem developed by Philips Lighting is also based on Zigbee[2]. Also, Signify uses the Zigbee Light Link (ZLL) standard[3] to enjoy the home ambience with its varied features. Since its launch in 2012 [2], Philips Hue has supported open interoperability using the Zigbee Light Link standard. This ensures that devices created by third parties can work alongside the official Philips Hue lights within the Hue ecosystem.

This work is carried out at Signify, which provides IoT lighting solutions such as commercial building lighting systems, street lighting, home lighting, etc. Sections 1.1, 1.2, 1.3 and 1.4 outline the remaining part of this chapter.

1.1 Problem Statement

This section discusses the main problem and the solutions to address these problems. Currently, the Philips Hue strips on the market do not allow control over their individual LEDs. Though the LEDs can dynamically change their color on the light strip, all LEDs produce the same color and brightness at a given instant. The idea of controlling individual LEDs on the light strips has not been widely researched. However, individually controllable light strips are available on the market and are used in professional entertainment systems, and also by hobbyists, e.g., controlled by Arduino. But a cost effective system design based on Zigbee does not exist. With the new hardware design of pixelated light strips as discussed in section 4.2 and more advancement in Zigbee technology, it should be possible to have control over individual pixels on the light strips. By having such a control on the light strips, they could be more personalized and could be used for several different purposes. However, the goal is not to come up with all possible innovative use cases for the pixelated light strips. Hence, the various commercial use cases for pixelated light strips have not been intensively studied in this research. The main aim is to achieve individual control of LEDs on the light strips by designing extensions to the existing Zigbee standard and carrying out a performance evaluation of the designed Zigbee extensions.

1.2 Research goals

This thesis concerns extending the Zigbee protocol to support control of individual LEDs in a LED strip. In order to control the LED strip, longer Zigbee messages must be sent to the strips to individually address the LEDs. However, the current Zigbee standard does not support sending longer messages (more than the maximum payload the Zigbee can handle). By defining and analyzing the use cases we derive the technical requirements for the extensions to the Zigbee protocol.

The prioritization of the use cases highly depends on efficient scalability of the LEDs on the light strips and their control using the zigbee protocol standards including delay constraints. It involves both analysis of the network architecture and the communication protocol design according to the application needs. Based on the analysis and conclusion from the experiments carried out on the Zigbee protocol extensions, the project aims to implement the best-suited protocol to control the pixelated light strips. The design was verified by making an implementation on a SiLabs chip using the SiLabs Zigbee SDK at Signify (formerly known as Philips Lighting). The required metrics that are used for the performance evaluation of the designed control methods is discussed in section 1.3. Based on these metrics, the implemented design is evaluated and is continuously iterated to achieve control of the pixelated light strips. Once the use cases for the pixelated light strips in the home environment were decided and a test set-up was available, the following research questions were addressed.

1. What is scalability in this context and how to achieve scalability (possible number of LEDs on a light strip) and reliability for an efficient control of pixelated light strips?
2. What are the metrics other than scalability and reliability that are considered to decide the best suitable protocol for pixelated light control?
3. What could be the best protocol design configurations (wireless control solution) for pixelated light strips?

Each research question mentioned above is answered or addressed in this paper. In order to address the first research question, a tradeoff between different design extensions is done with respect to the performance metrics. The term "scalability" in the third research question refers to the optimal number of LEDs that can be controlled efficiently in a given light strip, say a light strip of 1m length. The term "reliability" refers to a Zigbee protocol design which produces an overall acceptable delay up to 200ms beyond which the flicker in the light strip would be noticeable to the users. The term "efficient" means to control the light strips with as little power as possible, minimum latency and at low cost.

1.3 Performance metrics

Another research goal is the performance evaluation of extensions to the Zigbee wireless networking protocol from the user perspective. For lighting applications, the packet loss percentage is a very important performance aspect as it will have a direct influence on the user experience. While packet loss is remedied in most cases by retrying to send a message, this causes delays. A small amount is acceptable, but if delays are too long they will be noticeable by the user. For example, when a user presses a switch, the light is expected to turn ON within a certain number of milliseconds (<200ms). If this deadline is not met, there will be a noticeable delay in turning the light ON which is not acceptable in certain scenarios. This is applicable mainly to streaming applications where the light strips must be highly reactive to sounds and music. The designed Zigbee network has a low packet loss percentage. This means that there are many re-transmissions within the Zigbee stack. The number of re-transmissions will have an impact on the Round Trip Time (RTT). It is important to define the metrics for carrying out performance analysis and arriving at the Zigbee design protocol best fit for various use cases of the pixelated light strips. The following key performance metrics are used in this work to analyze the performance of the designed extensions to the Zigbee protocol.

- *Packet loss percentage* : This is defined as the ratio of the number of messages lost at the destination and the number of messages sent by the source. This metric relates to the Zigbee module.
- *Round Trip Time (RTT)* : RTT is defined as the total time taken for sending a Zigbee packet from the source to the destination node and the time taken for a response to be sent by the destination node back to the source upon successful transmission of the Zigbee payload.

- *Memory* : The amount of memory should not exceed what is available in a typical System on Chip (SoC) for Zigbee applications. In our design, the MG12 Mighty Gecko Mesh Networking Wireless SoCs for Zigbee are used to control the light strips. This metric is used to measure if the MG12 wireless SoCs have sufficient memory space to support the designed Zigbee extensions.

- *Processing Power* : Processing power is defined as the speed at which the CPU can pull data from its memory and perform calculations. The CPU used in this research runs up to 40 MHz clock speed.

Each of these performance metrics is described in detail in chapter 6.

1.4 Proposed approach

As a first step, specific use cases for the pixelated light strips are defined. To design extensions to the Zigbee protocol for controlling individual pixels of pixelated light strips, a simple test set-up is built in an office area (at Signify) using hardware nodes running a Zigbee stack. In this work, two hardware nodes acting as a switch and a light are used to build the test set-up using Silicon labs Mighty Gecko (MG12). The hardware node which acts as a light is connected to the light strip. Each of these hardware nodes has a Zigbee specified role (either Zigbee coordinator, router or end device) and has to perform functions defined at the application-layer level.

All the research goals are addressed in a sequential manner. The first research goal is addressed by performing certain calculations and estimations with the test set-up, to decide how many pixels could possibly be controlled with the newly devised Zigbee extensions. After determining the optimal number of LEDs on a light strip, performance metrics are to be defined (addressing the second research question) to carry out performance analysis on the design. Further improvements are done on the Zigbee design based on the performance evaluation and the third research question is addressed, arriving at the best protocol design for each of the defined use cases.

1.5 Contribution of this thesis

The major challenge in this work is evaluating the performance of Zigbee protocol extensions through experiments on a hardware setup. The evaluation results are highly dependent on various factors such as physical distribution of nodes, type of message traffic generated, environmental conditions etc. While all of these problems cannot be completely eliminated, this work aims to bring a trade-off between various performance metrics and gives an insight in the best possible Zigbee protocol extensions to control individual pixels of a LED strip. The main contributions of this work include:

- Investigation of the Zigbee specification and exploring different options supported by the software stack on the MG12 wireless SoCs to handle longer Zigbee packets.

- Sending longer data packets (more than 82 bytes) with hard-coded color values to the light strips.

- Performance evaluation of the extended Zigbee protocol standards.

- Design and implementation of a Flutter app to control the light strips via a mobile app.

1.6 Outline

This report is organized as follows. Chapter 2 provides a brief description about the value propositions of Philips Hue and the use cases defined for pixelated light strips. Chapter 3 provides a

brief background on the Zigbee protocol, which includes the Zigbee protocol stack, working procedures, network topology, Zigbee clusters and attributes. It also discusses cross-platform mobile application development. Chapter 4 focuses on related work on pixelated light strips and Zigbee Light Link standard (ZLL) for efficient lighting use cases. Chapter 5 describes the hardware design of the pixelated light strips, hardware platform and development tools used for lighting control with these light strips. In chapter 6, the Zigbee protocol extensions are explained. Chapter 7 discusses all the experiments done for analyzing the performance of designed extensions based on the performance metrics as discussed in the section 1.3. This chapter also discusses the results of the conducted experiments in detail. Chapter 9 concludes the work and presents possible research directions for future work.

Chapter 2

Value propositions and use cases

This chapter discusses the possible use cases for pixelated light strips and the requirements needed to implement them. Signify's research results need to be linked to value propositions. A value proposition in a business generally means, an innovation, service, or feature intended to make a company or product attractive to customers. Value proposition creates value to the customers by satisfying their needs. By customizing products and bringing innovation into its design, performance, price etc, it is possible to add value to the value proposition. The main value propositions of Signify's Philips Hue are,

- Ambience Creation
- Entertainment
- Well-being
- Comfort and Security

These value propositions also hold for pixelated light strips. Pixelated light strips are flexible light fixtures which use less energy and last longer. These light strips are extendable and easy to install. They can be mounted in different places according to users wish and convenience. They have integrated LEDs with small driver chips to control the pixels individually. In this chapter, the high-level use cases for pixelated light strips that are investigated in this thesis work are defined. The defined use cases are described in context of the four value propositions of Philips Hue. Various use cases that are explored during the thesis project are discussed in this section. Pixelated use cases are broadly classified into two categories, static and dynamic use cases. The use cases which involve setting up a nice static color pattern when turned on are considered as static use cases. The use cases which involve frequent updates of the light strips with a refresh rate of 30-50ms are considered to be dynamic use cases. Suitable design methods for these two types of use cases are realized. Comparison and trade off between the design methods is done based on the results from the performance analysis done on each design. By doing this, the best-fit Zigbee protocol extensions for each use case are decided in this thesis work. Sections 2.1, 2.2, 2.3, 2.4 provides further description of each value proposition and examples of current use cases and pixelated use cases.

2.1 Ambience creation

This section discusses the definition of ambience value proposition, the current use cases of ambience creation and the possible use cases for pixelated light strips for ambience creation.

2.1.1 Ambience definition

The value proposition is ambience creation. Ambience creation aims mainly to reflect and enhance the mood/atmosphere indoors by personalizing the Hue lights.

2.1.2 Current use cases

Currently, Philips Hue lights help to create the perfect ambience to suit the mood of the users. They can be used to have an adaptive ambience to adjust to the social setting. For instance, a nice ambience can be created while narrating bedtime stories for children at night. This can, for example, be achieved by using the color picker in the Philips Hue app as shown in fig. 2.1. With 16 million colors to choose from an app, a nice ambience can be set which matches the bedtime stories. Another example is creating a perfect ambience while hosting a dinner party. By creating a warm and classy glow in the background and bright lights in the dining area, a nice ambient atmosphere is set using Hue lights. This is usually done by choosing one of the scenes from the app. Scenes are basically presets of what the user wants the lights to do. The user can set a colorful ambience and store it in the app which can be recalled later when the user selects the stored scene. There are two types of scene creation[2]. The user can either pick a color from the palette as shown in fig 2.1 and by selecting a scene already stored in the app, represented by a picture.

2.1.3 Pixelated use cases

Pixelated light strips can also be used for ambience creation like the hue lights. One of the high-level use cases where pixelated light strips can be used for ambience creation is, slow dynamics i.e, the lights can change colors subtly such that the change is not noticeable by the users. Here the light strips give a more lively ambience. This could be done by scene creation similar to the Hue lights. Another use case of pixelated light strips is scene creation for natural effect. Scenes can also be used to create dynamic light effects to bring a feeling of nature inside homes, for instance like the Deep blue sea, arctic aurora, Lush green forestry etc, similar to fig. 2.2. This value proposition is considered to be a static use case. In order to implement the static use cases, it is necessary that the overall delay on the light strip must be less than 200ms.

Figure 2.1: Scene creation from color palette

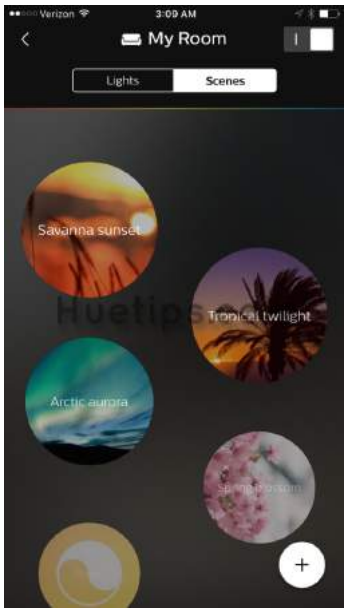

Figure 2.2: Scene creation from pictures

2.2 Entertainment

This section discusses the definition of entertainment value proposition, the current use cases for entertainment purposes and the possible use cases for pixelated light strips for entertainment.

2.2.1 Entertainment definition

The entertainment value proposition aims to entertain, i.e., to bring enjoyment and recreation to Hue users.

2.2.2 Current use cases

The current Philips Hue connected lighting system can sync to the movies, music or games played by the users to turn the room into a real entertainment space. Entertainment can be brought to the Hue users with Philips Hue Sync. Philips Hue Sync is a free PC/Mac application for unlocking immersive entertainment with Hue lights. It creates light scripts on the fly for any content that is played on a laptop or desktop, from movies and TV series to music and games.

2.2.3 Pixelated Use cases

Pixelated light strips can be put behind a TV cabinet or PC; this helps to extend the content of the PC game to the entire room with light and show special effects when gaming. By designing and deciding on a best-fit protocol, light strips can give a superior gaming experience. In case of gaming, the pixelated light strips could produce slow moving effects that match the game environments, or fast flashes during action packed sequences and thereby transforming the living space as shown in fig. 2.3

Figure 2.3: Gaming effect in Philips Hue

In case of music, these lights can transform the living room into a theatre by following the music that is being played in the background or by following the content on the television screen. In such cases, these light strips must be highly reactive (around 30ms) to the sounds[2].

In case of movies, usually when a picture is displayed on the television, the main colors present in the picture are extracted and sent to the light strips where a set of pixels or certain number of individual pixels react to the sound and to the color palette and produce color similar to that of the picture on the TV. This value proposition is considered to be a dynamic use case. In order to implement the dynamic use cases, it is necessary that a refresh rate of 25-30 ms is attained in order to update the light strips frequently to react to sounds and music.

2.3 Well-being

This section discusses the definition of the well-being value proposition, the current use cases of Hue lamps for well-being and the possible use cases for pixelated light strips for well-being.

2.3.1 Well-being definition

The third value proposition is well-being which mainly aims to maintain the health, happiness and prosperity of its users. Philips Hue produces many lighting effects. It can also have an influence on the mental health and satisfaction of its users. While well-being might not be the first thing that comes to mind when thinking about lighting at home, some of its impact is well known. For example, the effect of lighting on humans' circadian rhythm.

