Designing and Implementing a Web Application for Bluetooth Mesh Device Provisioning and User Management

Patrik Larsson

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6215
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2023

Acknowledgements

I would like to begin by thanking my industrial supervisors Mattias Wallinius and Maja Arvehammar, who conceptualized this project. They have been excellent supervisors. Secondly, I would like to thank my academic supervisor Karl-Erik Årzén and my examiner Martina Maggio for all their time and aid. Lastly, I would like to thank my colleague and friend Måns Hofvander for introducing me to the project and who has worked alongside me since day one.

Abstract

As technology progresses, it seems to become increasingly inevitable for everyday objects in our homes to incorporate advanced features, to increase functionality, convenience, and energy management. This master's thesis presents the continuation of a Bluetooth lighting system, aiming to develop a comprehensive website solution for handling the provisioning and configuration of Bluetooth Mesh devices. Bluetooth Mesh is one of the latest features of Bluetooth, which connects devices into a mesh network, allowing messages to be received and relayed by each device, offering a reliable and scalable network that can cover large areas. This thesis focuses on implementing a user-friendly web-based platform that simplifies the setup and management of Bluetooth Mesh networks.

The website development process encompassed several stages, including systems specifications, system design, implementation, and evaluation. The project uses well-established web development frameworks, such as HTML5, CSS3, and JavaScript, to create an intuitive and responsive user interface. The website integrates with Bluetooth Mesh devices using standardized protocols, allowing for device discovery, provisioning, and configuration.

The devices in question are custom-built circuit boards that, through the website, are included in a mesh network and can, with either the rising or setting of the sun or the change of a simple switch, turn on and off an E27 lightbulb.

Sammanfattning

I en tid av teknologiskt framsteg verkar det alltmer oundvikligt att vardagsföremål i våra hem integrerar avancerade funktioner för att öka funktionalitet, bekvämlighet och energihantering. Detta examensarbetet presenterar fortsättningen av ett Bluetooth-belysningssystem med målet att utveckla en omfattande webblösning för hantering av konfiguration och provianteringen av Bluetooth Mesh-enheter. Bluetooth Mesh är en av de senaste funktionerna inom Bluetooth, vilket möjliggör att enheter kan anslutas till ett mesh-nätverk där meddelanden kan tas emot och vidarebefordras av varje enhet. Det leder till ett pålitligt och skalbart nätverk som kan täcka stora områden. Denna avhandling fokuserar på att implementera en användarvänlig webbaserad plattform som förenklar inställning och hantering av Bluetooth Mesh-nätverk.

Webbplatsens utvecklingsprocess omfattade flera steg, inklusive kravanalys, systemdesign, implementering och utvärdering. Projektet utnyttjade väletablerade webbutvecklingsramverk som HTML5, CSS3 och JavaScript för att skapa en intuitiv och responsiv användargränssnitt. Webbplatsen integrerar med Bluetooth Mesh-enheter genom att använda standardiserade protokoll, vilket underlättar enhetsupptäckt, provianteringen och konfiguration.

Enheterna är kretskort som, genom webbplatsen, inkluderas i ett mesh-nätverk och kan med antingen solens upp- och nedgång eller en smart strömbrytare slå på och av en E27-glödlampa.

Contents

List of Tables

List of Figures

Acronyms

Acronym	Stands-for
AES	Advanced Encryption Standard
API	Application Programming Interface
BLE	Bluetooth Low Energy
CTL	Control
DSP	Design Science Paradigm
ECDH	Elliptic Curve Diffie-Hellman
GATT	Generic Attribute Profile
GDPR	General Data Protection Regulation
IoT	Internet of Things
IVI	Initialization Vector Index
MTU	Maximum Transmission Unit
NID	Network Identifier
NetMIC	Network Message Integrity Check
OOB	Out of Band
PDU	Protocol Data Unit
SAR	Segmentation and reassembly
SEQ	Sequence number
SRC	Source Address
SSO	Single sign on
TTL	Time To Live

1

Introduction

1.1 Background

This master's thesis is the continuation of the development of a Bluetooth lighting system that was started in 2021 at Adevo Consulting, building upon the result from a previous master's thesis conducted on the same project . [Hofvander, 2022] The system consists of Bluetooth-controlled lighting devices, a housing for the device, and a website, used to set up and configure the devices. The devices are luminaires affixed with a circuit board for control and communication, taking any E27 light-bulb. A first iteration circuit board and housing had been developed in prior phases of this project and will not be included in this master's thesis. Although, changes to the circuit board, and consequently the housing, are being made parallel with this master's thesis.

The project builds on Bluetooth Mesh technology, which is one of the newer features of Bluetooth. More specifically, it is a feature of Bluetooth Low Energy (BLE). BLE, as the name indicates, has low energy consumption, making it suitable for battery-powered devices, such as e.g. thermostats, fitness trackers, and head-phones. However, this project primarily uses the mesh functionality, as it is the way the devices communicate with each other. It links all the devices together into a network, and into applications within the network, when one device receives a message it broadcasts out the same message for other devices to receive and thus allows for a large message range. Bluetooth Mesh is explained further in Chapter 2.2. As the technology simply uses Bluetooth, it is not considered to be an Internet-of-things (IoT) device.

To implement the BLE mesh functionalities, a chip by Nordic Semiconductors is used by the devices. This chip implements the C-based open-source real-time operating system, Zephyr. See Chapter 2.5 for further information regarding Zephyr.

The website is an integral part of the system as it is used to establish a secure communication link with devices, connecting them to a network (connecting them

will be referred to as "provisioning" in this rapport) and sending essential information to the devices, such as time and location. Prior to this thesis, there was a skeleton website used by the company built with Node.js and Express.js, which are common models that use JavaScript as a primary language.

1.2 Purpose

The purpose of this master's thesis is to develop the website, with a main focus on implementing the provisioning, i.e., deriving and exchanging keys to enable secure communication between the user and the devices. Prior to this thesis, a third-party stand-alone application by Nordic Semiconductor had been used for provisioning. This is the most important part of this thesis and has taken the most time. To make the website fully functional, a user system needs to be designed and developed, as well as a login system. When this has been implemented, the company will be ready to reach out to different lighting vendors to sell this system, which is the ultimate goal of this project. If time allows, the website would be made into a framework so that the different lighting vendors can customize it to their liking.

1.3 Project Structure

The project has four main parts:

Part one: Javascript-based provisioning of Mesh network units. In this part of the master's thesis, the goal is to develop a Javascript-based provisioning of Bluetooth Mesh devices so that they can participate in a Bluetooth Mesh network. Due to the BLE technology being relatively new to the market, there are no extensive documentation nor ready-to-use APIs to perform web-based provisioning. The main difficulty of this part is developing and integrating a new API that uses Google Chrome BLE extensions, generating network and application keys using various methods of encryption, and sending the correct data needed according to the Bluetooth provisioning procedure.

Part two: Create a cloud-based web application with authentication via Single Sign On. The goal in this part is to use Google as a cloud platform, to develop a modern cloud-based web application storing user data around the user's BLE Luminaire network in a safe and secure manner. An overall important design factor is user integrity. This is a comprehensive method for storing user data that is in agreement with GDPR. The main difficulty is to find and understand the appropriate APIs for this and implement them into our existing website.

Part three: Store provisioning data from a user session locally on the browser device. In this part, the goal is to adapt the web application so that it functions from a device without an internet connection and that new network units can be

added to the BLE Luminaire Mesh network and the network can be administered. This will tackle problems such as storing data offline on local devices in a secure and reliable way. How to do this requires further research and will have several different solutions.

Part four: Create a framework for different hardware lighting vendors to style the website and view users. Using the same cloud-based application to create a new set of users allows styling and content change to the web application without allowing change to the core BLE Luminaire functionality. A basis and foundation for big data analysis for the cloud application data with high user integrity can also be prepared. This is in order to allow the product to be sold to hardware lighting vendors and for them to customize the website with their brand. It will require the website to be easily configurable and customizable, as well as for the APIs developed to be robust and user-friendly.

1.4 Limitations and Restrictions

The main limitations and restrictions of this project is the fact that this is a continuation of a product at Adevo and must thus be compatible with the company vision and with previous design choices, e.g., the decision to Bluetooth Low Energy and the Zephyr RTOS. The choice to use a web application, further explained in Section 4.2, limits which APIs that can be used.

1.5 Disposition

This thesis consists of 11 total chapters, the following list gives a short summary of each chapter.

- Chapter 1 is an introduction to the project, it contains the background to the product and the technology it is based on. It also presents the purpose and the project structure.
- Chapter 2 contains the theoretical knowledge of the technology used during the project, which is needed to understand the thesis. It also gives a brief overview of the home automation market and emerging standards that can affect the design of the product.
- Chapter 3 discusses the methodologies used in the product development.
- Chapter 4 lists the system specifications based on the customer's needs.
- Chapter 5 is the first technical chapter and an essential part of this thesis, it thoroughly describes the structure of the provisioning messages and lays out

the process of provisioning a device into the network. The chapter ends with a code example.