2.3.2 Current use cases

In this use case, the lights change over the day to mimic the rhythm of the daylight. A circadian rhythm is basically a 24-hour internal clock that is running in the background of peoples' brain and cycles between sleepiness and alertness at regular intervals. It is also known as the sleep/wake cycle. In this use case, there will be a gradual change of light over the day such that the users can hardly notice it but is in line with the users circadian rhythm.

2.3.3 Pixelated use cases

From the above described use case, the benefit of pixelated light strips is not so obvious. However, in Signify, a nice sunrise/sunset effect is also regarded as a part of the well-being proposition. Pixelated light strips can be used to keep a balance of its users' well-being by turning on and off at sunrise and sunset where the users wake up/fall asleep with a sunrise/sunset effect. By this, the

Figure 2.4: Sunrise and sunset effect in Pixelated light strips

light strips can make the morning waking hours and sleep routines more natural by reproducing the sunrise and sunset brightness and color pattern. This use case can be quite similar to the one shown in fig. 2.4. This figure depicts the use of a light strip that mimics the sunrise/sunset effect. For instance, it could represent a sunrise, starting at the bottom with an orange color, slowly climbing to a brighter yellow light at the top. This value proposition is considered to be a static use case. In order to implement the static use cases, it is necessary that the overall delay on the light strip is less than 200ms.

2.4 Comfort and security

This section discusses the definition of comfort and security proposition, the current use cases of Hue lamps in providing comfort and security and the possible use cases for pixelated light strips for comfort and security.

2.4.1 Comfort & Security definition

The last value proposition of Philips Hue is comfort and security where it aims to provide a sense of ease and reassurance among the users. The benefits of pixelated light strips for comfort and security is less clear at this point. However, some of the interesting use cases for this value proposition are described in this section.

2.4.2 Current use cases

In the current Philips Hue lamps, the lights turn on and then fade out after the users have left. Also, certain lights inside the home can remain on in the users' absence to mimic the users presence so that the house seems to be occupied when they are away. Similarly, the lights turn on by themselves when the users arrive home late in the dark.

2.4.3 Pixelated Use cases

Pixelated light strips can be used for similar purposes as mentioned above. This use case is applicable for both indoors and outdoors in which the lights notifies the users in case of certain events. The light strips can be placed along walking paths in the parking and garden area as shown in fig 2.6 to react to the sensors when the sensors detect the presence of humans when they arrive in dark. Indoors, they can be placed alongside the staircase as shown in fig 2.5, doors and beds to react to the sensors when it detects the users' movement in the dark and glow accordingly. For this purpose, motion and occupancy sensors can be used to detect the presence of people. These examples are given in the comfort aspect of the users. The pixelated light strips can also be used for home security. For instance, when the the lights turn on, an alert notification can be

Figure 2.5: Outdoor lighting for alert mechanism

Figure 2.6: Indoor lighting for alert mechanism

sent to the user when he/she is away from home. For this purpose, connected outdoor lights with presence sensing and ability to delegate security notifications to a contact is required.

Chapter 3

Technical Landscape

This chapter briefly describes the Zigbee protocol , which includes the Zigbee protocol stack, network devices roles, working procedures, network topology and security in section 3.1. A detailed description of the Zigbee payload is discussed in section 3.4 of this chapter. This chapter also explains ZLL which is a global standard used for consumer lighting in section 3.2.

3.1 Zigbee stack protocol

Zigbee is an IEEE 802.15.4-based specification for a suite of high-level communication protocols. It is a short range wireless technology with various advantages such as low complexity, low-power consumption, low data rate, low cost and duplex wireless communication. Zigbee protocols are intended for embedded applications which requires the some of the advantages mentioned above. This section discusses the stack architecture, network topology and security of the Zigbee stack.

3.1.1 Zigbee stack architecture

Zigbee operates in the industrial, scientific and medical (ISM) radio bands of 2.4 GHz in which Bluetooth Low Energy (BLE) also operates [4]; the data rates vary from 20 Kbits/s at 868 MHz to 250 Kbits/s at 2.4 GHz [5].The Zigbee stack architecture is built on top of the IEEE 802.15.4 standard. The software architecture of the 802.15.4 standard comprises four basic stack layers: Application layer, Network layer, Data Link layer and Physical layer as shown in fig. 5.2.1. The lower layers are the Physical Layer (PHY) and Medium Access Control (MAC) layers operating in a frequency range of 2.4 GHz. Zigbee uses the IEEE 802.15.4 physical layer and MAC layer [6]. The PHY provides the data transmission service, as well as the interface to the physical layer management entity. The PHY layer performs channel selection and energy and signal management functions. The MAC layer is responsible for accessing the shared medium using Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Its responsibilities may also include transmitting beacon frames, synchronization, and providing a reliable transmission mechanism [5].

The Zigbee specification includes four main components: network layer, application layer, Zigbee Device Object (ZDO) [7] and manufacturer-defined application objects. The Zigbee network layer supports different topologies like star, tree, and generic mesh networks. Within star networks, the coordinator must be the central node. Both tree and mesh topologies allow the use of Zigbee routers to extend communication at the network level. The Zigbee device type (Coordinator, Router, or End Device) is represented by the ZDO and is responsible for some tasks, including keeping track of device roles, managing requests to join a network, as well as device discovery and security. The next layer is the application layer. At the application layer, a large set of functions for various applications has been defined. These functions are grouped in clusters. Zigbee defines many standard clusters. For example, the on/off cluster, the level cluster (for con-

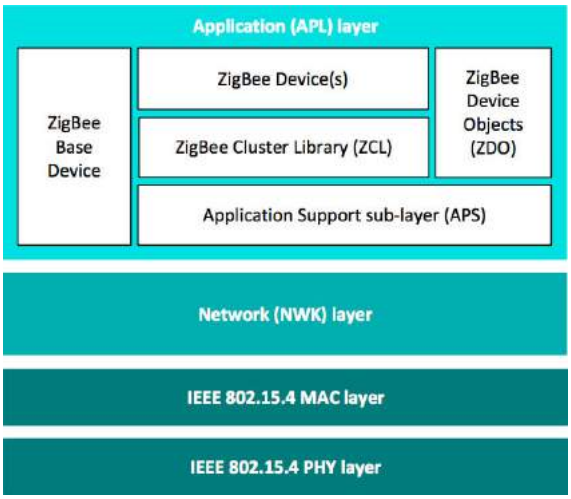

Figure 3.1: Zigbee Stack Architecture
[7]

trolling light levels), and the temperature measurement cluster. It is also possible to define new application-specific clusters. When a device supports very different functions like a light and a temperature sensor, the corresponding clusters can be assigned to different endpoints. Therefore, a Zigbee endpoint is basically a grouping of clusters which supports a certain application. Up to 240 application instances may be supported on a single Zigbee node. Each application instance communicates via an endpoint, where endpoints are numbered between 1 and 240 (note that endpoint 0 is reserved for the ZDO of the node). The Zigbee Cluster Library (ZCL) provides the standard Zigbee clusters used by the device applications that run on the endpoints. A clusters is a group of commands and attributes that are used by the device applications. ZDO deals with essential tasks for the whole node, such as commissioning and it does not occupy any endpoint. The APS provides the services necessary for application objects (endpoints) and the ZDO to interface with the network layer for data and management services.

Zigbee and its underlying network layers provide retries and acknowledgements designed to maintain efficient RF transmissions. The MAC layer attempts transmission five times for different types of messages. Zigbee message retries occur if the transmission channel was not clear ((Clear Channel Assessment (CCA) fails) or if the MAC level Acknowledgement was not received from the next hop destination. If the message is still not transmitted, then the network layer begins its retry mechanism. NWK layer attempts retries until either 'success' or the timer reaches 250ms [8]. If it reaches the 250ms, it reports failure to the APS layer. The APS layer has an Acknowledgement (ACK) flag. The APS layer waits for 100ms [9] and if the the acknowledgment is not heard, it asks the NWK layer to retry. APS layer retries up to 3 times. The Zigbee stack implementation made by Silabs performs the retry mechanism up to 12 times [9]and if the Zigbee message is not transmitted after 12 retries, the message is actually lost.

3.1.2 Zigbee network topology

The Zigbee standard supports mesh topology. Mesh networking is one of the key components of the Zigbee protocol. In a mesh network, nodes are interconnected with other nodes so that multiple pathways connect each node. Connections between nodes are dynamically updated and optimized through a sophisticated, built-in mesh routing table and this makes the mesh network topology highly reliable. A mesh network consists of one Co-ordinator, Routers and End devices as shown in fig. 3.2. Coordinators and Routers can be both parent nodes and have child nodes whereas End devices cannot have child nodes. There are certain rules for a mesh network,

- A Router can directly communicate with its child nodes, and with any other Router or

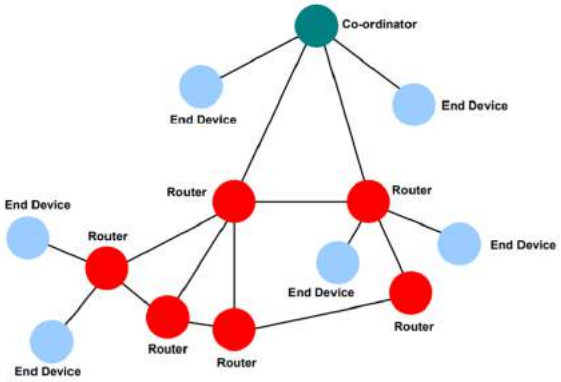

Figure 3.2: Zigbee mesh topology
[8]

Co-ordinator within radio range.

- An End Device can only directly communicate with its parent.
- The Co-ordinator can directly communicate with its child nodes and with any Router within radio range[6].

3.1.3 Zigbee security architecture

Another interesting feature of Zigbee is that the protocol facilitates secure communication. Zigbee allows developers and designers to build robust networks with the help of its strong security policies and the ease of deployment. It builds on the basic security framework defined in IEEE 802.15.4. Zigbee's security is based on symmetric-key cryptography, in which two parties must share the same keys to communicate. Zigbee uses the highly secure 128-bit AES-based encryption system [6]. The Zigbee security architecture includes security mechanisms at the network (uses network key) and the application (uses link key) layers. ZDO manages the security policies and configurations. Security services such as security key management, data stream encryption and decryption are provided by the network and application layers. Zigbee provides two types of network architectures. Their corresponding security models are shown in figures 3.3 and 3.4.

- Distributed security model- This security model consists of routers and end devices. A distributed network security model is formed when the Zigbee router does not detect an existing network when it powers up. In a distributed network, any router can issue network security keys[10]
- Centralized security model- For higher security, centralised systems include a third device type: the Trust Center (TC), which is typically also the Network Co-ordinator. The TC forms a centralised network and allows routers and end devices to join the network if they have proper credentials. These credentials are normally the centralised keys that are set by the developers in the code base. In a centralised network, only the TC can issue encryption keys[10]

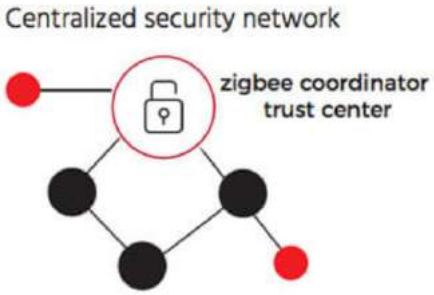

Figure 3.3: Centralized security model [10]

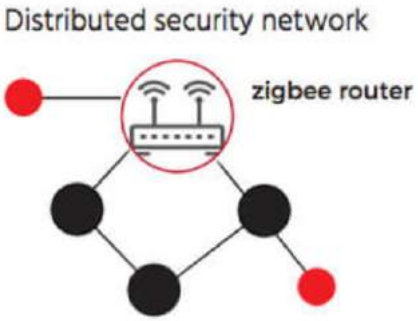

Figure 3.4: Distributed security model [10]

3.2 Zigbee Light Link standard

The Zigbee alliance supports consumer lighting solutions with the help of the ZLL. While the conventional Zigbee network requires a Zigbee co-ordinator for network formation or joining, the ZLL standard requires a commissioning application called Touchlink to carry out this functionality[3]. ZLL devices are of two types, controller devices and light devices.The controller devices may include light switches (e.g. on walls), occupancy sensors, remote control unit(s), smart phones and computing devices.The light devices include on/off light, dimmable light, color light, extended color light, and set color temperature. In the ZLL stack, the End devices are Controllers whereas the lighting devices are Routers[3]. In this project, the switch device is a Controller which is configured as a Zigbee End Device (ZED) and a light microcontroller which is a light device which is configured as the Zigbee Coordinator/Router (ZC/ZR).

ZLL networks use 16-bit network(short) addresses to identify devices. Zigbee also supports groupcasts[3]. In this case, groupcasts are used to address a subset of devices. They are typically used by a controller application residing at a certain endpoint, e.g., a remote control unit sending a groupcast to turn off all lights in a room. The ZLL protocol stack is shown in Fig. 3.5. The ZLL standard uses the IEEE 802.15.4 physical and MAC layers. The manufacturer application connects the ZLL Standard to the actual hardware of the manufacturer. In addition, manufacturer-specific extensions can also be added. The main advantage of using the ZLL standard is that it provides not just the application language (based on the ZCL) but also the coordinatorless commissioning (Touchlink) mechanism and associated security. ZLL uses AES 128 encryption to protect the lighting network against unauthorized use.

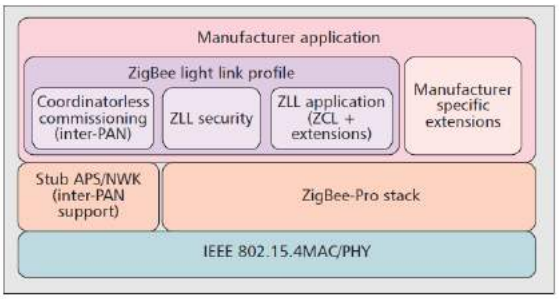

Figure 3.5: ZLL protocol stack [3]

3.3 Key concepts used by the APS layer

This section summarizes the key constructs used by the ZigBee networking stack's Application Sublayer (APS) to interface the stack to the application profiles. Application profiles, Clusters and End points are the main Zigbee concepts discussed in this section.

- Zigbee nodes - A Zigbee application and stack on a single network.

- Application profiles - Application profiles determine the application-level features and are identified by 16-bit application profile IDs. These profile IDs are assigned by the Zigbee Alliance and private profile IDs can be requested for custom applications or use one of Zigbee's published application profiles. Some examples of application profiles are Home Automation (HA), Commercial Building Automation (CBA) and Manufacturer-Specific Profile (MSP).

- Zigbee Cluster - A cluster is a set of message types relating to a certain device function[11] (e.g: ON/OFF functionality, color control). It is addressed by a 16-bit cluster ID[11]. A device can support multiple clusters to perform a wide variety of tasks. A large number of clusters are defined by the ZigBee Alliance and are listed in the ZCL which defines clusters for use in public profiles. Clusters consists of group of attributes and commands. ZCL groups cluster into "Functional domains" (e.g: Lighting, Measurement & Sensing). There are also Manufacturer Specific clusters that are defined by the manufacturer of the device.