- Chapter 6 is the second technical chapter, where the message structure of a proxy message, i.e. a non-provisioning message, is described. Which is needed to, ultimately, send an application key-binding message, allowing the device to take part in the mesh network. The chapter ends with a code example.
- Chapter 7 shows the evaluation of different means of logging in and creating a user database. The chapter ends with a code example.
- Chapter 8 discusses the new home automation standard 'Matter' and evaluates whether it is in the company's best interest to adopt the standard.
- Chapter 9 presents the results and performance of the website.
- Chapter 10 discusses the results and the design choices made.
- Chapter 11 presents a conclusion and suggestions for future work and new potential products.

2

Theory

2.1 Home Automation

Home automation, or Smart homes, refers to the integration of technology into private homes to control and automate various functions such as lighting, HVAC, entertainment, and security. It was valued at a 40 billion USD market in 2020 and is estimated to grow into a 63 billion USD market by 2025 [Markets and Markets, 2020]. Home automation systems are usually controlled through a central hub, referred to as a gateway, or a smartphone app to control and monitor devices and systems remotely. These systems allow for increased comfort, convenience, and energy efficiency and thereby lower costs by automating various functions in the home.

Matter

The growth of the home automation market resulted in a working group being formed in 2019 called Project CHIP (Connected Home over IP) by a consortium of major technology companies including, but not limited to, Apple, Google, Amazon, and CSA (Connectivity Standards Alliance). The group has since grown to include over 500 companies and has the goal of developing a new open-source standard for the smart home, named Matter, that aims to provide a secure and reliable communication protocol for various smart devices, allowing interoperability between different brands of devices. [Nordic Semiconductors, 2022a]

Matter uses a combination of IP (Internet Protocol), Thread, and BLE (Bluetooth Low Energy, see Section 2.2) and is designed to be simple, scalable, and secure. Each device in the system is given an IP address using IPv6. It allows the user to communicate with devices and exchange data over the Internet. Thread is a wireless mesh network protocol, designed specifically for the home automation market [Nordic Semiconductors, 2022b] and is the primary way devices communicate with each other. It is a proprietary technology bought by the Thread Group in 2014 and is supported by many of the same companies as Matter is, e.g. Google, Apple, Amazon and Samsung. The technology differs from Bluetooth Mesh in that it was designed for scalability and to be more compatible with IPv6 resulting in

lower power consumption than BLE. [Thread Group, 2023]. The range of the signal differs from Bluetooth mesh as well in that its maximum outdoor range is 30m [devops, 2021], whilst Bluetooth mesh has a maximum range of 200m [Bluetooth, 2017a]. Matter does, however, also use Bluetooth for commissioning, i.e., provisioning the device into the network. [Nordic Semiconductors, 2022a]

The standard provides a unified way for devices to discover and communicate with each other, regardless of the manufacturer, and it supports a wide range of device categories, from smart locks and lights to sensors and thermostats.

Matter includes security features to ensure the privacy and security of data transmitted over the network. The standard uses encryption to secure communication between devices and uses certificate-based authentication to ensure that only authorized devices can join the network. Additionally, Matter provides regular security updates to ensure the continued security of the devices. [Nordic Semiconductors, 2022a]

The Matter standard is expected to play a significant role in the growth of the home automation market, as it provides a unified and secure communication protocol that will allow consumers to seamlessly integrate devices from different manufacturers into their homes. The adoption of the standard is expected to increase in the coming years as more devices are certified to be compatible with the Matter standard.[Nordic Semiconductors, 2022a]

2.2 Bluetooth

Bluetooth is a wireless communication technology that allows devices to connect and exchange data over short distances. It was invented in 1989 by Nils Rydbeck, CTO at Ericsson, a Swedish telecommunication company based in Lund, Sweden, with the goal of creating a standard for wireless communications between mobile phones and other devices [Jeanette Irekvist, 2022]. Since then, it has become a widely adopted technology for a variety of applications, including mobile phone, audio and video streaming, wireless headset and speaker systems, and internet-of-things (IoT) devices.

The Bluetooth technology has since 1998 been taken over by Bluetooth Special Interest Group (SIG), which manages the development and licensing of the technology. [Jaap Haartsen, 2018] The latest version of Bluetooth, Bluetooth 5.2, was released in December 2021 and provides enhanced features such as increased data rates, longer range, and improved coexistence with other wireless technologies. With its widespread adoption and continued development, Bluetooth is expected to play an important role in the growth of wireless applications in the future.

Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a low-power version of Bluetooth technology designed for use in remote devices. BLE was introduced in 2010 with the release of Bluetooth 4.0 and has since become a popular choice for remote devices due to its low power consumption and low cost. BLE operates in the same 2.4 GHz frequency band as classic Bluetooth but uses a simpler protocol and requires less power, making it well-suited for small, battery-powered devices. BLE devices can be connected to smartphones, tablets, and computers, allowing for seamless communication and data exchange between devices. BLE has an estimated maximum range of 200 meters. [Nordic Semiconductors, 2021]

Bluetooth Mesh

One of the main features of BLE is mesh capabilities, i.e. the capability to include devices, known as nodes, in a mesh network and allow for many-to-many communication. The mesh network operates with a flooding principle, i.e. that nodes receive and relay incoming messages to more nodes, leading to a cascading effect where a message from a single node can be relayed from node to node until the entire network of nodes has received the message, as long as each individual node is in the range of at least one other node in the network. This allows the mesh network to cover a large area. The messages sent do not need to originate from a mesh device either, they can be sent from an external device, e.g. a mobile phone or a laptop. Messages can also be targeted to a specific device, i.e., a message can be sent to a specific device within the network, and all other devices do not read the message and simply relays the message, ensuring that only the targeted devices receive it. [Woolley, 2020]

GATT

The Generic Attribute Profile (GATT) is a standardized protocol within BLE that defines how devices exchange data and interact with each other. It consists of two main components: services and characteristics. Services represent a collection of related data and behaviors, while characteristics define specific data values within a service. For example, there is a GATT server that handles communication with BLE devices regarding provisioning, and there is another proxy service that handles all other messages. GATT enables devices to discover, read, write, and notify each other. It also allows for the establishment of connections between devices. GATT plays a vital role in enabling interoperability and efficient communication between BLE devices, and web applications, making it a fundamental component of many Bluetooth applications and systems. [Bluetooth, 2021]

Provisioning

Bluetooth provisioning is the process of setting up and configuring Bluetooth devices. This process involves establishing a secure connection between the device

and the user's external device and transferring the necessary information, such as the network keys, to the device. This allows the device to partake in a mesh network, where devices within the network share the same keys and thus have the ability to communicate with each other. It uses an assortment of different encryption methods to achieve a secure communication channel. The use of Bluetooth provisioning helps to reduce the complexity and cost of setting up and configuring devices.

2.3 Encryption

Elliptic Curve Diffie-Hellman

Elliptic Curve Diffie-Hellman (ECDH) is a key agreement algorithm used to securely exchange cryptographic keys over an insecure communication channel. It is a fundamental building block in many secure communication protocols, including Bluetooth Provisioning, and is widely used in a variety of applications, including secure web browsing, virtual private networks (VPNs), and secure data transmission. [Pierre Philip du Preez, 2020]

ECDH is based on the mathematical concept of elliptic curve cryptography, which provides strong security with relatively short keys compared to traditional cryptography methods. The algorithm allows two parties to agree on a shared secret key without any prior knowledge of the key, and without the risk of the key being intercepted by a third party during the exchange. [Pierre Philip du Preez, 2020]. It has been widely adopted due to its efficiency, security, and compatibility with existing cryptography standards. It is also included as a standard in many cryptography libraries and is supported by a variety of hardware devices, making it a popular choice for secure communication in both software and hardware applications. [Pierre Philip du Preez, 2020]

In recent years, ECDH has been widely adopted in the IoT and other embedded devices, as it provides a secure and efficient way of exchanging keys in these devices, which typically have limited computational resources. [Pierre Philip du Preez, 2020]

The following algorithm, in Figure 2.1, is a simplification of, and a way to visualize, how an ECDH key exchange between Alice and Bob is performed. The algorithm uses colors and the mixing of them to represent the number and mathematical formulas used in the process. In the real algorithm, the colors represent points on an elliptic curve, the mixing of the colors represents a non-reversible mathematical operation, and the resulting 'Common secret' is a 64 byte number used as a common key to encrypt messages. The exchange, noted as 'Public transport' in the figure, is done over an unsecure channel, i.e. third parties can read the

message, but due to the non-reversible mathematical operation, it will not matter as they will not be able to derive the 'Secret colours'.