- Zigbee Endpoint - An endpoint is a service point within a Zigbee node. It may vary from 1 up to 255 endpoints per node. Endpoints 0 and 240-255[11]are reserved for special functions whereas endpoints 1-239 are available for user applications. Each endpoint implements a single device type from a single application profile.

- Zigbee device - A specification that defines a unique device identifier and a set of mandatory and optional clusters to be implemented on a single endpoint. The functionality of a Zigbee device is defined by a set of supported clusters[11].

3.4 Analysis of the Zigbee data payload

This section discusses the maximum payload acceptable by the Zigbee stack, supported by MG12 Mighty Gecko wireless SoCs from Silicon Labs. Currently the Philips Hue lamp uses the ZigBee wireless protocol to communicate with the bulbs, telling them to change their color according to the specified settings. For this purpose, a data value of atmost 5 bytes is expected to be sent through Zigbee messages.

There are various color models to control lights like the Hue Saturation (HSV) model, the Color Temperature (CT) model etc. In the HSV color model, the lights are controlled with hue and saturation values, whereas in the CT model the lights are controlled with color temperature vales. In Philips Hue, the lights are mostly controlled with xy and brightness values. x and y are values from the color space chromaticity diagram ranging from 0 to 1. This typically requires 5 bytes (2 bytes each for x and y and 1 byte for brightness) for each pixel.

To control the pixels on the light strip individually, color values must be explicitly set for each pixel. Hence, to have control over individual pixels in a light strip, a large number of bytes must be sent through Zigbee commands to the light microcontroller. However, Zigbee can handle only up to 82 bytes of payload at the APS layer when security is implemented at the network layer. In a non-secure mode (i.e. without security,) the APS layer can handle up to a payload of 100 bytes.Hence, each unicast transmission may support up to 82 bytes, Since the maximum payload of Zigbee is limited, a detailed analysis of the Zigbee headers is provided in the following section.

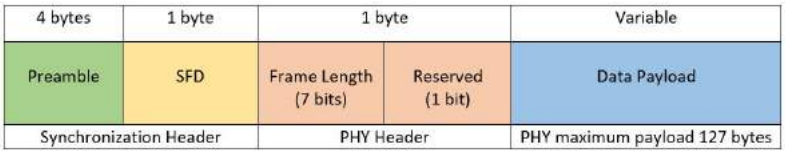

Figure 3.6: Physical layer frame structure
[12]

3.5 Detailed Analysis of Zigbee payload

The different layers in Zigbee are Physical, Media Access Control (MAC), Network and Application Support Sublayer (APS) layer. Each layer requires headers which reduces the available payload size at the next layer. The frame format of the different layers of the Zigbee stack are discussed in this section.

The Physical layer packet consists of a Physical Service Data Unit (PSDU) which can have a payload up to 127 bytes and Synchronization and Physical headers. The synchronization header consists of a Preamble (4 bytes) which allows the devices in the network to synchronize easily with their receiver clocks and is followed by a SFD (Start Frame Delimiter) to mark a new incoming frame.The frame format of the Zigbee Physical layer is shown in fig.3.6

The IEEE 802.15.4 MAC layer frame is composed of a MAC header, MAC payload and Frame Check Sequence (FCS). Figure.3.7 shows the frame format of the MAC layer. The generic MAC layer frame has a frame control field of 2 bytes. It carries useful information such as frame type and source and destination addressing modes. The Frame type specifies whether the frame is a beacon frame, data frame, ACK of data, MAC command frame etc. The sequence number header consists of a 1-byte sequence number. The MAC layer frame also consists of a 2-byte value called the PAN ID which is unique to each network and is randomly chosen to identify the network. The MAC layer has a 64-bit MAC address[5] that is assigned to each device during its manufacture. This is unique to each device and never changes. The MAC layer has 2 bytes each to represent the destination and source address. The Frame Check Sequence (FCS) is a 2-byte CRC Checksum. It is an error detecting code added to a frame while transmitting from the source to the destination.

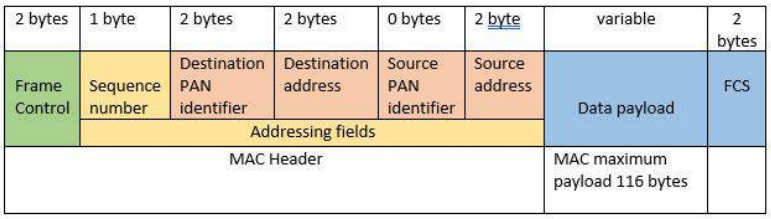

Figure 3.7: MAC layer frame structure
[12]

The frame format of the network layer is shown in Fig.3.8. The network frame consists of a NWK header and a payload. As seen in Fig.3.8, the frame control field is 2 bytes in length and contains information about the frame type and other sub fields. The sub-fields can indicate the route discovery, multicast messaging, security etc. The discover route sub-field may be used to control route discovery operations for the transmission of a specific frame. The multicast flag is set to 1 if the transmission is a multicast or set to 0 if the transmission is either unicast or broadcast. The destination address field is always present and has a length of 2 bytes. If the multicast flag sub-field of the frame control field has the value 0, the destination address field holds the 2-byte network address of the destination device or a broadcast address[5]. If the multicast flag sub-field has the value 1, the destination address field holds the 2-byte Group ID of the destination multicast group. The source address field is always present and is 2-bytes in length and contains the network

address of the source device of the frame. The radius field specifies the range of a radius-limited transmission. The 1-byte sequence number field is present in every frame. The sequence number value is incremented by 1 with each new frame transmitted. Mostly, the security is enabled at the network layer and it comprises of an 18-byte security header. The security header consists of Security frame control(1 byte), frame counter (4 bytes), IEEE 802.15.4 source address (8 bytes), Key sequence number (1 byte) and Message Integrity Code (4 bytes)[5].

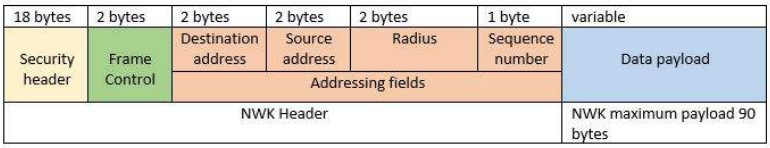

Figure 3.8: Network layer frame structure
[12]

The APS layer sits on top of the network layer in the Zigbee stack. The APS frame is composed of an APS header and an APS payload. The fields of the APS header appear in a fixed order, however, the addressing fields may not be included in all frames. The general APS frame is formatted as illustrated in Fig.3.9.

The 1-byte frame control field defines the frame type. The 1-byte destination endpoint field specifies the endpoint of the final recipient of the frame. The 2-byte cluster ID specifies the identifier to the cluster to which the frame relates[5]. The profile ID is only present for data or acknowledgement frames and specifies the Zigbee profile identifier for which the frame is intended.The 1-byte source endpoint specifies the endpoint of the initial originator of the frame. A source endpoint value of 0x00 indicates that the frame originated from the ZigBee device object (ZDO) resident in each device[5]. The APS counter is used to prevent duplicate APS frames. The APS security header is usually not enabled at Signify, it is considered that enough security is provided in the network layer for transmission of unicast messages. However, when the security is enabled at the APS layer, the APS layer takes an extra 9 bytes for the APS security header which consists of security frame control (1 byte), Frame counter (4 bytes)[12], and Message Integrity Code (MIC) (4 bytes). Depending on the security at the APS layer, the data payload in APS layer thus varies from 75 to 82 bytes.

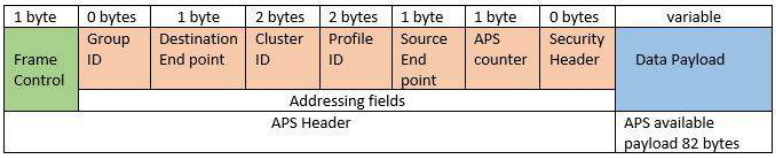

Figure 3.9: APS layer frame structure
[12]

The Zigbee application layer offers support for two types of network messages:

- ZDO (ZigBee Device Object) - ZDO messages provide facilities for a device to discover other devices and their services. It is used in the basic management of a Zigbee network.
- ZCL (ZigBee Cluster Library) - A Zigbee application allows the Host to create and transmit custom ZCL unicast, ZCL multicast and ZCL broadcast messages as well as receive responses to ZCL requests.

In this thesis, the network messages are ZCL messages. ZCL Messages are either profile-wide general commands (e.g., Attribute Read / Attribute Write) or cluster-specific commands (e.g., Move to Level, On/Off State Update, etc.). Incoming ZCL general commands (e.g., ZCL Read

Attribute Request) are handled automatically by the application. Incoming cluster-specific commands, if recognized by the application, are relayed to the Host as cluster call-back commands. If the application does not recognize an incoming cluster-specific command, the message is dropped. Zigbee packets must contain at least 3 bytes for the ZCL header. The format of the ZCL header is shown in 3.10

Frame Control	Manufacturer code	Transaction sequence number	Command identifier
1 byte	2 bytes	1 byte	1 byte

Figure 3.10: ZCL Header structure
[12]

The manufacturer code consists of 2 bytes and is used only when Manufacturer Specific Clusters (MSC) are used in the implementation. The 1-byte command identifier specifies the ZCL cluster command, for instance read,write etc. In the current design, the information that is sent in the ZCL header while transmitting longer Zigbee messages are

- Frame Control
- Manufacturer Specific Code - For manufacturer specific clusters, it is important to set the manufacturer-specific code before reading/writing commands to that specific cluster. If this is not done, the stack returns an error saying it is an unsupported command/attribute. Hence 2 bytes are included in the ZCL header specifying the manufacturer code. The manufacturer code used in this Zigbee design is 0x100B and the corresponding manufacturer name is Signify.
- Sequence number - Since the longer packets can contain several fragments depending on the total packet size, a sequence number is included in the header to identify if all the fragments have been received in order; if a specific fragment is lost during data transmission, it can be easily identified using the sequence number and can be re-transmitted.
- Command identifier - Custom clusters consist of various custom attributes and commands. There can be various custom commands like Pixel on (to switch on the light strip), Pixel Off (to turn the light strip off). In order to specify which command is addressed in a given Zigbee message, 1-byte command identifiers are used.

Chapter 4

Implementation Details

This chapter describes the specific hardware and software tools and how they are used in the thesis for designing a suitable zigbee protocol design to control pixelated light strips in sections 4.1 and 4.4. This chapter also discusses the architecture of the test setup in section 6.3and the hardware design of pixelated light strips in section 4.2.

4.1 Hardware

The main component involved in the test setup implementation is the EFR32MG12 Mighty gecko wireless SoC from Silicon Labs. The Mighty gecko SoCs is used because it is ideal for designing energy-friendly, wireless connected devices. The EFR32MG family includes wireless networking stacks for Zigbee, Thread, and Bluetooth Low Energy[13]. The EFR32MG12 family also includes support for proprietary wireless protocol development, Low Power Wide Area Networks etc. Some of the typical applications supported by Mighty gecko SoCs include IoT multi-protocol devices, connected home, lighting, health and wellness, metering, home and building automation and security.

Fig. 4.1 shows the EFR32MG12 SoC module, where the board marked in red is the actual radio board containing the chip set; the other hardware is the wireless Web Services Tool Kit (WSTK) main board which is used for flashing firmware, debugging, power input, resetting and various other purposes[13].

Table 4.1 indicates some of the specifications of the hardware that is used for the test setup. This hardware solution provides industry-leading energy efficiency, ultra-fast wakeup times, a scalable power amplifier, and no-compromise MCU features[13]. The devices are well suited for battery operated applications as well as other systems requiring high performance and low energy consumption.

Feature	Specifications
Processor	32-bit ARM Cortex M4
Programmable Flash Memory	1 MB
RAM	256 KB
Voltage	1.8 to 3.8V
Clock Speed	40 MHz
Compatibility	BLE, Zigbee, Thread protocols

Table 4.1: SiLabs EFR32MG12 hardware Specifications
[13]

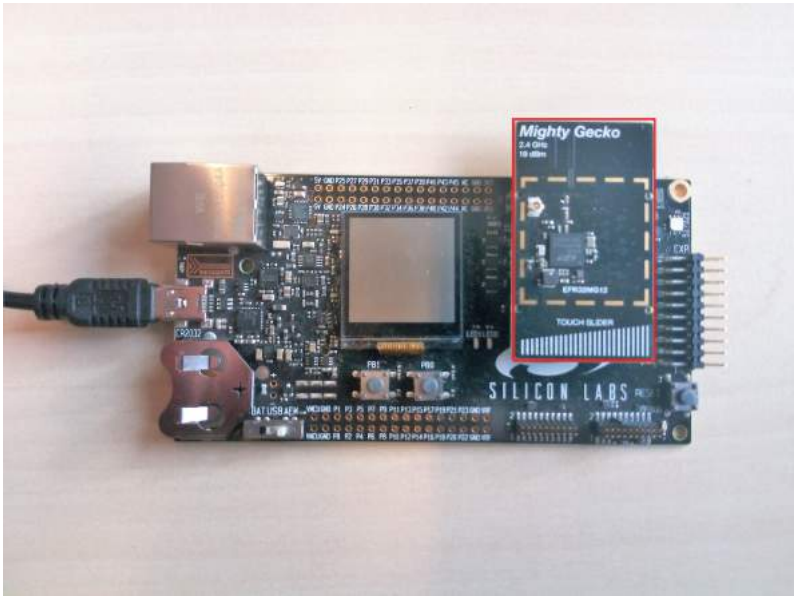

Figure 4.1: EFR32MG12 Mighty Gecko Multi-Protocol Wireless SoC

4.2 Design of pixelated light strips

The pixelated light strips consists of integrated RGB LEDs with TLC5971 LED drivers to control these LEDs on the light strip as shown in fig 4.2. TLC5971 is a 12-channel, 16-bit, PWM (Pulse Width Modulation) RGB LED driver[14].Each channel has an individually-adjustable, 65535-step, PWM grayscale (GS) control. The LEDs on the light strip are addressable with this TLC5971 driver which consists of a clock, data lines and a shift register[14]. These serial data lines and clock shifts in 8-bit values for red, green and blue. Then the driver generates a PWM pulse whose width is controlled by an 8-bit value. The driver outputs the PWM pulses to the red, green and blue LEDs through a SPI which is a synchronous serial communication interface specification used for short-distance communication[14]. Each driver controls 2 pixels which is basically 6 LEDs. In this hardware design of light strip, 3 LEDs are internally wired and are called a pixel. A pixelated light strip of 1m has 16 pixels. It is possible to control a individual pixels e.g, setting the 16th pixel on a light strip to blue and as well as control a set of pixels on the light strip.