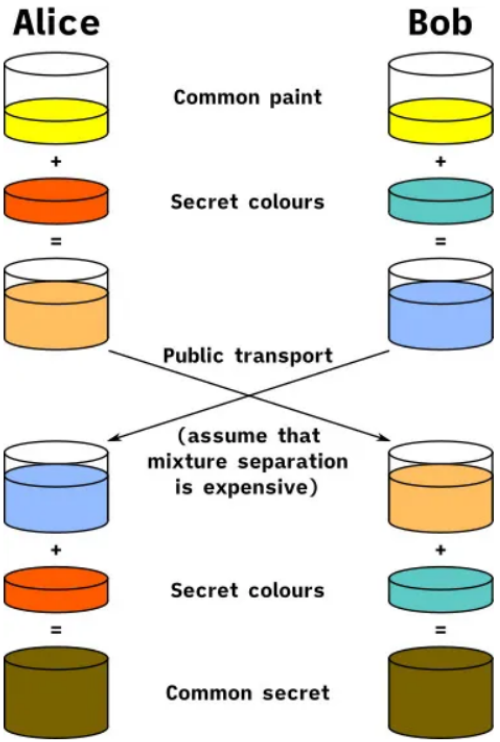

Figure 2.1 ECDH simplification

Advanced Encryption Standard

Advanced Encryption Standard (AES) is a widely used symmetric encryption algorithm that provides secure encryption of sensitive information. It is a popular standard for a variety of applications, including secure data storage, data transmission, and secure communications. AES is widely adopted in Bluetooth's own encryption function, as seen in the next section. It is not necessary to understand these functions to understand the thesis, but they are used and will be referred to in later chapters. [Woolley, 2020]

1. `AES-CCM(key, nonce, data)`:

    - Purpose in this thesis: Encrypts and authenticates variable length data using AES Counter with CBC-MAC (CCM).

- Inputs:
  - `key`: 128-bit key for AES encryption.
  - `nonce`: Number used once.
  - `data`: Variable length data
- Outputs:
  - `ciphertext`: Encrypted data
  - `mic`: Message integrity check value.

2. `AES-CMACT(key, data)`:

   - Purpose in this thesis: Generates a Message Authentication Code (MAC) during the provisioning process.
   - Inputs:
     - `key`: 128-bit key for AES-CMACT.
     - `data`: Variable length data
   - Output: Message Authentication Code (MAC).

3. `AES-ECB(key, data)`:

   - Purpose in this thesis: Encrypts data in proxy messages
   - Inputs:
     - `key`: 128-bit key for AES encryption/decryption.
     - `data`: Variable length data
   - Output: Encrypted data.

Bluetooth's encryption

Bluetooth uses AES when encrypting data in messages, here is a list of functions used during the provisioning process and application key binding. It is not necessary to understand these functions to understand the thesis, but they are used and will be referred to in later chapters. [Woolley, 2020]

1. `s1(M)` SALT Generation Function:

   - Purpose in this thesis: Generates a SALT value using AES-CMAC with a fixed key and input data. SALT is a random string generated from another string.
   - Inputs:
     - `M`: ASCII string.
   - Output: SALT value.
   - SALT Generation Process:

- Apply AES-CMAC with a fixed key and ZERO (128-bit value) as the input to generate the SALT value.

2. `k1(N, SALT, P)` Function:
   - Purpose in this thesis: Generates a confirmation value from EDCH secret and all values sent during the provisioning process.
   - Inputs:
     - `N`: EDCH secret.
     - `SALT`: Confirmation inputs.
     - `P`: fixed string of "prck".
   - Output: Confirmation value.
   - Key Computation Process:
     - Compute the key `T` using AES-CMAC with `SALT` and `N`.
     - Compute the AES-CMACT result using `T` and `P`.
3. `k2(N, P)` Encryption key and Privacy Key generation from network key :
   - Purpose: Generates EncryptionKey, PrivacyKey, and NID.
   - Inputs:
     - `N`: Network key.
     - `P`: Fixed string of "00".
   - Output: String containing EncryptionKey, PrivacyKey, and NID.
   - Key Computation Process:
     - Compute SALT using s1("736d6b32")
     - Compute the key `T` using AES-CMAC with `SALT` and `N`.
     - Compute `T0`, `T1`, `T2`, and `T3` using AES-CMACT with `T`, `P`, and specific values.
     - Compute the key material `k2` by combining `T1`, `T2`, and `T3` modulo 2263.
4. `e(data, PrivacyKey)`:
   - Purpose in this thesis: Generates encryptedData of the string data using the PrivacyKey.
   - Inputs:
     - `data`: A string of the data that will be encrypted.
     - `PrivacyKey`: PrivacyKey.
   - Output: encryptedData.
   - Encryption Process:
     - Perform AES-ECB encryption using `data`, `PrivacyKey`

2.4 Login and userdata

GDPR

The General Data Protection Regulation (GDPR) is an EU regulation that governs how personal user data is collected, processed, and stored by organizations. It is designed to protect the privacy and personal information of individuals within the EU and applies to all companies that process or store the data of EU citizens. [Ben Wolford, 2020]

In terms of storing user data, GDPR ensures that companies are legally obligated to obtain explicit consent from the user before collecting their personal data, and must clearly outline how that data will be used. Additionally, GDPR requires companies to provide individuals with the ability to access, modify, and delete their personal data. [Ben Wolford, 2020]

All data that can be used to identify the user by a third party falls under GDPR, e.g. name, email and address, but, also information about the user, e.g. race, gender and religion. [Ben Wolford, 2020]

SSO

Single Sign-On (SSO) is a widely used feature in applications that allows users to input their credentials, e.g. username and password, and to authenticate themselves once in a carrier, and then use that authentication to access individual applications without having to re-enter their credentials. This is in comparison to the traditional authentication model where the user needs to input their credentials, i.e. creating a new profile, for each application. A common way to implement this feature is to use Open-Authentication (OAuth), which allows the application access to the SSO for commonly used carriers such as Google, Apple, and Facebook, i.g. allowing the user to login with one of these companies instead of creating a new profile. [Peyrott, 2023].

2.5 Zephyr & Nordic Semiconductor

Zephyr is a real-time operating system (RTOS) developed by the Zephyr Project, a part of the Linux Foundation project. It is an open source collaboration to build a small, scalable RTOS for resource-constrained devices. The aim is to reduce cost and accelerate time to market for the development of embedded devices. Zephyr is a good choice for small connected sensors, modems and wireless gateways. Zephyr is widely implemented by Nordic Semiconductors on the programmable developer kits and the System on Chip (SoC) they offer. Together, Zephyr and Nordic Semiconductors offer several desirable features for developers. A notable feature is the device tree modification capabilities in their software templates. This allows devel-

opers to customize the hardware configuration according to their specific requirements. Another feature is the implementation of Bluetooth Low Energy and includes Bluetooth mesh [Semiconduct, 2020]. Zephyr also implements the Generic Attribute Profile (GATT) for BLE communication. GATT is a standardized framework that defines the structure and behavior of services and characteristics in BLE devices [Zephyr, 2023].

2.6 Web Bluetooth API

The Web Bluetooth API, developed by Google, uses the Generic Attribute Profile (GATT) as a foundational protocol for communication with Bluetooth devices. GATT is used to enable web applications to interact with Bluetooth devices. By implementing GATT, the API provides a standardized framework for discovering nearby Bluetooth devices, establishing connections, and exchanging data. It allows developers to access the services and characteristics offered by Bluetooth devices, e.g. devices by Nordic Semiconductors that use Zephyr, enabling read and write operations, as well as subscription to notifications. The utilization of GATT within the Web Bluetooth API ensures interoperability and compatibility across different Bluetooth devices, enabling web applications to communicate effectively with a wide range of Bluetooth devices. [Beaufort, 2023]

3

Methods

3.1 Agile

Agile development has been one of the main ways to develop software in the industry, since its introduction in 2001 with the release of the Agile Manifesto [Beck, K. et al, 2001]. One of the key tenets of Agile programming is to divide the development into small increments of planning, design, development, and testing before receiving feedback from customers and then continue with a new iteration of phases from planning to feedback. Typical for agile programming is working with the kanban cards where tasks are divided into smaller sub-tasks, given an estimated time to execute, and are then moved in between tasks that need to be done, are being done, and have been done. All, in order to give the developer a clearer overview of all tasks and what needs to be done. It also means the developer does not get stuck with tasks too large to handle, and instead gives them reasonably sized tasks and a better sense of progress. A common practice within Agile is to divide projects into so-called sprints and stand-ups. This project has stand-ups twice a week, where a 15-minute meeting is held to discuss which issues are encountered and what is currently being worked on. Sprints are longer meetings, held every two weeks and functions to more thoroughly plan the comming two weeks.

3.2 Ulrich and Eppinger's Methodology

The development of the devices has followed the product development methodology by Ulrich and Eppinger [Ulrich, 2004] in previous phases. Ulrich and Eppinger's methodology is primarily used to develop physical products and is not commonly used for designing websites. However, parts of the methodology are useful to understand customer needs and the desired specifications of the project. It also works well together with agile programming as one of the key tenets of U&E is to divide the problem into sub-problems, similar to agile programming. Thus, the methodology will be used, in part, in this thesis. Particularly, to evaluate different APIs and databases, as well as understand customer needs and specifications.