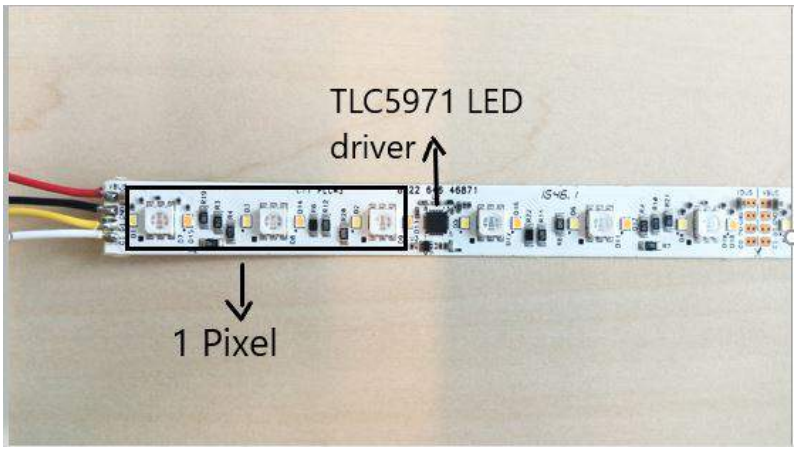

Figure 4.2: Pixelated Light strip

4.3 Test set-up implementation

Fig. 6.3 shows the test setup for implementing the Zigbee protocol extensions. The figure shows the EFR32MG12 SoC module, where the hardware board marked in red is the switch microcontroller which is configured as a ZED and the board marked in green is the light microcontroller which is configured as the ZC. The light microcontroller is connected to the pixelated light strip. The light strip is connected to a 12V power supply. The ZC initiates the process of network formation and forms a centralized network. ZED scans for the network and joins the centralized network set up by the initiator. The ZC communicates with the light strip through a SPI bus.

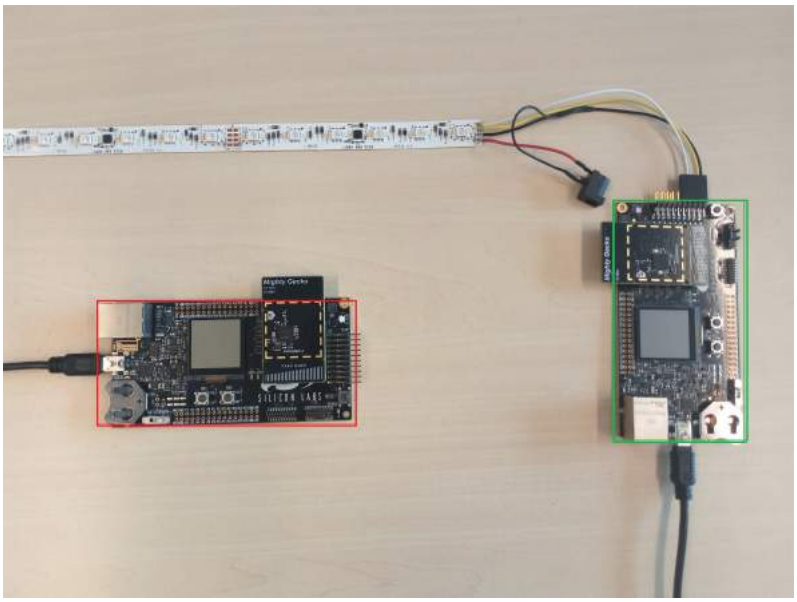

Figure 4.3: Test setup implementation

4.4 Development Tools

This section discusses the various development tools needed to implement the Zigbee design on the hardware (EFR32MG12 Mighty Gecko SoCs)

4.4.1 Simplicity Studio

Simplicity Studio is an Integrated Development Environment provided by Silicon Labs. It simplifies the IoT development process and includes a powerful suite of tools for energy profiling, configuration and wireless network analysis, demos and software examples. The app builder provides various features under different tabs. Some of the important features of the app builder, used in this work are listed below. Note that, only some of the features that seemed to be interesting and that have been used in this thesis work are listed below and do not include all possible features provided by the tool.

- Energy Profiler - This provides a good way to visualize, debug and optimize energy consumption.
- Network Analyzer - This feature uses a packet tracer and analyzes networks which improves wireless connections.
- RTOS - This provides a real-time operating system called Micrium OS which can be used to run tasks with priorities, for triggering task execution from interrupts etc.

- Serial interface - This provides functionality to send/receive commands using serial USB.
- General - This shows the hardware architecture of the boards used and the application configuration. For instance, the Zigbee networking protocol stack used in this project is EmberZNet PRO version 6.5, which is a complete Zigbee protocol software package containing all the elements required for robust and reliable mesh networking applications on Silicon Labs' Ember platforms. The toolchain used to build the application is GNU ARM v7.2.1 which provides a suite of tools used for developing software applications.
- ZCL clusters - This tab in the app builder allows endpoint configuration. It is possible to select a specific endpoint and Profile Id for that endpoint. A list of clusters with their attributes and commands are displayed under this tab. New custom clusters that are created in the project are visible under this tab.
- Plugins - This tab shows plugins are manufacturer implementations that can be optionally included based on the application requirements. There are several plugins with different functionalities. For instance, a plugin called "Fragmentation" is used in one of the design implementations as described in section 5.2.1.
- Includes - This tab includes various custom events and their functions. For instance, finding and binding event control allows a node in a network to pair up with another another node in the same network. It is possible to create custom events. In this work, a new custom event called "LED Strip display event handler" was created to produce a delay of 3000 ms between different color patterns on the light strip.

4.4.2 Flutter

A user interface is created for demo purposes to test and control the pixelated light strip through a mobile application in order to be more user-friendly. The mobile application is developed on a cross platform. For this purpose, Googles mobile UI framework Flutter is used for crafting high-quality native development environment experiences on iOS and Android[15]. A user interface app can be built with Flutter using an editor plugin which supports Flutter. For this purpose, Android studio 3.0 is required since it supports both Flutter and the Dart languages[15]. The app communicates with the ARM processor using the BLE protocol. Further details about the Flutter implementation are described in chapter 7.

Chapter 5

Proposed Designs

This chapter discusses possible Zigbee design extensions to control individual pixels of a LED strip. As discussed in section 1.1, the current Philips Hue strips can change the LED colors dynamically, however the color and brightness of the LEDs on the LED strip cannot be controlled individually. While LED strips exist which support controlling individual pixels, there are no standard Zigbee commands to support this. In order to mitigate this problem,various options for designing (manufacturer specific) extensions are described in this section.

In Philips Hue, the lamps are usually controlled by sending short Zigbee messages. For instance, values between 0 to 255 values are used to adjust the brightness of lamps using Zigbee commands in level cluster. However, to control individual pixels of a LED strip, longer Zigbee packets must be sent to the light strips. The current ZCL commands do not support longer Zigbee packets. In order to mitigate this problem, two design options are proposed in this section.

- Stack Fragmentation - This design option involves fragmentation of longer Zigbee packets by a frameworkk provided by the Zigbee stack

- Handling data subsets - This design option involves sending subsets of a long data array to the light strips.

The detailed description of the above mentioned design options and the requirements needed to implement them for lighting control is explained in this chapter.

As discussed earlier in section 3.4, we know that the Zigbee payload supports only 82 bytes including the ZCL header at the APS layer. The standard ZCL header is 5-bytes in length as described in section 3.4. However, depending on the design, this number can vary up to 7 bytes. The additional ZCL headers in the Zigbee payload is described below.

- Number of pixels - For certain use cases, which don't really require individual control of each pixel, the number of pixels to be controlled is given in the header and the message is transmitted. Based on this, certain algorithms like interpolation can be used and thus efficient control of pixelated light strips is achieved. This header byte is used only for virtual pixelation design as discussed in chapter 4.

- Number of bytes in the buffer - The total number of bytes in buffer (total payload before fragmentation) is given in the header of the Zigbee message. This helps during reassembly of fragments at the receiving end. For pixels consisting of color values of more than 255 bytes, 2 bytes of ZCL header is required to specify the number of bytes in the array buffer.

The above mentioned parameters are specified in the ZCL header of the implemented Zigbee design which sum up to 7 bytes. Hence, the actual Zigbee payload available to specify color values of individual pixels is 82-7=75 bytes which is discussed earlier in section 3.4.

5.1 Payload estimation for pixelated light strips

This section discusses the estimated number of bytes required to address all the LEDs individually in a pixelated light strip. According to the hardware design, described in section 4.2, a 1m length light strip contains 16 pixels. Based on the hardware design of pixelated light strips, certain estimates are made for a light strip of 1m and 4m lengths (minimum and maximum length assumed).

Estimates to control pixelated light strips			
Length of the light strip	Number of pixels	Number of bytes required	Number of Zigbee messages
1m	16	80 bytes	2
4m	64	320 bytes	5

Table 5.1: Payload estimations for 1m and 4m length pixelated light strips

$$Required\ number\ of\ bytes = Number\ of\ pixels \times 5 \tag{5.1}$$

$$Required\ number\ of\ Zigbee\ messages = \left\lceil \frac{Number\ of\ bytes}{75} \right\rceil \tag{5.2}$$

Equations 6.2 and 5.2 are used to calculate the number of bytes required and number of Zigbee messages respectively to control an N meter length light strip Currently, the Philips Hue strips in the market consist of 54 LEDs approximately on a 1m length light strip. Although the hardware design of the current Hue strips is different, if the designed Zigbee protocol extensions would be applied for the worst case scenario of 54 LEDs per 1m length light strip, then the estimated payloads are shown in Table 5.2.

Parameters	Estimates
Length of the light strip	1m
Number of Pixels	54
Total number of bytes required	270
Number of Zigbee messages	4

Table 5.2: Payload estimations for current Hue strips

According to the estimates shown in Table 8.1, the payload size required for controlling pixelated light strips will definitely be more than the payload size supported by the current Zigbee standards as discussed in the section 3.4. However the Silicon Labs EmberZNet Pro stack supports a Zigbee feature called fragmentation that allows to send long packets comprising of payloads more than 82 bytes (including the ZCL headers).

5.2 Fragmentation technique

In section 3.4 we have seen that a single manufacturer-specific cluster command can contain up to 75 bytes of data. Section 3.4 also shows that depending on the binary representation of colors, 3-5 bytes are needed per pixel. With respect to the estimates presented in section 5.1, it is clear that to address many pixels individually on the light strip, multiple Zigbee messages are needed. There are two ways to handle this problem. One approach is to use the Zigbee standardized mechanism to send larger messages, which is called fragmentation. The other approach is to define manufacturer specific custom commands in such a way that it can consist of multiple ZCL messages. Both these approaches are investigated in this section. Zigbee fragmentation is described in section 5.2.1

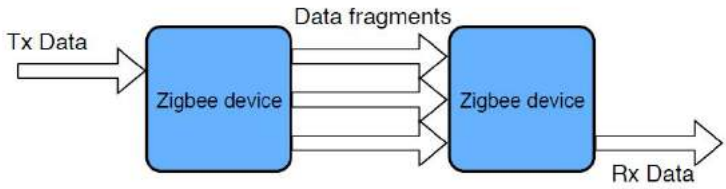

Figure 5.1: Fragmentation technique

The Silabs EmberZNet Pro stack provides various plugins that can extend certain functionalities or add new features to the stack. The Fragmentation plugin provides the ability to support fragmented transmissions. This Zigbee feature allows a single large data packet to be broken up into multiple unicast transmissions and reassembled by the receiver before sending data from its serial port as shown in Fig. 5.1. The EmberZNet framework and fragmentation library currently supports messages upto 255 bytes. The transmission can include up to 255 bytes of data broken up into multiple transmissions and reassembled at the receiving side. In order to use this feature for transmitting longer Zigbee packets, the plugin called "Fragmentation" must be enabled under the plugins tab of simplicity studio.

5.2.1 Stack fragmentation

As discussed earlier in the implementation details chapter, the main component involved in the project is EFR32MG12 Mighty gecko wireless SoC from Silicon labs. In this project, two of these hardware boards are used, one configured as a Zigbee light application and another configured as a Zigbee switch application. The switch microcontroller acts as the sender (sending large data packets) and the light microcontroller acts as the receiving end. In order to send large data packets over the air, the fragmentation plugin is enabled under the plugins tab of Simplicity Studio 4.0. The plugins tab in the AppBuilder has over 100 plugins where each enables one or more features on top of the application framework. After enabling the fragmentation plugin, the AppBuilder generates a software project with customized headers file and source code to represent the desired application behaviour. After generating, the resulting project can be loaded onto the target wireless SoC. The fragmentation plugin is an application level utility above the stack APIs. This plugin allows to set the maximum number of incoming and outgoing fragmented packets. Though the AppBuilder specifies that the maximum payload size is 74-10000 bytes (prior to fragmentation), the current framework is capable of handling only up to 255 bytes of payload prior to fragmentation. Once the fragmentation plugin is enabled, the stack has the functionality to fragment larger Zigbee packets (exceeding beyond 82 bytes) into smaller fragments. Each of these fragments can hold a payload of up to 82 bytes. Once all the fragmented packets are received at the receiving end, they are reassembled by the stack again into a large Zigbee packet. One advantage of stack fragmentation is that the stack takes care of both fragmentation and reassembly by itself. However, the drawback is that the current framework can handle only up to 255 bytes. When it was tried to send packets containing more than 255 bytes through the Zigbee command, the stack returned an error saying that the message is too long and thus the transmission failed. Stack fragmentation works only for unicast transmissions and not for multicast. The advantage of using stack fragmentation is that all the necessary work is carried out by the Zigbee stack itself, right from fragmenting the longer Zigbee packets to reassembling the fragmented packets. This design was investigated and prototyped but was not a part of the final implementation, because it was unclear in which order the message fragments where received since there was no clear sequence number for the message fragments.