3.3 Design Science Paradigm

Another way of tackling software design problems is to use the Design Science Paradigm (DSP), which is a framework used to develop practical solutions to real-world problems and has been in use in fields such as information systems and management. It is gaining traction in software development as well. It is suitable to partially use in this thesis due to sharing much of the same philosophy of U&Es methodology in that DSP involves identifying problems, designing solutions, and validating those solutions [Runeson et al., 2020]. This thesis did not cover or utilize all the guidelines of DSP since the paradigm was not initially known to the company and proved challenging to implement during the later stages of development.

4

System specifications

4.1 Overview

This is a brief chapter that aims to further explain the system specifications, which are done as one of the steps of Ulrich & Eppinger's design methodology. The system specifications will look at the needs of the customer, which in this case are my supervisors at Adevo, and will function as a more in-depth evaluation of the purpose of this project.

4.2 Website

As mentioned before, this project centres around the development of the website for the provisioning and configuration of the lighting devices. This is the first need of the customer, i.e., that the lights are controlled from a web application. This is due to a company evaluation of the home-automation market, leading to the conclusion that the mobile app market is over saturated, i.e. that there are too many unnecessary apps. Alongside being an over saturated market, apps require to be downloaded by the user and regularly maintained and updated by the developers. This led to the use and development of a web based application for the control of the devices.

4.3 Provisioning

The provisioning of the devices is the main function of the website and needs to be able to be done with as few steps, i.e. clicks, as possible, as one of the key tenets of the product is to be as simple to use as possible. Provisioning also needs to be reliable, fast and reproducible.

4.4 Users & Data collection

The website will need to have a login system, to ensure that no one else can connect to your devices, i.e., a user-base needs to be established. The login process should

be cloud-based, simple and secure. The website should also limit the amount of data gathered about the user, partly to avoid the user having to give consent due to GDPR, see Section 2.4, and thus adding steps and complexity to the login process, but mainly because it goes against the company's ethics to save unnecessary amounts of information about the user. If time allows, there should be an off-grid alternative to logging in, i.e. all user-data should be stored locally and thus no user-data can be stored.

4.5 Customer Needs

The customer needs, described above, can be seen in Table 4.1, alongside with an importance rating of each need, which will determine the need's priority.

Table 4.1 Customer Needs

No.	Need	Importance
1	System needs to be a web application	5
2	Prov. needs to be secure	5
3	Prov. needs to be reliable	4
4	Prov. needs to be fast	3
5	Prov. needs to be simple	4
6	User-data needs to be minimized	5
7	Login needs to be fast	4
8	Login needs to be simple	4
9	Login needs to be cloud-based	3
10	Login needs an off-grid alternative	1

5

Provisioning

5.1 Overview

To communicate with a Bluetooth device, a communication link needs to be established between the devices. For this project, the link will be between the device and a web browser. To achieve this, an API, called Web Bluetooth API, has been developed by Google for their Chrome browser, which is also compatible with Microsoft Edge and Opera. The API is still being developed and has changed several times during the project. There are unfortunately not any other APIs available for web-based Bluetooth communication, and thus the project needs to use this one. The API is used both for communicating to the provisioning service mentioned in section 2.2[explain the different services] and the proxy service, which is used for all other communication, such as Key Binding, see Section 6, and sending time, date and location for instance.

For the Web Bluetooth API to connect to an unprovisioned device, the device broadcasts an 'Unprovisioned device beacon' that contains the unique UUID allocated to each unit, which allows for an unsecured link to be established. To secure the link, provisioning needs to take place where secure keys are generated and exchanged, as seen below in Section 5.2.

5.2 Provisioning Process

To establish a secure link, a series of messages called Protocol Data Units, PDU for short, need to be sent, which is described in the specification Mesh Profile [Bluetooth, 2017b] and can be seen below in the sequence diagram in Figure 5.1. The following subsections will delve deeper into what each PDU entails.

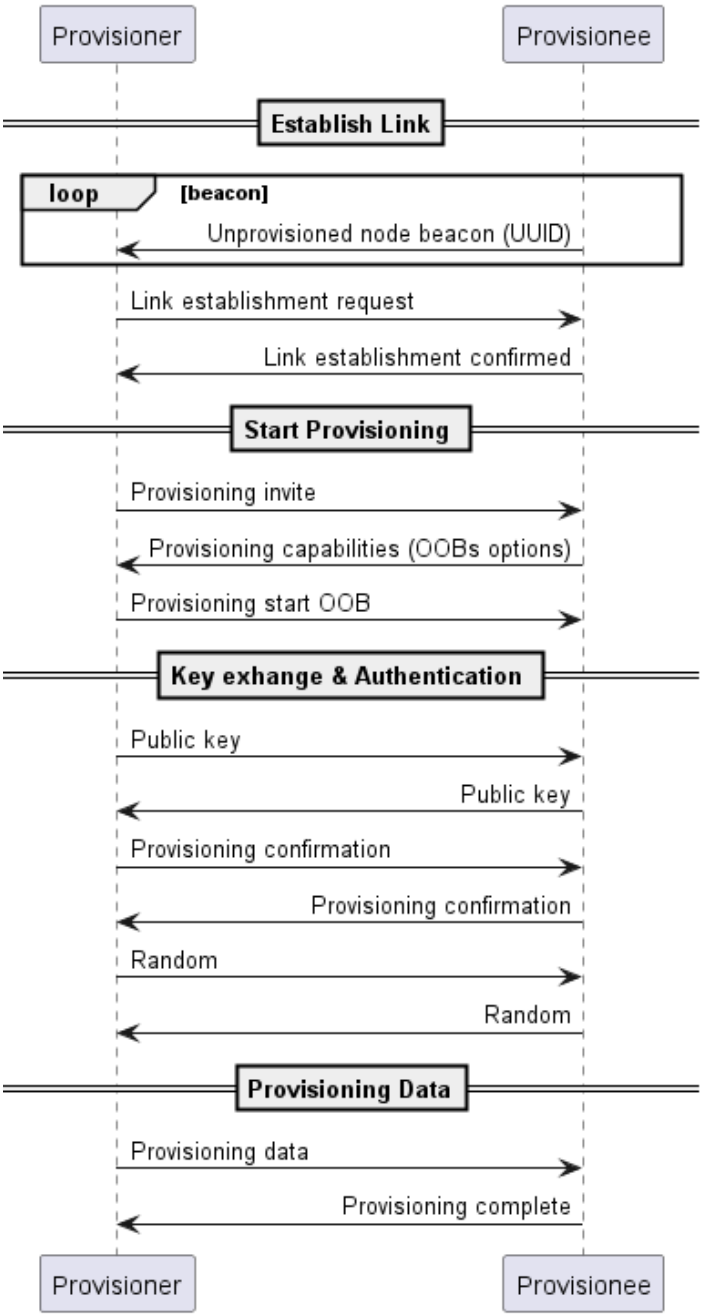

Figure 5.1 UML sequence diagram of provisioning process

Provisioning PDU

This subsection shows the structure of a generic Provisioning PDU. In Figure 5.2 a Provisioning PDU is visualized. The two first bits in the first byte (referred to as an 'Octet') are Segment-And-Reassemble, SAR, bits that show whether the message is segmented or not. The maximum transmission unit (MTU) of a PDU is 24 bytes, which some of the Provisioning PDUs exceed, meaning there is a need for segmentation for some of the messages. The different values for the SAR bits can be seen in Table 5.1. The type value indicates which type of PDU is being sent, which in this case are Provisioning PDUs, and has the type value of 0x03 given by [Bluetooth, 2017b].

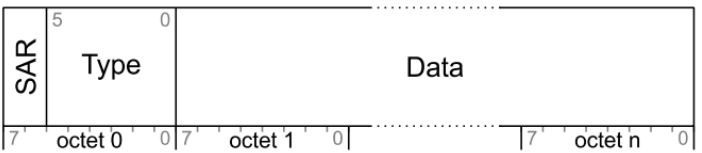

Figure 5.2 Provisioning PDU Format

Table 5.1 Segment and Reassemble

Field	Value	Description
SAR COMPLETE	0b00	An unsegmented message
SAR FIRST	0b01	The first segment of a segmented message
SAR CONTINUE	0b10	The continuation of a segmented message
SAR LAST	0b11	The last segment of a segmented message

The variable-sized segment call Data in Figure 5.2 contains the unique information for each different provisioning PDU, which is shown in Figure 5.3 and in Table 5.1 more information is given. The first byte consists of two bits of padding followed by six bits that indicate which provisioning PDU is being sent, which can be seen in Table 5.3 along with a short description of what each PDU seeks to accomplish. After the first byte, a variable-sized segment is attached, which size and value change depending on which type of Provisioning PDU is being sent.