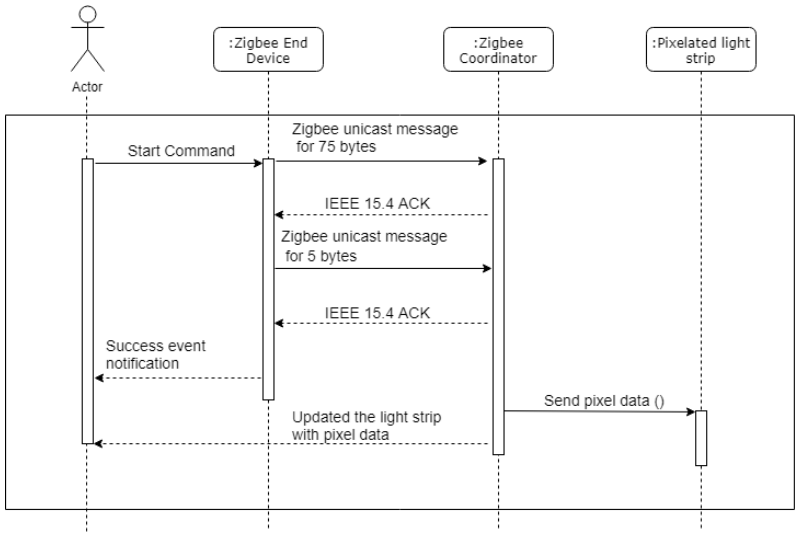

Figure 5.2: Sequence diagram for single update of light strip

5.2.2 Handling data subsets

For Zigbee messages comprising of payloads of more than 255 bytes, subsets of data arrays with color values are trasnmitted through multiple ZCL messages. This design is quite similar to stack fragmentation except for the fact that Fragmentation plugin is not used. Hence, longer Zigbee packets with payload more than 255 bytes can also be fragmentated. In this case, the source code that handles fragmentation of longer Zigbee packets is deployed at the switch microcontroller end. Each subset of the data array may consist up to 82 bytes. In the current implementation, each fragment has 75 bytes of payload and 7 bytes of header. Unlike the stack fragmentation, fragmentation reassembly is not handled by the stack. Based on this concept, two types of Zigbee protocol extensions are investigated in this section.

Single update of light strip

In the hardware set-up described in section 6.3, the ZED transmits the necessary color values to the ZC for the purpose of controlling each pixel of the light strip. The large data array with color values hard-coded at the ZED is transmitted as subsets of the array with payload size 82 bytes (75 bytes of data payload + 7 bytes of ZCL header) to the ZC. Once all the subsets of the data array are received at the ZC, the light strip is updated with the color values by the ZC. This Zigbee protocol design consists of two design steps,

- Design of manufacturer specific custom commands in such a way that it can send subsets of data array through multiple ZCL messages at the ZED.

- Reassembly of the subsets of the data array according to the sequence number specified in the ZCL header at the Zigbee Coordinator device.

Once the reassembly of subsets of data arrays is done at the ZC device, the light strip is updated with the color values specified for each individual pixel through the SPI bus. This design is used for the final implementation to control pixelated light strips since the total delay produced on the light strip is less than 200 ms which is not visible for the users. Fig. 5.2 shows the sequence of events for a successful control of the light strip with this design. This design was investigated on a light strip with 18 pixels which requires 90 bytes to control the individual LEDs and implemented as well.

Intermittent update of light strip

In contrast to the design described in section 5.2.2, this Zigbee protocol handles intermittent updates of the light strips with subsets of data arrays. This Zigbee protocol design consists of two design steps as described below. The first step is similar to the previous design described in section 5.2.2.

- Design of manufacturer specific custom commands in such a way that it can send subsets of a data array through multiple ZCL messages at the ZED.
- Updating the light strips with subsets of the data array at a given instant at the receiving end (at the ZC).

Though this design was investigated and prototyped, it was not a part of the final implementation to control the light strips because of the visible delay that could be seen on the light strip. This delay might be because each subset of array must be processed by various functions defined at the application layer level before being sent to the light strip. For instance, the xyb to RGB conversion algorithm might take several milliseconds before processing the next subset of the data array.

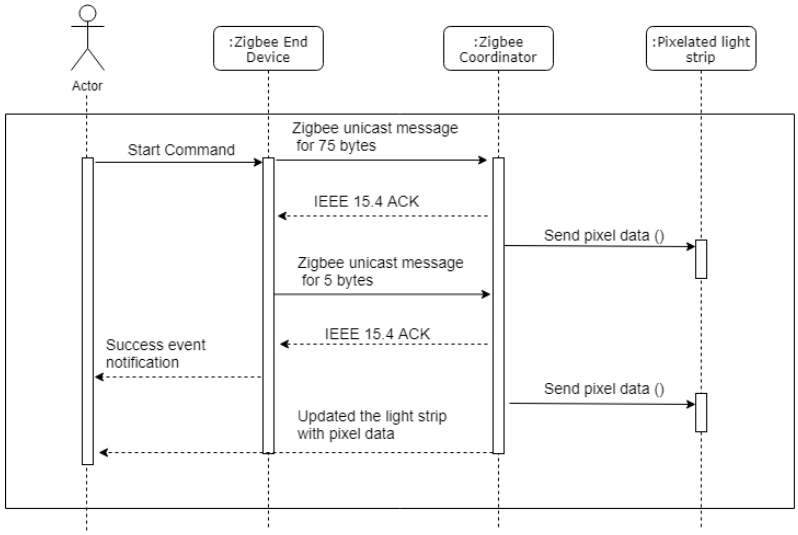

Figure 5.3: Sequence diagram for intermittent updation of light strip

Fig. 5.3 shows the sequence of events for a successful control of the light strip with the intermittent update design. To design Zigbee protocol extensions, it was found that new clusters to perform pixelated light strips are essential. Section 5.2.3 briefly discusses creation of custom clusters to control pixelated light strips.

5.2.3 Creation of custom clusters

Zigbee allows extending clusters with manufacturer commands, or even add manufacturer-specific clusters to an endpoint. When our own clusters are created, it is necessary to identify such clusters as manufacturer specific clusters. Manufacturer-specific clusters can be added to a standard profile. A new cluster XML was created with this necessary set of message types and added to the App builder in Simplicity studio. To create custom clusters, the following two obligations must be satisfied.

- The cluster ID MUST be in the manufacturer-specific range, 0xFC00 - 0xFFFF.

- The cluster definition must include a manufacturer code which is applied to all attributes and commands within that cluster and must be provided when sending and receiving commands and interacting with attributes.

The cluster id for "Pixelated cluster" was chosen to be 0XFC03 and the manufacturer code is 0x100B for manufacturer Signify.

5.2.4 Creation of custom commands and attributes

Similar to the custom clusters, custom commands and attributes can also be defined to cater for the needs of the developed application. The obligations that must be satisfied to create custom commands and attributes are as follows,

- Manufacturer-specific commands must use any command id within the command id range, 0x00 - 0xFF.
- Manufacturer-specific attributes must use any attribute id within the attribute id range, 0x0000 - 0xFFFF.
- Both custom clusters and attributes must provide a manufacturer code so that it can be distinguished from other commands and attributes in the cluster and handled appropriately.

Table 5.3 shows some of the custom commands and attributes created in the pixelated cluster.

Custom Commands & Attributes				
Command Name	Command ID	Attribute name	Attribute ID	Manufacturer code
pixel off	0x00	pixelated on off	0x0000	0x1002
pixel on	0x01	pixel current level	0x0003	0x1002
pixel dynamics on	0x02	pixel on level	0x0011	0x1002

Table 5.3: Examples of custom commands & attributes

Chapter 6

Performance Analysis

This chapter presents the results of the performance analysis which was done in order to support the design of extensions to the Zigbee protocol. The results are used to decide on the best fit protocol for the defined use cases for pixelated light strips as discussed earlier in the chapter 2. It includes testing various (unicast) Zigbee packet transmission scenarios, a memory consumption analysis, and CPU load measurements and calculations for the SiLabs ARM processor discussed in sections 6.1, 6.4 and 6.5 respectively . In order to do performance analysis of the above-mentioned metrics, different types of experiments were conducted on Zigbee nodes and the results are summarized in this chapter.

6.1 Zigbee unicast transmission analysis

This section discusses the Zigbee unicast performance evaluation of the extensions of the Zigbee standards, as discussed in chapter 5. In order to control the individual pixels of the light strips, large data arrays with color values must be sent to the light strips. These large data arrays with color values are sent as unicast messages from the Zigbee device to the light strips. Fig. 6.1 depicts the one-way data transmission in the stack between two Zigbee nodes.

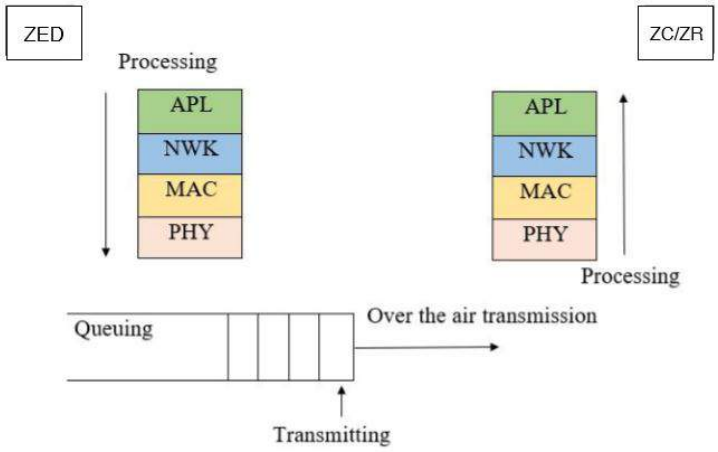

Figure 6.1: One-way packet transmission
[16]

Here, the data that are transmitted are the large color arrays that are fragmented as subsets

of color arrays. They are generated at the APL layer which progresses all the way down to the physical layer and is transmitted over the air to the physical layer of the other Zigbee node. The remaining subsets of the color arrays are queued up at the transmitting end and are transmitted after the success of the preceeding packet. If there is a delay in updating the light strip with certain subset of color arrays, or if a subset of color array is lost in-between, then there is a flicker in the light strip which is clearly visible to the users. Therefore, we want to understand the impact of using large Zigbee messages on packet loss and latency. One of the metrics to decide on the best design implementation for the defined use cases in chapter 2 is Packet loss during Zigbee packet transmission. Another is the evaluation of latencies by measuring the round trip time. The Zigbee protocol relies on the underlying IEEE 802.15.4 Physical (PHY) and MAC layers for data transmission. The MAC layer uses CSMA/CA just like WLANs to avoid any collisions during data transmission.

6.1.1 Test Environment

Packet loss in a Zigbee network is mainly due to interference from Wifi networks, Bluetooth and other Zigbee networks. Previous work carried out by Kui Liu in "Performance Evaluation of ZigBee Network for Embedded Electricity Meters" in 2009 confirms this[16]. This paper focuses on performance evaluation of Zigbee networks based on performance metrics such as throughput, latency, security, network size etc. They performed several experiments both indoors and outdoor to measure packet loss for varying distances between Zigbee nodes.

Distance	Average RTT	Packet loss rate
20m	18.0ms	0%
50m	18.1ms	0%
70m	18.4ms	0.1%
90m	18.9ms	0.05%
105m	18.0ms	0.05%
115m	18.2ms	0.1%

Figure 6.2: Packet loss % in low traffic zone
[16]

Fig. 6.2 is taken from [16]. It shows a series of tests performed in a company environment within line of sight. It was presumed that there was no interference from obstacles. The experiments were repeated for varying distances from 10m to 115m between the Zigbee nodes however, for a fixed packet length of 50 bytes. It was inferred that even for very long distances of more than 100m, there is a negligible packet loss of less than 1%. Similar experiments have been carried out with different doors in between and on different floors. The results were compared to see how the transmission condition is affected by different obstacles (high interference condition). It was noticed that the packet loss rate was around 3.55% which is comparatively higher than in the low traffic zone.

In order to study the impact of packet length on packet loss, we decided to conduct our experiments in two type of environments and for varying distances between the Zigbee devices. The two types of environments are,

- High Traffic Zone
- Typical Traffic Zone.

High Traffic Zone

The experiments were conducted in a research environment in Signify where the office area is occupied by different research teams working on wide range of wireless devices. Next to an abundance of Wifi signals, a very large number of Zigbee networks and Bluetooth devices are present. A high traffic density was confirmed in this area due to the wireless activity in the

background from devices such as Zigbee, Bluetooth and Wifi access points. This environment is treated as a worst case scenario. Numbers on packet loss and latency will certainly be much higher than in the typical case. Table 6.1 shows the test conditions in high traffic area.

Parameter	Test Conditions
Environment	2nd floor,Signify HTC 7,Eindhoven
Temperature	25C
Humidity	38 %

Table 6.1: Test Conditions in high traffic zone

Typical traffic zone The experiments were conducted in a home environment with less wireless activity in the background. In a low traffic zone like the home environment, the number of devices present in the environment was comparatively less than a high traffic zone. There was little interference due to Wifi routers and microwave ovens. The test conditions in the typical traffic zone are shown in 6.2

Parameter	Test Conditions
Environment	In an apartment, Beukenlaan 5616VC,Eindhoven
Temperature	34C
Humidity	36 %

Table 6.2: Test Conditions in low traffic zone

For the low traffic zone, we decided to perform tests at home. In the home environment we expect interference from the home Wifi network, Neighbors' Wifi networks, Bluetooth devices and Zigbee networks other than the test network. As this situation is typical for our application, it is an important environment to test. Although results could be influenced by external factors, e.g. a neighbor starting video streaming via Wifi, repeating some of the measurements did not show significant differences between measurements.

Instead of a Low traffic zone, it would be possible to perform measurements in a shielded room. However, we assumed that this will not produce new insights related to our use cases.

6.1.2 Test Setup

In order to perform the experiments explained in section 6.1.1, a simple experimental set-up as shown in 6.3 was used for determining packet loss during data transmission in a centralised network. In this experimental setup, longer Zigbee packets are transmitted from the ZED which is the switch microcontroller, to the light microcontroller configured as a ZC/ZR. Since the final implementation was done on a light strip with 18 pixels, a long Zigbee message with 90 bytes (calculated from the equation 6.2 was transmitted by the ZED. For this purpose, the Mighty Gecko wireless SoCs from Silabs described in 4.1 were used. The test software was developed and deployed at the application layer of these ARM processors. The packet to be transmitted is generated in the application layer of the transmitting node and sent to the lower layers of the Zigbee stack. The experiment involves only two Zigbee devices in a simple centralized network. This is because, in such a scenario, there is direct transmission, the ZED sends data packets directly to the coordinator. In case of large networks with a multi-hop configuration through routers, when the router re-transmits its packets to the coordinator, the medium is occupied; and, the ZED must wait before transmitting the next data packet. It was decided to have a simple

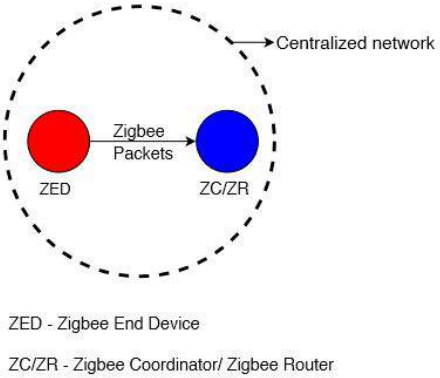

Figure 6.3: A simple representation of the experimental setup

setup with two Zigbee nodes and not to measure other influencing factors like number of hops, routing time etc.

6.1.3 Number of tests

The experiment was repeated for 200 test cases i.e. 200 Zigbee unicast transmissions were sent between the two Zigbee nodes. Performing 200 test cases was considered a sufficient number needed for analysis. The number of test cases could have been restricted to 50 or 100. However, these tests did not consume much time and it was decided to repeat all the measurements 200 times. The test cases were repeated at various times during the day and since it was noticed that there was not much difference recorded between different iterations, they were not consistently repeated. It was also noticed that the measurements were independent. If there was a re-transmission of a Zigbee packet during one of the test cases, it did not have an influence on the transmission of the next Zigbee packet.

6.2 Packet loss related to packet length

The main goal of estimating packet loss between sender and receiver nodes is to check the reliability of the Zigbee protocol and to understand if increasing payload sizes has an impact on packet loss during transmission. Fig. 6.4 describes the sequence of events for a successful transmission of a unicast message. In this figure, it is implicit that when the developer starts the process of packet transmission, the ZED stack initiates the packet transmission after sensing the medium to be idle. The switch device has to send an 15.4 ACK packet to the ZED node indicating the successful reception of the unicast packet. After the reception of the ACK packet, the ZED stack notifies successful transmission of the packet to the developer by displaying a success message in a PuTTy console. If the ACK is not received by the ZED node, then it displays an appropriate error message to the user. Equation 6.1 gives the packet loss percentage at a given node,

$$packetloss\% = \frac{Number\ of\ packets\ lost}{Number\ of\ packets\ sent} \times 100 \tag{6.1}$$

Tests and Result Analysis

Test scenarios were designed to determine packet loss during data transmission between two nodes. Since many wireless systems use the frequency spectrum of 2.4 GHz, there could be interference which is a major cause of packet loss. Mainly in commercial lighting scenarios, there can be complex interference due to the presence of many active radio devices like other Zigbee

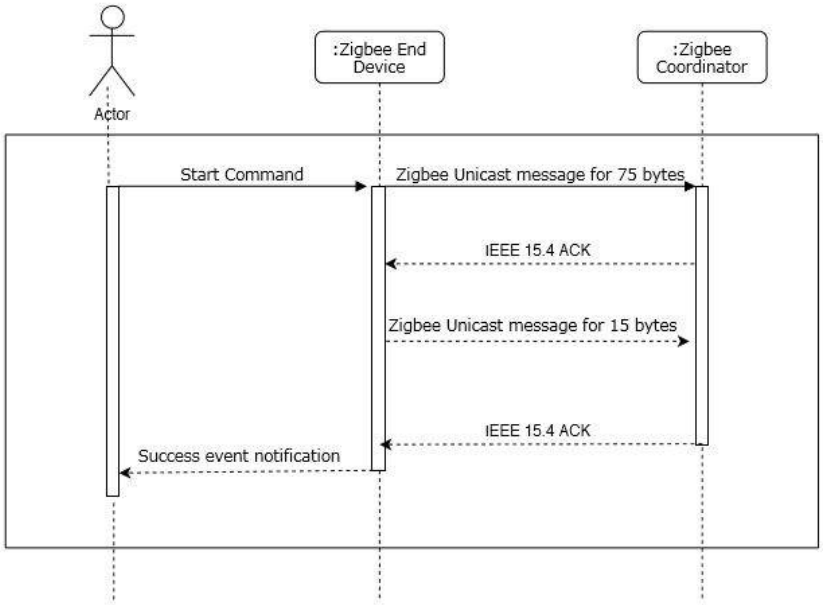

Figure 6.4: Sequence diagram for successful unicast transmission of a long Zigbee packet

networks, Wifi and Bluetooth devices. The main goal of these test scenarios is to understand if the background traffic has an impact on packet loss. The experiments were conducted in two types of environments as described in section 6.1.1.

To analyze the impact of message length on packet loss, messages were sent using payload sizes of 15 bytes, 75 and 90 bytes. The maximum payload size is chosen to be 90 bytes because the experiment was done on a light strip with 18 pixels which requires 90 bytes (calculated from equation 6.2) to control individual pixels. The test was repeated for varying distances of 1m, 5m and 10m between the two Zigbee nodes to see the impact of increasing distances between the transmitter and the receiver. The test settings are shown in Table 6.3.

Parameter	Test Settings
Distance between nodes	Varying distances of 1m, 5m, 10m between the nodes (point to point communication)
Payload length	Varying payload sizes (15, 75 and 90 bytes)
Interval between packets	0ms
Number of test cases	200
APS Security	Disabled
APS retry	Disabled

Table 6.3: Test Settings for packet transmission

The test was repeated for 200 test cases as discussed in section 6.1.3. The results that are obtained from the experiments, carried out with respect to the parameters in table 6.3 are depicted in the graph as shown in Fig.6.5. Fig. 6.5 shows the performance of Zigbee unicast transmissions with respect to packet loss in the high traffic environment. It can be seen from the graph that for packets with payloads of 15 and 75 bytes, all the 200 Zigbee unicast messages are received. Substituting the number of sent and received packets in equation 6.1, it is clear that payloads 15 and 75 bytes have 0% packet loss. When the distance between the two Zigbee nodes increases (to 5m and 10m), it was recorded that the longer Zigbee packet (90 bytes) has a packet loss of

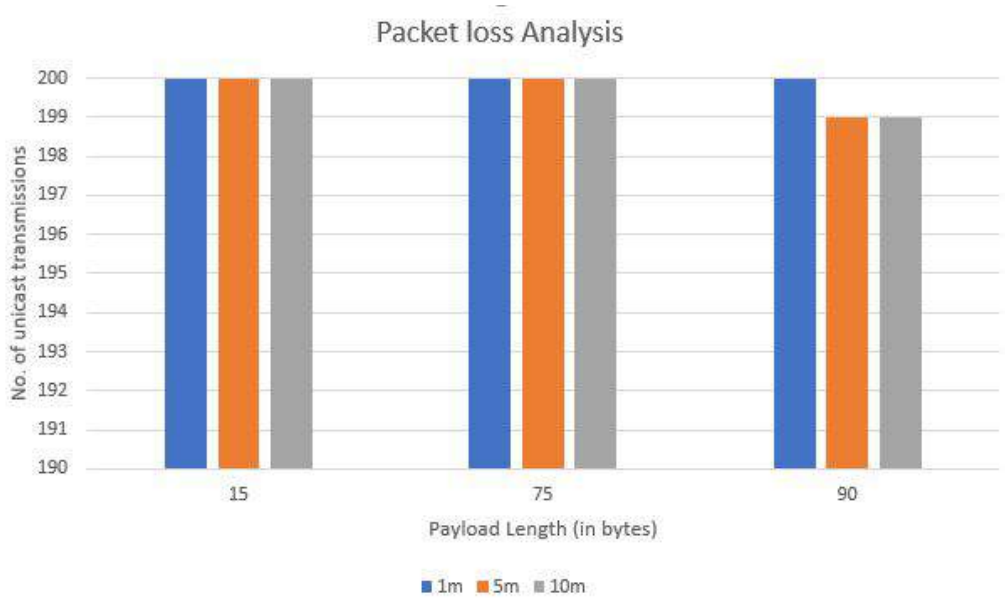

Figure 6.5: Unicast transmission performance evaluation w.r.t packet loss in high traffic zone

0.5%. The distance between the nodes can be an influencing factor on the packet loss percentage. This is because, as the distance increases, the interference due to other devices increases and and the received signal strength decreases. Although the measurements seem to suggest that for longer packets, the package loss is higher. However, the tests are insufficient provide a predictable relationship. However, since packet loss was only 0.5 percent in a very harsh environment we conclude that using long packets does not cause reliability concerns.

The experiment was again repeated for 200 unicast transmissions in a home environment (typical environment) which has less interference compared to the high-traffic zone. Fig. 6.6 shows that the Zigbee packet lengths of 15 and 75 bytes receives all the 200 transmitted messages. Packet loss% is calculated from equation 6.1 and it is found that we have a packet loss of 0.5% only for a distance of 10m between the Zigbee nodes [payload size: 90 bytes]. The packet loss might be due to the interference from a Wifi router at home or in the neighbouring homes or from bluetooth devices.

The experiments conducted in both the environments show that there is negligible packet loss. The packet loss is mostly 0% for both the minimum payload size of 15 bytes and for the maximum payload of 90 bytes though the distance between Zigbee nodes has an influence on the packet loss percentage. This confirms that Zigbee is a highly reliable and robust networking protocol to send longer data packets in order to control the individual LEDs on a light strip. Hence, Zigbee commands designed to send longer data packets as described in 5.2.2 can be used to control the pixelated light strips.

The reason that there is so little packet loss is because of packet re-transmissions. Zigbee packets are re-transmitted at various layers of the 802.15.4 stack before they actually are lost. However, as the number of re-transmissions of Zigbee packet increases, the time taken to successfully transmit a packet also increases. We assumed that longer Zigbee packets are more likely to undergo a number of re-transmissions before getting transmitted successfully. Therefore, we wanted to determine if a relationship between message length and increased latency due to packet retransmissions can be found.

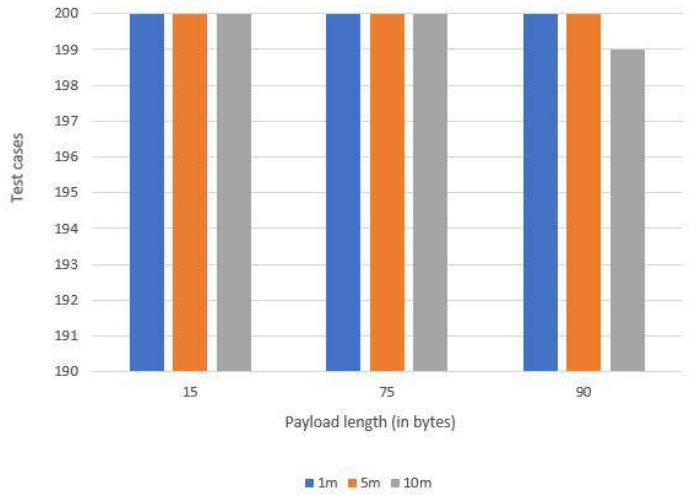

Figure 6.6: Unicast transmission performance evaluation w.r.t packet loss in typical traffic zone

6.3 Impact of packet length on latency

Latency is the other performance metric that is discussed in this paper. To determined the latency, we decided to measure the RTT of a message, instead of amount of time for the packet to reach the destination node. RTT is the time calculated from when the packet is generated in the application layer to the time the application layer receives the response from the target node

Prior to packet transmission, the sending node senses if the shared medium in a wireless network is busy. If the medium is busy, it waits for a certain period of time (backoff period) and starts transmitting when the medium is idle. After sensing the medium to be idle, the ZED initiates data transmission and receives an acknowledgement once the transmission is successful. This is calculated as the RTT. Here the ACK is sent and processed by MAC layer so the application layer wont be aware of it. Fig. 6.7 shows the round trip time for transmitting one Zigbee packet.

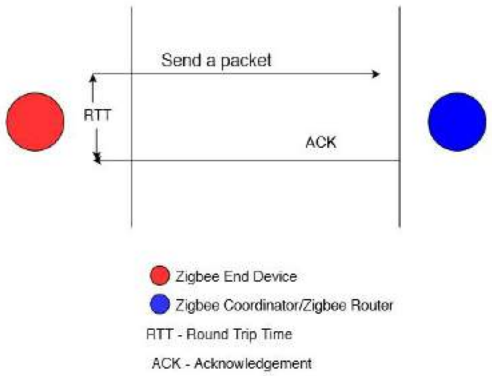

Figure 6.7: Round Trip Time of a packet

When a packet is lost during transmission, re-transmissions also contributes to the RTT/latency. Re-transmission is dealt within the APS layer, the NWK layer and the MAC layer. The APS layer sends the packet to the Network (NWK) layer. NWK tells MAC layer to transmit and starts a timer before sending any packet. The MAC layer waits for a random period (backoff period), transmits the packet and waits for an ACK. If the ACK is received, the MAC reports success to the NWK layer. If the ACK is not received, the MAC retries the entire CSMA transmit process

up to five times. If the ACK is not received in any of the attempts the MAC reports failure to the NWK layer. Once the MAC layer retries are exhausted for that packet and the packet is still not transmitted, the packet actually is lost. If APS retry is enabled (software option), the APS layer will attempt to resend the packet several times(configurable in the Silabs software, default value 3) before receiving ACK. However, in this implementation, the APS retry mechanism was disabled. MAC-level acknowledgements and retries were kept as default and automatic at the Silabs software. MAC layer retries 5 times(configurable) before returning a failure status to the network layer. The time interval between each attempt depends on the CSMA/CA algorithm. Latency can change with varying packet lengths and distances between Zigbee nodes. The previous work by E. Dalila Pinedo-Frausto and J. Antonio Garcia-Macias on "An experimental analysis of Zigbee networks" focusses on the Zigbee network analysis based on various factors among which Zigbee networks were evaluated based on latency as well[17]. This paper confirms that the latency increases with increasing distances between the Zigbee nodes. The experiment was carried out in a parking lot with little interference from other devices and for varying distances from 20m to 75m. It was recorded that a delay of 46ms was noticed for 20m distance while a delay of 49ms was noticed for 75m distance between the nodes[17]. This gives an insight that distance between Zigbee nodes can be an interesting factor influencing the number of re-transmissions, which in turn has an impact on latency.

Tests and result analysis

For critical industrial applications latency is one of the most important characteristics of a sensor network. We tested 2 important factors that contribute to latency: distance and packet length. Experiments were carried out with the test setup as described in section 6.3 both in the high traffic zone and the typical traffic environment. The experiment was repeated for 200 unicast transmissions as discussed earlier in section 6.1.3 for varying packet lengths of 15, 75 and 90 bytes and for varying distances of 1m, 5m and 10m between the Zigbee devices. However, the results for a payload size of 90 bytes is presented in this paper. This is because the packet length of 90 bytes is sent as a subset of 15 bytes and 75 bytes as discussed in 5.2.2. To measure the RTTs for each test case, the Network analyzer provided by Simplicity Studio was used.

Fig. 6.8 shows the scatter plot of RTT for a 10m distance between the Zigbee nodes, for payload size of 90 bytes in the high traffic zone. It can be seen that a maximum of 98ms RTT has been recorded in one of the test cases. During this test case, the packet has undergone the maximum of 5 re-transmissions at the MAC layer. This proves that in high traffic zones the Zigbee messages can undergo the maximum number of re-transmissions before it gets transmitted successfully to the destination node. Retries at other layers are not discussed here because the packet got transmitted successfully with the MAC retries itself and hence, retry at the NWK layer was not required. APS retry was disabled in this experiment.