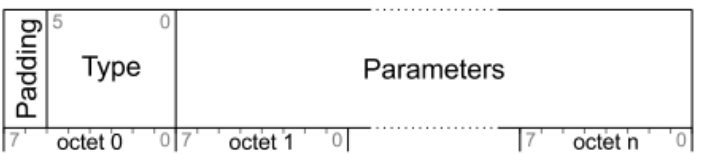

Figure 5.3 Provisioning PDU Data Format

Table 5.2 Provisioning PDU Format

Field	Size [bits]	Description
Padding	2	0b00, Fixed Value
Type	6	Provisioning PDU Type
Parameters	variable	Message Parameter

Table 5.3 Provisioning PDU Types

Field	Type value	Description
Prov. Invite	0x00	Invites a device to join a mesh network
Prov. Capabilities	0x01	Indicates the capabilities of the device.
Prov. Start	0x02	Indicates the provisioning method.
Prov. Public Key	0x03	Contains the ECDH Public Key.
Prov. Input Complete	0x04	Completed inputting a value.
Prov. Confirmation	0x05	Provisioning confirmation value.
Prov. Random	0x06	Provisioning random value.
Prov. Data	0x07	Unicast address, a network key, NetKey Index, Flags and the IV Index
Prov. Complete	0x08	Provisioning is complete
Prov. Failed	0x09	Provisioning was unsuccessful

Provisioning Invite

The first Provisioning PDU being sent is the Provisioning Invite, which is sent from the provisioner to the unprovisioned device and has the Type value 0x00 according to Table 5.3 and has a one-byte parameter, see Table 5.4. This PDU invites the device into a Provisioning session and for the device to draw the attention of the user in a preprogrammed way, for example by blinking lights. This is so that the user has an external way of identifying which device is being provisioned. The Attention Duration parameter in the PDU is the number of seconds the device should draw the user's attention.

Table 5.4 Provisioning Invite PDU parameters format

Field	Size [bytes]	Description
Attention Duration	1	Attention Timer State

Provisioning Capabilities

The device responds with a Provisioning Capabilities PDU, which has the Type value 0x01 according to Table 5.3. The PDU sends the provisioner the device's provisioning capabilities, mostly the device's OOB (Out of Band) capabilities, which is an optional security measure that requires physical interaction with the device. Seen in Table 5.5 For example, it can be to press buttons on the device a certain amount of times. For this project, it is not of interest to use any OOB options, as the goal is for provisioning to be as seamless as possible, and require as few interactions with the device as possible.

Table 5.5 Provisioning Capabilities PDU parameters format

Field	Size [bytes]	Description
Number of Elements	1	Nbr of supported elements
Algorithms	2	Supported algorithms
Public Key Type	1	Supported public key types
Static OOB Type	1	Supported static OOB types
Output OOB Size	1	Max. size of output OOB
Output OOB Action	2	Supported Output OOB Actions
Input OOB Size	1	Max. size of input OOB
Input OOB Action	2	Supported Input OOB Actions

Provisioning Start

The provisioner then sends a Provisioning Start PDU, with the Type value of 0x02, seen in Table 5.6. The goal of this PDU is for the provisioner to send the chosen OOB capabilities, alongside the algorithm to generate the Public Key, which is explained below in the subsection "Provisioning Public Key". As of writing this report, there is only one option of algorithms to choose from, that being using Elliptic Curve Diffie-Hellman with the curve "FIPS P-256".

Table 5.6 Provisioning Start PDU parameters format

Field	Size [bytes]	Description
Algorithm	1	Algorithm used for provisioning
Public Key	2	Public Key used
Authentication Method	1	Authentication Method used
Authentication Action	1	Selected OOB action
Authentication Size	1	Size of OOB used

Provisioning Public Key

In this subsection, the public key for the provisioner is generated and sent to the device. The Type value for the PDU is 0x03 and the parameter is 64 bytes, i.e., the message needs to be segmented. With a MTU of 24 bytes, that means the message needs to be segmented into three segments. In this step a unique private and public key need to be generated according to the ECDH encryption algorithm with the P-256 curve, see Section 2.3 for a more thorough explanation. To generate the keys an API developed by MIT is used [input reference?], which performs the calculations quickly in pure JavaScript, meaning it is easy to implement into the project. The API generates a 64-byte key, where the first 32 bytes are the X component of the point on the curve and the next 32 bytes are the Y component, which is the parameter for this PDU as seen in Table 5.7. When the key is received in the device, it responds with its public ECDH key.

Table 5.7 Provisioning Public Key PDU parameters format

Field	Size [bytes]	Description
Public Key X	32	The X component of public key for the FIPS P-256
Public Key Y	32	The U component of public key for the FIPS P-256

Provisioning Input Complete

As the previous message was a segmented message, the device sends an Input Complete message as confirmation that the segment was completed and properly reassembled. This message has the Type value 0x04 and does not have a parameter.

Provisioning Confirmation

At this point in the provisioning process, a confirmation value is bilaterally calculated using all the shared data up until this point, i.e., the attention duration, the provisioning capabilities, chosen OOB and algorithm, the public device key, the public provisioner key and a random 16byte value which will be sent in the next PDU. All these values are used to create a unique confirmation value, as seen in Table 5.8, that the device and provisioner can use to verify that all the values sent so far are correct, which is done in the next step when a random value is sent. The

confirmation value is calculated according to the following algorithm.

$$
\begin{aligned}
EDCHSecret &= privateKey * deviceKey \\
rand &= random(16bytes) \\
temp &= inviteVal + capVal + startVal \\
confInputs &= temp + publicKey + deviceKey \\
confSalt &= salt(confirmationInputs) \\
confKey &= k1(ECDHSecret, confirmationSalt, "prck") \\
confVal &= AesCmac(confKey, rand)
\end{aligned}
$$

where the ECDHSecret is ECDH multiplication of the provisioner's private key and the device's public key, and where the functions salt(M), k1(N, salt, info) and AesCmac are explained in Section 2.3.

When the device receives this PDU it will respond by sending its own Confirmation value. Note that the confirmation value can not be confirmed until the next PDU, where the randomly generated value "rand" will be sent.

Table 5.8 Provisioning Confimation PDU parameters format

Field	Size [bytes]	Description
Confirmation	16	The values exchanged during the Prov. process so far

Provisioning Random

This PDU sends the random 16 bytes "rand" value, shown in 5.9, that was used to calculate the confirmation value in the previous PDU. When this PDU is received by the device, it will respond with its own random value

Table 5.9 Provisioning RandomPDU parameters format

Field	Size [bytes]	Description
Random	16	The final input to the confirmation

Provisioning Data

This PDU is the most important, as it will share the encrypted information and keys that will be used for communicating securely. The parameters for this PDU can be seen in Table 5.10. The information being sent is a network key alongside a network key index (used for faster searching after keys), a key refresh flag, IVIndex, and a

unique unicast address. The network key and network key index are used to invite the device into a network. The key refresh flag and IVIndex are used for security measures that are not mandatory and are not relevant to this project. The deviceKey is derived in this segment and is used to encrypt PDUs that are being sent directly to the device, which will be used to send an appkey in Chapter 6. Below you can see the algorithm that generates the data being sent as well as how it is encrypted.

```
provInput   = confSalt + provRand + deviceRand
provSalt    = salt(provInput)
deviceKey   = k1(ECDHSecret, provSalt, "prdk")
sessionKey  = k1(ECDHSecret, provSalt, "prsk")
sessionNoice    = k1(ECDHSecret, provSalt, "prsn")
netkey      = random(16 byte) | existing netkey
netkeyIndex = "0000"  | existing netKeyIndex
flags       = "00"
IVIndex     = "0000"  | existing IVIndex
unicast     = uniqueNumber(2 byte)
data        = netKey + netKeyIndex +
                + flags + IVIndex + unicast
encData     = CCM(sessionKey, sessionNoice, data)
MIC         = Last 8 bytes of encData
```

Table 5.10 Provisioning Data PDU parameters format

Field	Size [bytes]	Description
Encrypted Prov. Data	25	Encrypted network key, netkey index, netkey refresh flag, IV Flag, IV index and unicast address.
Prov. data MIC	8	PDU Integrity Check Value

where the ECDHSecret, confSalt, provRand and deviceRand are values derived in the Provisioning Confirmation PDU. The function CCM(key, nonce, data) is explained in Section 2.3.

5.3 Code Examples

Below is a code example for the function getProvKey() that gives the PDU message for the Provisioning Public Key PDU.

```
static getProvKey = function () {
    let type = ProvValues.TYPE.PROV_KEY;
```

```
/*Type Value = 0x03*/

let key = ProvValues.getProvKey();
let par = type + key;

let seg0 = par.toString().slice(0, 42);
let seg1 = par.toString().slice(42, 86);
let seg2 = par.toString().slice(86);

let message0 = ProvValues.SAR.FIRST + seg0;
/*
ProvValues.SAR.FIRST = PDUType|SAR =
ProvisioningPDU|First = 0x03 | 0b01000000 = 0x43
*/
let message1 = ProvValues.SAR.CONT + seg1;
let message2 = ProvValues.SAR.LAST + seg2;

let message = message0 + message1 + message2;
return message;
}       //message is later split back into message0,
          message1 and message2
```

where ProvValues.getProvKey() returns a 64 byte EDCH public key.

6

Key Binding

6.1 Overview

This chapter aims to describe how to bind an application key to the provisioned device, i.e. invite the device into an application. An application is a cluster of devices that use a common key to communicate with each other. It differs from a network in that one network can consist of several applications. The application key is, however, not shared over the provisioning service as the previous PDUs, instead, it communicates over a proxy service, 2.5. Thus, the PDUs have a different format, with extra levels of security and, consequently, higher complexity. It is also over this service that PDUs containing time and location, as well as an on or off signal, are sent. When sending different messages using Proxy PDUs the only difference will be a different opcode, i.e. the opcode shows which Zephyr service that shall receive it, and a parameter that holds the payload of the message, e.g. the application key when trying to send the application key-binding message.

6.2 Proxy PDU

In the following sections, the general format of a proxy PDU, and then all different segments of the PDU, will be explained, in order to understand how to send an application key binding message.

Proxy PDU Format

In Figure 6.1 the structure of a general proxy PDU can be seen, and it can be seen that its format is more complex compared to a provisioning PDU seen in Figure 5.3. It consists of the IV Index (IVI) and Network Identifier (NID) in the first byte, index 0 in Figure 6.1. The control (CTL) and Time To Live (TTL) are in the second byte, followed by a three-byte long Sequence Number (SEQ) and a two-byte Source Address (SRC), will be obfuscated. Obfuscation is an umbrella term of a security measure that deliberately makes the data harder to read and understand, in order

to protect against certain attacks. Then a two-byte Destination Address (DST), followed by the variable-sized TransportPDU, which contains the payload, is attached. These two are encrypted. Lastly, a Network Message Integrity Check (NetMIC) is attached. These segments and their size are listed in Table 6.1 and will be explained more thoroughly in the following sections.

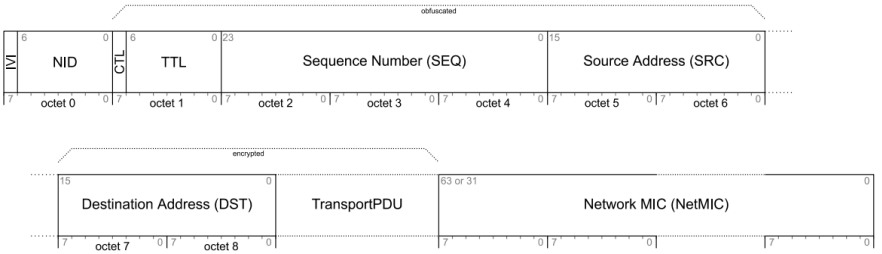

Figure 6.1 Proxy PDU Format

Table 6.1 Proxy PDU Segment Size

Name	Size [bit]
IVI	1
NID	7
CTL	1
TTL	7
SEQ	24
SRC	16
DST	16
TransportPDU	8 to 128
NetMIC	32 or 64

IV Index & Network Identifier

The IVI is the least significant bit of the IV Index which is sent during the provisioning process. The IVI is used to extend the lifetime of the network, by counting up when the SEQ has reached its maximum value and needs to be reset. The NID is a value derived from the Network key, shared during the provisioning process, and is used to verify which network the device belongs to, alongside the privacy key and encryption key. It is derived using the k2(N, P), explained in Section 2.3. The two keys are used in the following segments.

Network Control & Time To Live

The CTL is a fixed bit depending on which type of message is being sent, either an access message or a control message, as seen in Table 6.2. The type of message also determines the size of the NetMIC, as seen in the table. App key binding messages are a type of access message. The TTL is a seven-bit value that indicates whether the message can be relayed to other devices, as seen in Table 6.3. When sending an application key binding, it is sent to a specific device and is not meant to be relayed further on. Thus the TTL value should be either 0 or 1, according to Table 6.3.

Table 6.2 Network control field

CTL field	Message Type	NetMIC Size (bits)
0	Access Message	32
1	Control Message	64

Table 6.3 Time To Live Field

Value	Description
0	Has not been relayed and will not be relayed
1	May have been relayed, but will not be relayed
2 to 126	May have been relayed and can be relayed
127	Has not been relayed and can be relayed

Sequence Number & Source Address

The SEQ is a three-byte counter that the website keeps track of, it starts at 0 and counts up with each message being sent. When it reaches the maximum value of 16 777 216, it counts up the IVI and then restarts at 0. Updating the IVI is not relevant for this project as messages are rarely sent and will not reach the max value within a reasonable time. The SRC is a two-byte value that indicates which unit sends the message, this is also not relevant for this project as the device does not take into account from which unit the message is sent.

Obfuscate

The CTL, TTL, SEQ, and SRC are all obfuscated together according to the following algorithm.

```
Privacy Rand = (EncDST | EncTransportPDU | NetMIC)[0-6]
Privacy Plaintext = 0x0000000000 | IV Index | Privacy Rand
PECB = e(PrivacyKey, Privacy Plaintext)
ObfuscatedData = (CTL | TTL | SEQ | SRC) XOR PECB[0-6]
```

where EncDST, EncTransportPDU, and NetMIC are the encrypted destination, encrypted TransportPDU and NetMIC, and generated and explained in the following sections.

Obfuscating differs from encryption the message, as the name indicates it changes the message making it harder to understand the structure of the message. This is done to increase security and prevent attacks that analyze the structure of the message.

Destination Address

The DST is the two-byte unique unicast address of the device when communicating with a single device, given to the device during the provisioning process. When broadcasting a message to all devices within the network then the DST should be set to the fixed value of 0xFFFF.

Unsegmented TransportPDU

The TransportPDU will be explained in two sections, Unsegmented TransportPDU and Segmented TransportPDU, respectively. In Table 6.4 the format of the Transport PDU can be seen. The SEG indicates whether the message is segmented or not. If the message is segmented due to the message exceeding the MTU of 15 bytes, the format shown in Table 6.4 no longer applies, see the next section for the format of a segmented message. The AKF is a flag that indicates whether the message is encrypted using an application key or the device key, sent during the provisioning process, i.e., if the message is being sent to all devices within an application or to a single device. The goal of the Application Key-Binding message is to bind an application key to the device, i.e., it has not been shared yet, i.e., the message has to be encrypted using the device key. The AID is an application key identifier, which is set to 0b000000 if AKF is set to 0, i.e. if it is a message sent to a specific device, encrypted with the device key. The upper transport PDU will be explained in one of the following sections.

Table 6.4 Unsegmented TransportPDU Format

Field	Size [bits]	Description
SEG	1	Unsegmented = 0, Segmented = 1
AKF	1	Application Key Flag
AID	6	Application Key Identifier
Upper Transport PDU	40 to 120	The Upper Transport PDU

Segmented TransportPDU

The format for a segmented transportation PDU can be in Table 6.5. The first byte of a segmented message is the same as an unsegmented message, it consists of the SEG, which is set to 1, and AKF and AID as is set in the same way as an unsegmented message. The SZMIC is a flag that indicates the size NetMIC, which is set by CTL field earlier, where 0 = 32 bit and 1 = 64 bit. The SeqZero is the 13 least significant bits of the current sequence number. SegO is the current segment number, out of the SegN, the total number of segments (zero-based counting). Then Segment m is the actual segment of the payload being sent.

Table 6.5 Segmented Transport PDU

Field	Size [bits]	Description
SEG	1	Unsegmented = 0, Segmented = 1
AKF	1	Application Key Flag
AID	6	Application Key Identifier
SZMIC	1	Size of NetMIC
SeqZero	13	13 least significant bits of the sequence number
SegO	5	Which segment is being sent
SegN	5	Number of segment (zero-based)
Segment m	8 to 96	The segment of the upper transport PDU

Upper TransportPDU

The upper transport PDU consists of the unique opcode for the specific Zephyr service, e.g. the opcode of binding an application key is 0x00, and of the parameter. The parameter is the payload, and in the case of the application key binding, it contains the 16 byte application key. The opcode and parameter are then encrypted using AES CCM encryption, mentioned in Section 2.3, according to

```
data = opcode + parameter
encrypted = AES_CCM.encrypt(data, key, nonce)
```

with a nonce generated according to the following:

```
Nonce = NonceType + 0x00 + SEQ + SRC + DST + IVI
```

where NonceType = 0x01 or 0x02, depending on whether the message is encrypted using the application key or device key, respectively, as determined by the AKF value. For binding the application key, the device key is used to encrypt the message, as mentioned before.