The same experiment was repeated for 200 test cases in the low traffic zone with less background interference for a packet length of 90 bytes for varying distances of 1m, 5m and 10m. Fig.6.9 shows the scatter plot of RTT for 90 bytes of payload repeated for 200 test cases in a less traffic or home environment. The distance between Zigbee devices is 10m in this case. It can be seen that the packet length of 15 bytes always has an RTT of 3ms whereas the packet length of 75 bytes has a round trip time of 5ms. Plotting RTT for the two packet lengths in the same graph shows a combination of 3ms and 5ms round trip time for each test case. It can be noticed that the measurements are independent of each other. If a packet length of 15 bytes gets re-transmitted, then the RTT for that specific packet increases and does not have an impact on the next Zigbee packet.

Fig.6.10 shows a histogram for the RTT of 90 bytes packets. It can be seen from the histogram that most of the 15 bytes Zigbee packets were transmitted successfully without re-transmission. Packets of size 75 bytes undergo a higher number of re-transmissions compared to Zigbee packets of 15 bytes. This confirms that a bigger size of the payload results in a higher latency.

The distance between Zigbee nodes is also another factor that influences latency. Experiments were done for 200 test cases for a payload size of 90 bytes. Fig.6.11 shows the latencies in ms,

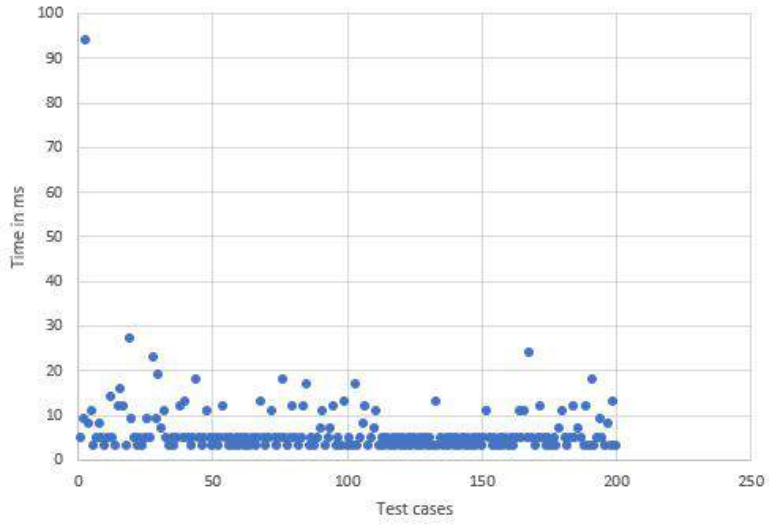

Figure 6.8: RTT for 90 bytes payload in HIGH-traffic zone

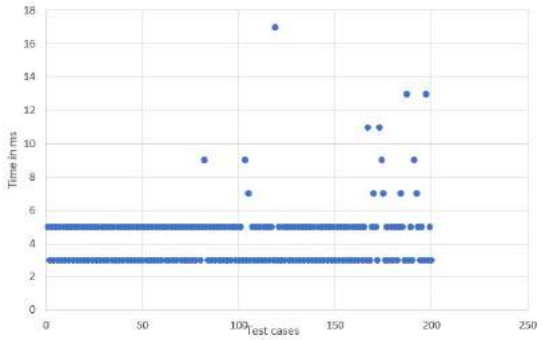

Figure 6.9: RTT for 90 bytes payload in less-traffic zone

accumulated for all the test cases for varying distances of 1m, 5m and 10m between the Zigbee devices. The figure shows the impact of varying distances in both high traffic and less traffic zones. For the distance of 10m the cumulative latency over the 200 test series recorded is 1300 ms.

From this section it can be inferred that, when longer Zigbee packets are sent to the light strips, then there is small possibility of delay caused by the retransmissions of the long Zigbee packets. If the delay is more than 200ms, the delay is noticeable by the user.

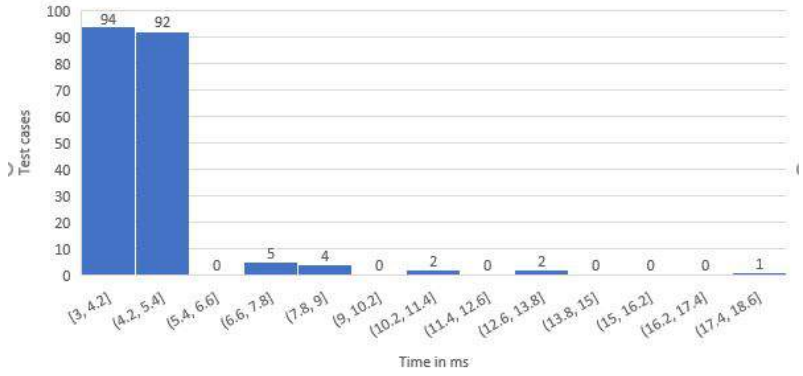

Figure 6.10: Histogram showing RTT for 90 bytes packet

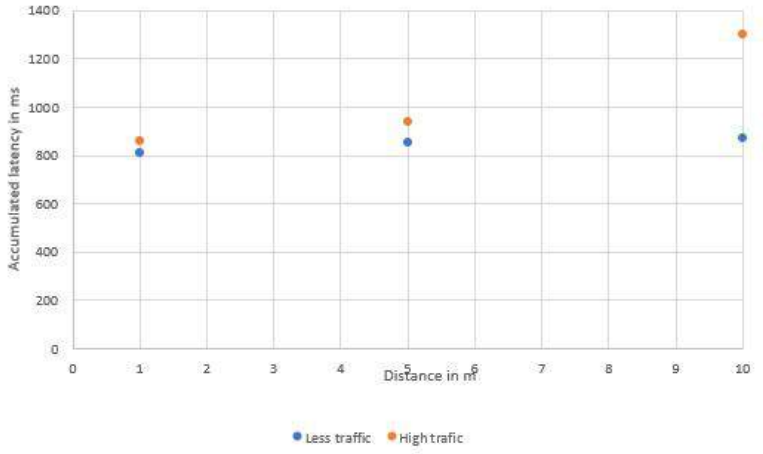

Figure 6.11: Maximum accumulated latency vs distance

6.4 Memory consumption analysis

For our design, it is important to calculate the hardware memory required to control the individual pixels on the pixelated light strip. The hardware used in the implementation is the EFR32MG12 Mighty Gecko Wireless SoCs from Silabs as discussed in section 4.1. These Mighty Gecko SoCs have a Programmable Flash memory of up to 1024 bytes and 256 Kb RAM. The memory requirement can vary for different types of design options as described in chapter 5. Only the data memory required for the implementation is calculated in this work. Memory consumed to control a light strip containing 18 pixels was calculated in this work. As discussed in section 4.2, each driver chip integrated on the light strip drives 2 pixels. To drive the PWM signals per chip, (226/8) bytes is required. Below equation was used to calculate the required memory.

$$Required\ memory\ in\ bytes = \left\lceil \frac{Number\ of\ pixels \times 5 \times 226}{8 \times 2} \right\rceil \tag{6.2}$$

Hence for a light strip containing 18 pixels requires will require 1.2 Kb RAM. For a worst case scenario of 4m light strip containing 64 pixels, it was estimated that approximately 4.5 Kb of RAM would be required.The Silabs chip has 256 Kb RAM, hence there will be sufficient memory to control the individual pixels of a light strip.

Implementation of Scenes can also have an impact on the memory requirements. A scene is a set of stored attribute values for one or more cluster instances, where these cluster instances

may exist on endpoints on one or more devices. To create a scene, a scene table must be created where an entry for the scene must be added to the Scene table on every device that contains a cluster which is associated with the scene. In Philips Hue, we can create up to 16 scenes. All the scene entries are usually stored in the RAM. To implement scenes. For instance, a user might create set a nice ambience using Hue lamps and store it as a scene in the Hue app. This scene gets stored in the RAM and when recalled by the user, it gets retrieved from the SoC's RAM. Hence to implement scenes more RAM would be required and the memory estimation for scene implementation is not carried out in this work.

6.5 CPU load analysis

For our design it is important to analyse whether a typical SoC (System on a chip) for Zigbee applications is capable of processing large color arrays. The hardware used in the implementation is the EFR32MG12 Mighty Gecko Wireless SoCs from Silabs as discussed in section 4.1. This board has a 32-bit ARM processor with 40 MHz clock speed and 256 KB RAM. The processing speed can vary for different types of tasks. For instance, computing an effective address within the array may happen quickly, in a single clock cycle, and whereas reading a value from the array often takes multiple clock cycles, and is influenced by the prefetcher and state of each cache level. Additionally, the EFR32MG12 has a hardware floating point unit running at 40 MHz. The floating point unit can perform multiplications and additions in a single clock cycle. Since a complete CPU load analysis with a profiling tool would require implementation of the complete application, we decided to estimate CPU performance based on calculations.

Our main concern for a high CPU load was the color processing of each pixel. Usually, Philips Hue uses the xyb color model to control the lights. In case of pixelated strips , we define xyb values to address each pixel on the LED strip. The LED strips produce coloured light by mixing Red, Green and Blue leds. Therefore, the xyb values need to be converted to RGB values before sending it to the light strips. From the implementation of the color processing for a single LED, we know that it takes 200 microseconds on the EFR32MG12. Let's consider a 4m length light strip with 64 pixels. In this case, the processing time will be around 13ms (64*200). This means that after receiving the Zigbee message, the CPU needs 13ms to convert the colors, and then send the RGB values to the LED strip hardware. This means to show correct the light output, 13ms needs to be added to the latency measurements described above. While this is not a negligible amount, picking a color from a mobile application and sending it to the light strip is still fast enough.

For use cases which require a fast update of the light strip, such as reacting to music or games the situation is different. If we assume we want to sent a new set of colors 25 times per second to create smooth transitions (because in Signify, the refresh rate usually lies between 25-30ms), it means that every second the CPU needs to spend 25*13ms on color processing. The total processing time will amount to 325ms. This means that the CPU needs to spend a third of its time on color processing. Since one-third of the CPU time (say around 35%) is spent on color processing, around 65% of the CPU time is available for other stack operations. At this moment it is not clear whether 65% is sufficient for all other tasks. It may require some further optimisation, lowering the update frequency, reducing the number of colors, or optimising the color computations.

Chapter 7

Flutter Implementation

To control the pixelated light strip through a mobile app, a user interface was implemented using Flutter. This chapter discusses the features of the Flutter app and how they are implemented to control individually addressable LED strip. Section 7.1 discusses the cross platform application development and the framework used in this work to implement a hybrid mobile app and section 7.2 discusses the features developed in the application to control the pixelated light strips.

7.1 Cross platform mobile applications

Nowadays, cross platform application development has gained a great potential. Although native platform application development has its own pros, it is quite a lot of work to code the application for both Android and IOS platforms and deploy them. When a development change is required in the app, it has to be done separately in both Android and IOS platforms for the change to reflect. This is where the cross platform application development comes into play where a common source code is deployed for various platforms. There are various tools for development of such apps like Xamarin, PhoneGap, Appcelerator etc. among which Google's UI framework, Flutter, is used in this project to create a user interface. Flutter apps are written using the DART language. The Flutter application size has been reduced as compared to native platform applications. This will improve the application performance and with this, Flutter has become a suitable platform for developing heavy and complex apps.

7.2 Implementation

A basic user interface was implemented using Flutter to control the pixelated light strips. The app has 4 main features. The first feature is, it scans for the device when launched. The home screen consists of 2 features such as turning the pixelated strip on and off and controlling the brightness of all the pixels on the light strip asshown in fig. 7.2. The app has a separate tab called "Pixelated" which consists of a grid view of images as shown in the fig. 7.3. When an image is selected by the user, 3 dominant colors are picked from the image and is sent to the light strip. Fig 7.1 shows the control of pixelated light strip containing 18 pixels using a Flutter app.

Figure 7.1: Control of a pixelated light strip using a Flutter application

Figure 7.2: Home screen in Flutter app

Figure 7.3: Grid view of scenes

Chapter 8

Related work/Literature study

The thesis aims to design Zigbee protocol extensions to control pixelated light strips. This section discusses the work related to the thesis. To do this, the literature survey was split in three parts i.e. similar works carried out to control individual pixels of light strips, work related to the ZLL standard, and similar work on some of the performance metrics used to evaluate performance of Zigbee networks.

Related work on Pixelated light strips

The goal of this project work is to control/address each of the LEDs on the light strip individually. When a careful study was done on similar work on LED strips it was seen that similar work has been carried out by ENTTEC, an Australian company in the design and manufacture of LED lights and controls. The pixel strip from ENTTEC has 12V RGB LEDs, each having 60 pixels per meter[18]. The pixel strip from ENTTEC also has three LEDs grouped together to form one pixel similar to our hardware design described in section 4.2. However, these light strips can be used only indoors unlike the Philips Hue strips. The drawbacks of ENTTEC's pixel strip is that it can work only with an ENTECC pixie driver. This pixie driver uses LED protocols such as WS2811, WS2812, WS2812b[19] etc. in which, when one of the LED is broken, the remaining LEDs are more likely to fail. For this purpose, the pixie driver is now upgraded to WS2813, but it is still not efficient when compared to a wireless communication protocol like Zigbee. They cannot be controlled wirelessly like our pixelated light strips and the strip is very costly. The ENTTEC pixel strip along with the driver can cost upto 400$ AUD. Fig. 8.1 shows the individually addressable pixel strip from ENTTEC.

Figure 8.1: Pixel strip from ENTTEC
[19]

Another similar product is Adafruit NeoPixel Digital RGB LED Strip. The Adafruit light strip has 60 individually addressable LEDs per meter[20] similar to the ENTTEC pixel strip described above. It comes in varying lengths starting from 1m to 4m and is flexible. The color of each LED's red, green and blue component can be set with 8-bit PWM precision (24-bit color per pixel)[20] which is quite similar to our pixelated case strip under study. The LEDs are controlled by shift registers that are chained up down the strip. But these light strips can be controlled only with a protocol which is highly timing-specific and can only be controlled by microcontrollers with highly repeatable 100ns timing precision[20]. Arduino Uno/Mega microcontrollers at 8MHz and

16MHz are used for this purpose[20]. Though this light strip has some similarity to the pixelated light strips, they are inefficient when compared to the pixelated light strips since they don't use a wireless protocol like Zigbee which is highly reliable as discussed in section 3.1

Comparison of pixel strips			
Parameters	Signify's Pixelated light strip	ENTTEC Pixel strip	Adafruit's Pixel strip
Hardware	EFR32MG12 wireless SoCs from Silabs	WS2812B chips and a pixie driver	Arduino Uno
Connectivity type	Wireless	Wired	Wired
Number of pixels per meter	16	60	60
Power consumption per meter	20 watts	10.8 atts	18 watts
Lumen capacity	1600 for 2m	291 lumens per meter	Not clearly specified
Cost	150 euros for 2m	400$	50 euros for 2m
Usage	Both indoors and outdoors	Only indoors	Both inddors and outdoors

Table 8.1: Comparison between different types of pixel strips

Related work on ZLL applications

In this thesis, a wireless control of pixelated light strips in the home environment has been implemented. This requires the use of ZLL standard which is endorsed for use in the residential lighting by the Connected Lighting Alliance, which is an industry consortium of leading lighting companies. Zigbee is the standard light choice for wireless network connectivity of smart lighting devices and used in products such as Philips Hue and OSRAM lightify smart lights. The paper "Zigbee Light Link standard and its applications" by Jiafeng Wang[3] proposes to use the ZLL mechanism for lighting applications which is currently followed by companies like Philips. With the help of ZLL it is possible to connect and control home lighting from the internet. All that is required are the ZLL light devices that need to be controlled, a router,a ZLL control bridge and a control device[3]. The ZLL control bridge and the the device with the app installed must be connected to a router which should have an internet connection. There can also be other switch devices (connected to the router) with which the ZLL devices can be controlled. The ZLL standard not only allows to control the on/off (start) of the smart lights but also to control the color temperature, hue, saturation and brightness.The control devices usually are smartphones, tablets or PCs. A control bridge is required to access the internet.