Encryption

DST and the transport PDU are encrypted according to the following algorithm.

```
data = DST + TransportPdu
encrypted = AES_CCM.encrypt(data, netkey, nonce)
```

where the key is the network key shared during the provisioning process. The nonce is calculated according to:

```
nonce = 0x00 + CTL + TTL + SEQ + SRC + 0x0000 + IVI
```

This is done in addition to the encryption of the upper transport PDU, as an additional security measure. The variable encrypted will contain the encrypted destination, encrypted transport PDU, and the Network Message Integrity Check. These values are also used to obfuscate the CTL, TTL, SEQ, and SRC, as described in a previous section.

Network Message Integrity Check

NetMIC is a 32-bit or 64-bit value, generated during the encryption of the DST and transport PDU in the previous section, serving as an additional security feature. It corresponds to the last 32 or 64 bits of the encrypted data.

6.3 Code Example

Below is a code example for the function getHexSeg, it divides the upper transport PDU message into segments and attaches SEG, AKF, SZMIC, SeqZero, Seg0, and SegN, according to the description above.

```
getHexSeg() {
    let self = this;
    let message = this.upperTransportPdu.getHexSeg();
    let segments = [];
    if (this.#needSegmentation(message)) {
        segments = self.#splitHexString(message);
        this.#setSegmentation(segments);
        for (let i = 0; i < segments.length; i++) {
            segments[i] = this.#getFirstByte() +
            this.#getSecondByte(i) +segments[i];
        }
        return segments;
      } else {
        return this.#getFirstByte() +
```

```
                this.upperTransportPdu.getHex();
          }
        }

    #getFirstByte() {
        let firstByte = (this.seg << 7) |
            (this.akf << 6) | this.aid;
            return Utils.intToHex(firstByte);
        }
    #getSecondByte(seg0) {
        let secondByte = (this.szmic << 23) |
            (this.seqZero << 10) | (seg0 <<5) | this.segN;
        return Utils.intToNHexBytes(secondByte,3);
    }
```

7

Login and User Data

7.1 Overview

This chapter aims to describe the selection process of which single-sign-on (SSO) method to create and login users is the most appropriate to implement in this stage of development. Alongside the choice of SSO method, a database provider and the structure of the data need to be chosen. The choices made here are subject to change, as this primarily serves as a proof of concept. The level of knowledge about databases was limited in this project, which was taken into account when making decisions.

7.2 SSO

There are several ways to implement SSO. One common and well-established way is to use OAuth2 through Passport. OAuth2, is an industry-standard protocol that enables the use of SSO. Passport is an authentication middleware for Node.js and Express applications, that implements OAuth2. There are thorough guides and documentation available for the implementation of Passport, making an obvious choice. Passport offers several means to create and authenticate users, which simplifies the implementation of OAuth2 and thus an SSO user base. From the code example in Section 7.4, a Passport user system is implemented. The code example uses the Passports Google strategy, which creates a user based on their Google profile. Similar strategies exist for all major online profile carriers, e.g. Facebook, Apple, Twitter, and GitHub. It allows for the expansion of the user login system to include a large assortment of login options. As of writing this thesis, the only option for logging in is through the Google strategy but is intended to be expanded to include additional strategies in the future.

7.3 Database

After having the means to create a user and log in, provided by 'passport', that user needs to be saved into a database. Creating an in-house database system was deemed to be unfeasible. With the current level of knowledge of how to set up and operate a database, it would also have taken more time than this thesis allows for. Thus, a database provider needed to be chosen. An evaluation of commonly used databases was made in Table 7.1 with the criteria Price, Free storage, Scalability, Ease-of-use, and Online guides. The different criteria were weighted according to Table 7.2. Heavy weights were assigned to the ease-of-use and online guides, due to the limited amount of knowledge about databases.

Evaluation of different databases

Table 7.1 Database scoring

Database	Price	W.S.	Free storage	W.S.	Scalability	W.S.	Ease-of-use	W.S.	Guides	W.S.	Total	W.S.
MongoDB	5	1	3	0.3	4	0.4	4	1.2	5	1.5	21	4.4
Google Cloud SQL	2	0.4	1	0.1	4	0.4	4	1.2	4	1.2	15	3.3
Amazon RDS	3	0.6	1	0.1	4	0.4	4	1.2	3	0.9	15	3.2
Microsoft Azure SQL	3	0.6	1	0.1	4	0.4	4	1.2	4	1.2	16	3.5
MySQL	5	1	5	0.5	4	0.4	3	0.9	3	0.9	20	3.7
PostgreSQL	5	1	4	0.4	4	0.4	4	1.2	3	0.9	20	3.9

Table 7.2 Database criteria weight

Criteria	Weight
Price	0,2
Free storage	0,1
Scalability	0,1
Ease-of-use	0,3
Online guides	0,3

Chosen database and which data to collect

From Table 7.1 it is seen that MongoDB scores the highest, primarily due to their ease-of-use and the availability of online guides on how to implement the database. This was thus chosen as the best database to implement the user base with. Figure 7.1 displays the data associated with a user, including their unique ID (partly covered up) from the passport SSO, the sequence number, and a mesh network object. The mesh network object consists of a keychain with the network and application key, along with the user's devices stored in three separate maps: nodesByName, nodesByAddress, and nodesByDeviceID. Each device entry includes the unicast address, name, unique device ID, and device key. The ID from the SSO can not

be used to identify the individual behind the user, neither by the company nor an external party, and does thus not conflict with GDPR.

Figure 7.1 User example

7.4 Code Example

In this code example, it can be seen how 'passport' is used to create a new user from their Google profile, and in particular, how Google ID is used to create a unique user.

```
passport.use(new GoogleStrategy({
    clientID: GOOGLE_CLIENT_ID,
    clientSecret: GOOGLE_CLIENT_SECRET,
    callbackURL: "http://localhost:8080/auth/google/callback",
    passReqToCallback: true,
    userProfileURL: 'https://www.googleapis.com/
    oauth2/v3/userinfo',
}, function(request, accessToken, refreshToken, profile, done) {

    db.collection('users')
        .findOne({ id: profile.id })
        .then(existingUser => {
            if (existingUser) {
```

```
                console.log("Welcome back");
                // If a user with the same id exists, return it
                return done(null, profile);
            } else {
                // If a user with the same id does not exist,
                insert the new user
                let user = new User({
                    id: profile.id,
                    seq: String(0)
                });
                db.collection('users')
                    .insertOne(user)
                    .then(result => {
                        console.log("User added", result);
                        return done(null, profile);
                    })
                    .catch(err => {
                        return done(err);
                    })
            }
        })
        .catch(err => {
            return done(err);
        })
}));
```

8

Matter

8.1 Overview

This is a brief chapter on the new Matter [Nordic Semiconductors, 2022a], which is a new standard for home automation, released during the development of this project, and developed by, among others, Google, Apple, Amazon, and Nordic Semiconductors, introduced in Chapter 2.1. The Matter standard poses a threat to the viability of this product, due to having so many companies supporting and implementing the standard. Thus, the discussion of whether to switch to Matter or not was needed.

8.2 Matter technology

Matter combines three technologies for provisioning and communication, this is a quick summary of each:

Bluetooth. Bluetooth is used in the standard, however, not in the same way as this product. Matter only uses Bluetooth for connecting devices to the network, e.g., for provisioning, and not for communicating between devices. It implements a similar provisioning procedure for doing so, with the same level of security.

Wi-fi. Wi-fi is a widely used wireless standard for connecting devices to the internet. Matter devices that support Wi-Fi can be connected to a user's home network, which allows them to be controlled remotely using a smartphone app.

Thread. Thread is a wireless mesh network protocol that is designed for smart home devices. It provides a secure way for devices to communicate with each other, using the perks of having a mesh, in the same way BLE mesh does. Thread differs from BLE mesh in that Thread was designed for home automation and has better compatibility with IPv6. However, the range of Thread is much lower at a maximum range of 30m, compared to Bluetooth Mesh with a maximum range of 100m.

In summary Bluetooth is used for provisioning, Wi-fi is used for user-to-device communication, and thread is used for device-to-device communication.

8.3 Compatibility

The main chip which runs the Zephyr OS is manufactured by Nordic Semiconductors, which is one of the contributors to Matter and is making their products compatible with the standard. They are also offering extensive guides and documentation on how to implement the standard. Since provisioning is done over Bluetooth much of the code developed during this project would be reusable to some extent. However, the circuit board used for the devices in this project needs to be fully redesigned, due to the Matter devices using Thread and Wi-fi to communicate.

Besides technological compatibility issues, there are issues with conflicting company ideals. One of the core values of Adevo is to respect the user's privacy and limit data collection, something that can not be guaranteed when adhering to a standard developed with companies that do not necessarily share the same values.

To partake in the standard requires that the product passes a Matter certificate test, which adds complexity and cost to the product development.

8.4 Decision

The company has chosen not to switch to the Matter standard, due to incompatibilities with the technology the standard uses, as the company does not want to allocate the resources needed to redesign the circuit board, and, alongside it, the embedded programming. Also, the limitations of Thread's maximum range of 30m, make it not suitable for outdoor lighting. A final reason is the potential conflicting views of data management.