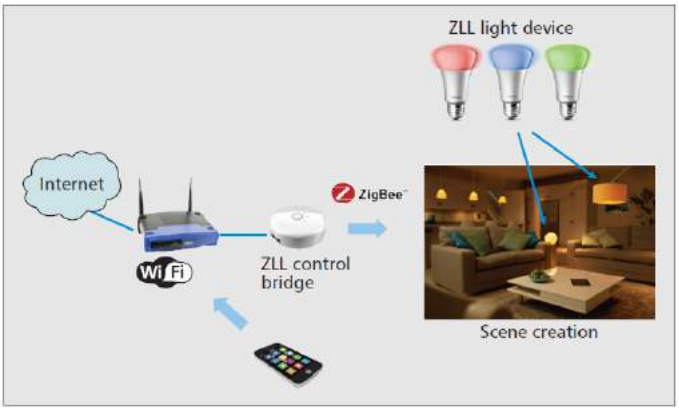

Figure 8.2: ZLL Application
[3]

Related work on the zigbee protocol

SmartBridge-ZCL protocol

The SmartBridge-ZCL protocol is based on ZCL messages and is used to communicate between IP and ZigBee modules via serial interface [21]. ZCL Messages are either profile-wide general commands (e.g., Attribute Read/Attribute Write) or cluster-specific commands (e.g., Move to Level, On/Off State Update, etc.). The Zigbee module sends the messages. The ZCL messages are then processed into ZigBee packets and sent to the lights. The message format used in this protocol is:[Zcl,(address),(command,subcommand,parameters)][21], where addresses and commands are defined in the ZCL for the lighting system. The messages sent from the ZigBee module can be response messages containing lights attributes or commissioning information. Similar ZLL mechanisms and Zigbee protocols could be used to control the pixelated light strip as well, but extensions to the Zigbee protocol might be needed to have a better control and scalability over the LEDs in the pixelated strips.

Related work on performance metrics

Though various experiments with respect to the metrics like latency, packet loss, scalability were conducted to evaluate the Zigbee protocol extensions to control LEDs on a pixelated light strip, there can be other possible performance metrics too, like number of hops which can be further investigated.

A significant indicator of ZigBee network performance can be the number of hops between two Zigbee nodes. Fig. 8.3 is taken from previous work by Rajeev Piyare and Seong-ro Lee on "Performance Analysis of ZigBee (ZB) Module Based Wireless Sensor Networks"[22]. It shows the impact of the number of hops in a network. Here, the packet delay for this experiment is defined as the duration between sending a packet and the reception of the entire packet by the source also known as round-trip time (RTT). The distance between nodes is 3m. Fig.8.3 shows the results of direct transmission from an End Device to the Coordinator (Blue line) and indirect transmission through Routers. For single hop transmission, the average delay is around 0.11 sec for the maximum payload offered, while for two hops, the delay is 0.27 sec and for 3 hops, the delay is 0.37 secs[22]. Hence, the packet delay increases significantly with the number of hops due to an extra processing delay at the routers and re-transmission delay due to additional hops.

To control each pixel of a pixelated light strip, a large data array with color values must be sent to the light strip. Hence, the length of the Zigbee packets to be transmitted will be large. For instance, if a user is in the living room and has to control the light strip at a different location, then there might be a number of hops between the sending node and the light strip. In this case, there

can be a delay in controlling the light strip. From [22], it is clear that when the Zigbee network has more than one hop, the packet delay will be increased, which increases the overall delay and there can be a flicker on the light strip which is visible to the user. For streaming applications in particular, it is advisable to have fewer hops in the network.

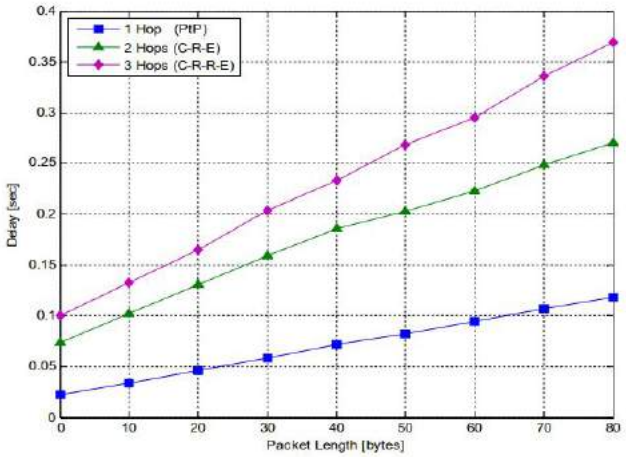

Figure 8.3: Number of hops vs Packet delay
[22]

Chapter 9

Conclusions

This chapter discusses the best fit Zigbee design option for the pixelated use cases based on the results discussed in Chapter 6.

The pixelated use cases defined in chapter 2 can be broadly classified into two categories, static and dynamic use cases. The use cases which involves setting up a nice color pattern on the light strip are considered to be static use cases. For instance, under ambience creation described in section 2.1, the light strips can be updated with a colorful pattern setting up a nice ambience. Scene creation can be considered static, since it involves sending preset values to the light strips. Use cases defined in section 2.2 involve continuous updating of the light strip at an interval of say 50ms. For instance, the streaming applications where the light strip has to react to music can be considered a dynamic use case. This is because the light strip keeps changing the pattern according to the streaming music.

9.1 Static use cases

For static cases, longer Zigbee packets with hard-coded color values must be sent to the light strips. Even if just one of the color values is lost while sending, then the design pattern on the light strip gets disrupted. Hence, there must be 0% packet loss for static use cases. The results presented in section 6.2 confirm that the Zigbee design extensions have negligible packet loss with only MAC retries. Though there are packet re-transmissions within the 802.15.4 stack, the overall latency for transmitting long Zigbee packets for a 4m length light strip lies below 200ms based on the results presented in section 6.3. Thus, it confirms that for static use cases, the design options discussed in section 5.2.2 are suitable to address the individual pixels of a LED strip.

9.2 Dynamic use cases

For dynamic use cases, like the streaming applications, the light strips must be highly reactive. In section 6.3, it is seen that sending longer Zigbee packets can have higher latency due to packet re-transmissions. The RTT of a Zigbee packet of 90 bytes takes up to 98ms after five MAC re-transmissions (worst case scenario). It is not recommended to send longer Zigbee packets to the light strip after many re-transmissions which produces high latency. Even the CPU load is more for such dynamic use cases, as discussed in section 6.5. Therefore, this design option is not suitable for streaming applications. However, there are alternative design options for dynamic use cases like streaming applications. One of the design options is virtual pixelation. Let's consider a pixelated light strip of length 3m consisting of 48 pixels. Fewer pixels can be selected in the light strip, say 12 pixels. An interpolation algorithm can be used to interpolate the color values between these 3 pixels. This design option is called virtual pixelation. This design option is not investigated in this thesis work. Other optimizations could involve updating only selected pixels.

Another design option is to configure the number of re-transmissions at different layers of the 802.15.4 stack. APS retry is disabled in this implementation. However, the MAC layer can retry upto 5 times which gives rise to high latency. In future work, the MAC retry can be completely disabled or configured to a lower value so that the latency remains less. For streaming applications, it is okay to lose some packet information. However, the latency must remain low so that the strips are highly reactive to sounds and music.

This thesis proposes three Zigbee design options to control individually addressable LED strips. While investigating on the hardware design of pixelated light strips used in Signify, it was decided that 16 pixels per meter will be used. Based on the Zigbee payload calculations as discussed in section 3.4, 16 pixels would be optimal to control with a Zigbee message of at most 82 bytes. Thus, the first research question is addressed by deciding on the acceptable number of LEDs on a light strip. This work also evaluates the designed options based on the performance metrics discussed in chapter 6. In addition to this, the thesis describes various use cases for pixelated light strips and evaluates the best fit Zigbee design for the defined use cases. To control the individually addressable LED strips, three Zigbee design options namely Stack fragmentation, Handling data subsets with single update of the light strip and Handling data subsets with intermittent updates of the light strip were investigated. After careful analysis and investigation, it was decided that sending data as subsets of longer Zigbee packets and updating the light strips after all the data subsets have been received at the destination node would be an efficient design option to control pixelated light strips. A software framework was developed on Mighty Gecko wireless SoCs from Silabs which uses the Zigbee protocol stack to implement this efficient design. After implementation, several experiments were done to evaluate the Zigbee extensions. During the investigation on performance metrics, it was decided that the experiments will be done based on packet loss, latency and CPU load. Thus, the second research research question is addressed by deciding and defining the performance metrics. The results obtained showed negligible packet loss with only MAC retries which proves Zigbee design extensions to be highly reliable and robust. While investigating packet loss, it was understood that the low percentage of packet loss is due to the packet re-transmissions at different layers in the Zigbee stack. The results obtained from analysis of packet length on latency showed that there could be a possible delay on the light strips while sending longer Zigbee packets. For dynamic use cases, like streaming applications where the light strips must be highly reactive to sounds and music, it is not recommended to send longer Zigbee packets with high latency to the strips. In this case, it was decided that either an alternative design called "Virtual Pixelation" discussed in 9.2 will be used or the number of packet-transmissions will be kept low in the software framework, to avoid high latencies. However, it is still okay to send longer Zigbee packets to the light strips for static use cases like setting a nice color pattern on the light strip as long as the total delay is less than 200ms. Finally, the third research question on deciding the best-fit protocol for the static and dynamic use cases is addressed.

9.3 Future work

Further investigations could be carried out on dynamic use cases like streaming applications. It could be investigated whether the software framework provided by Silabs allows us to configure the number of re-transmissions at different layers of the Zigbee stack. Though the APS retry was disabled in this implementation, MAC retries (maximum of 5) were kept as default values, provided by Silabs. It would be nice to have these reconfigured for the dynamic use cases. Because the dynamic use cases must have a delay below 30ms for every single update of the light strip. Hence, having more retries will produce a latency more than 30ms which is not suitable for streaming applications Another investigation could be carried out in measuring the number of hops to see if number of hops in a Zigbee network has an influence on latency. More research could be done on Virtual pixelation for streaming applications.

Bibliography

[1] I. Markit, "Iot trend watch." Information Available at: https://cdn.ihs.com/www/pdf/IoT-Trend-Watch-eBook.pdf, 2018. 1

[2] "Meet hue, philips lighting." Available at: https://www2.meethue.com/. 2, 8, 10

[3] J. Wang, "Zigbee light link standard and its applications," *IEEE Wireless Communications*, September 2012. 2, 16, 46, 47

[4] L. J. Shyan, S. Y. Wei, and S. C. Chou, "A comparative study of wireless protocols: Bluetooth, UWB, Zigbee and Wifi.," *Annual Conference of the IEEE Industrial Electronics Society (IECON).*, vol. 33, pp. 5–8, 2017. The paper compares Short range wireless protocols and some performance evaluation. 13

[5] "Zigbee specification, pp 425-551." Available at: https://www.Zigbee.org/download/standards-Zigbee-specification. 13, 18, 19

[6] "IEEE 802.15.4 specification." Available at: https://standards.ieee.org/standard/802_15_4-2015.html. 13, 15

[7] "Zigbee stack 3.0 user guide by NXP." Available at: https://www.nxp.com/docs/en/user-guide/JN-UG-3113.pdf. 13, 14

[8] "Simple tutorial about zigbee mesh networking." Available at: http://www.rfwireless-world.com/Tutorials/Zigbee-mesh-network-tutorial.html. 14, 15

[9] "Zigbee retry in emberznet stack of Silabs." Available at: https://www.silabs.com/community/wireless/zigbee-and-thread/knowledge-base.entry.html. 14

[10] "Wireless solutions part 4 on Zigbee." Available at: https://no.farnell.com/wireless-solutions-part-4-zigbee. 15, 16

[11] "Tutorial on key concepts used by APS layer." Available at: https://www.youtube.com/watch?v=DbjovO8zvOo. 17

[12] A. Potsch, "Zigbee frame overview at different layers." Available at: https://www.microchip.com/forums/download/,, 2006. 18, 19, 20

[13] "Detailed specs and datasheet for EFR32MG12 node." Available at: https://www.silabs.com/documents/public/data-sheets/efr32mg12-datasheet.pdf. 21

[14] "Detailed specs and datasheet for tlc5971 drivers provided by texas instruments." Available at: http://www.ti.com/lit/ds/symlink/tlc5971.pdf. 22

[15] "Flutter cross platform development tool." Available at: https://flutter.io/. 24

[16] K. Lui, "Performance evaluation of zigbee network for embedded electricity meters," *Digitala Vetenskapliga Arkivet*, 2009. 31, 32

[17] E. D. Pinedo-Frausto and J. A. Garcia-Macias, "An experimental analysis of zigbee networks," *IEEE Conference in Canada*, 2008. 38

[18] "Pixel strip product by enttec." Available at: https://www.enttec.co.uk/en/range/lights/individually-addressable-pixel-strip/. 45

[19] "Data sheet of components used by enttec pixel strip." Available at: http://dol2kh495zr52.cloudfront.net/pdf/datasheets/190221_8PL60-12.pdf. 45

[20] "Pixel strips manufactured by adafruit." Available at: https://www.adafruit.com/product/1138?length=1. 45, 46

[21] T. Bui, J. Lukkien, E. Frimout, and G. Broeksteeg, "Bridging light applications to the IP domain," *29th International Conference on Consumer Electronics (ICCE, Las Vegas NV, USA*, September 2011. 47

[22] R. Piyare and S. Lee, "Performance analysis of zigbee (zb) module basedwireless sensor networks," *International Journal of Scientific Engineering Research*, April 2013. 47, 48