9

Results

9.1 Website design

The main website can be seen in Figure 9.1 and the login page can be seen in Figure 12.1 in the Appendix. The login page, as of writing this thesis, only features one way to sign in, that being through the user's Google account. The main website features a connect button to establish a connection with provisioned devices' proxy service. When searching for a device a pull-down window appears, seen in Figure 9.2, with all available Bluetooth devices, this is however filtered to only show Adevo devices. The website also includes three columns: one for controlling devices (sending on/off messages to all devices), one for adding new devices or controlling individual ones, and one for configuring devices, which is not yet properly implemented yet. The product has been given the preliminary name of 'Simply Light', displayed in the header with the company name and logo.

Figure 9.1 Main webpage

Figure 9.2 Web Bluetooth API connect to device

9.2 Provisioning

When adding a new device, the Web Bluetooth API searches for unprovisioned devices that are broadcasting the GATT provisioning service, the user then manually selects the device they wish to add to the network, and the API connects to the provisioning service and the messages in the provisioning process are automatically sent. The Web Bluetooth API also handles the replies containing the necessary information needed for provisioning from the device, e.g. the device's private key. When all messages in the provisioning process have been sent, the device stops broadcasting the provisioning service and starts to broadcast the proxy service.

The application key needs to be shared so that the device can access the network. This has to be done through the proxy service, and the Web Bluetooth API requires connecting to the proxy service. Unfortunately, due to safety measures implemented by the API, the user needs to manually connect to the proxy service. This is done by using the main connect button on the website. Thus, the solution was to mark the device with a flag and when connecting to that device the application key binding message is automatically sent. After that, the date and location are also sent to the device, which is used to determine the setting and rising of the sun, which is used to automatically turn the lights on and off, respectively.

Testing

The reliability of the provisioning was tested at four different locations, with minor changes made in between each test. Each test consisted of trying to provision a device ten times and noting the Bluetooth noise level, the average time to find the device, the success rate, and the cause of any errors. The result of the tests can be seen in Table 9.1, and will be discussed in the next chapter. The noise level is measured by looking at the number of available devices the Web Bluetooth API

detects when not filtering out all non-Adevo devices. The average time is measured as the time from clicking 'Add new device' to when the device pops up.

Table 9.1 Testing result

No.	Noise	Avg. Time [s]	Success rate [%]	Error sources
1.	High	9,7	60	*2x Gatt service disconnected *Replay messages *Invalid confirmation value
2.	Low	3,2	70	*2x Gatt service disconnected *Devicekey not received
3.	Low	2,1	80	*2x Gatt service disconnected
4.	High	7,2	70	*2x Gatt service disconnected *Invalid confirmation value

9.3 Login and userdata

The login and user system works as intended and fulfilled the goal set out by the system specifications. The amount of data collected from the user's Google profile is minimized, i.e., just the Google ID, and does not conflict with GDPR and the company's values, which was one of the primary objectives of the system specifications. The user creation and login process are also as seamless as possible, consisting of only two clicks, one click to log in/sign up, and then one to select the Google account used. If it is the first time that a Google account is used, a new user will be created. As previously mentioned, the only login option currently available is through Google. However, additional SSO alternatives will be incorporated into the same system, using Passport. While the inclusion of an off-grid alternative is of lower priority, it is planned as a future feature.

10

Discussion

10.1 Development process

The product has been in development for over two years, and I have been a part of it for the past one and a half years. My tasks in the project have varied widely during that time, and these last couple of months when this thesis has taken place, has been very educational. Parallel to this thesis, development in other parts of the product is being made, done by other parts of the team but to which I partly contribute. As previously stated, the devices are currently under development, and design flaws are being addressed. Additionally, a market analysis is being conducted to identify suitable market segments where the product can thrive and compete effectively. The development of the website and the product as a whole will not end with this thesis, nor will my role in the product. Most of my time was spent on the development of the provisioning process and application key message, as these were novel and the most technically intensive.

Methods used

As mentioned in Chapter 3.1 the project followed the Agile development process using kanban boards and standup meetings. The standup meetings were held twice a week according to plan. However, the usage of the kanban boards fell out of favor and were not updated regularly enough to fulfill their purpose When properly implementing kanban boards, one should divide tasks into much smaller parts and apply a time limit to perform that task. This was not done, instead, they were used as an overview of what the project needed to include. It proved to not be overly necessary to use kanban boards as the team consists of so few people and the intuition of being able to estimate a time limit for a given task was lacking by me.

The use of Ulrich's and Eppinger's product development methodology and the Design Science Paradigm could have been implemented more thoroughly, especially the DSP. The U&E method was primarily used to understand the functions of the system needed to accomplish, according to the customer, i.e., my industrial supervisors. The method was also used to evaluate the different databases in Table 7.1,

as doing concept evaluations is a primary part of the methodology. Doing a concept evaluation is always useful because it forces the developer to consider what criteria affect the choice of concept. The Design Science Paradigm shared a lot of the same principles as Agile and U&E, e.g., in identifying problems, designing solutions, and then validating those solutions. However, several visualization tools included in the paradigm were not utilized in this thesis. This was primarily because the paradigm remained unknown to me until a late stage in the design process, leaving insufficient time for proper implementation.

10.2 Final product

To discuss a final product in this thesis is somewhat challenging as the product is undergoing continuous development, even as I write this chapter improvements are being made. But the product, as of writing this, is fairly well functioning and is ready to be presented. There are, however, several flaws that need to be addressed.

Flaws

As seen in Table 9.1 in Section 9.2, the success rate for provisioning varies a lot and can be fairly low in high noise level areas. The average time to connect can also be frustratingly long in those areas as well. Resulting in a worsened user experience, there is also no proper error management implemented as of writing this, making it hard to know if the provisioning process was successful or not. Also seen in the table is that the most common error source is 'GATT service disconnected'. This could be solved by automatically reconnecting to the service, something that is unfortunately not implemented by the Web Bluetooth API. The API requires the user to manually select a device when connecting to a service as one of their safety measures. The possibility to reconnect to a device would also solve the issue of sending the application key, it needs to be sent to the proxy service and currently, the user needs to manually reconnect using the 'Connect to device' button on the website. If changes are not made by Google to the Web Bluetooth API, then perhaps the company needs to reconsider the use of a website. Much of the code generated during the thesis could be used in a mobile application, as the provisioning process and message structure are the same regardless of the type of application.

11

Future work & Conclusion

11.1 Future work

This project has developed a prototype website but requires additional functionality to meet the desired end product specifications. Future enhancements include:

- Implementing a toggle feature to control the usage of sun-related data, enabling users to choose whether to utilize the rising and setting of the sun.
- Incorporating an energy-saving time input to limit energy usage and reduce light pollution.
- Expanding the single sign-on (SSO) options for user authentication during login.

In addition to the website improvements, there are other aspects that need attention for the finalization of the project. These include:

- Ongoing modifications to the devices and their housing to ensure compatibility and functionality.
- Enhancing weatherproofing capabilities to withstand outdoor conditions.
- Addressing electromagnetic compatibility (EMC) considerations to meet CE marking requirements.
- Exploring new device options, such as Bluetooth mesh switches and light sensors, to expand the network's capabilities.

As the thesis nears completion, it is important to explore potential partnerships and outline the necessary steps to prepare the product for the market.

11.2 Conclusion

To conclude, this master's thesis has been greatly successful in achieving the goals it set out to achieve. The exception is modifying the website to a framework, allowing different lighting vendors to easily integrate their own brands. There have been market analyses done parallel to this thesis, and it is looking prosperous for the future of this project, a future I would gladly be a part of. Over the past two years, this project has been instrumental in utilizing and advancing the three fundamental aspects of mechatronics: electronics, mechanics, and programming. This thesis has significantly deepened my comprehension of protocols, web development, and programming in general, providing me with a unique level of expertise that few students have the opportunity to attain before entering the workforce.

Bibliography

Beaufort, F. (2023). *Communicating with bluetooth devices over javascript*. https://developer.chrome.com/articles/bluetooth/. Accessed: 2023-05-19.

Beck, K. et al (2001). *Agile manifesto*. https://agilemanifesto.org/. Accessed: 2023-05-13.

Ben Wolford (2020). *What is gdpr*. https://gdpr.eu/what-is-gdpr/. Accessed: 2023-05-17.

Bluetooth (2017a). *Introducing bluetooth mesh networking*. https://www.bluetooth.com/blog/introducing-bluetooth-mesh-networking/. Accessed: 2023-05-17.

Bluetooth (2017b). *Mesh profile*. https://www.bluetooth.com/specifications/specs/mesh-profile-1-0/. Accessed: 2023-04-24.

Bluetooth (2021). *Intro to bluetooth generic attribute profile (gatt)*. https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-gap-gatt/. Accessed: 2023-05-19.

devops (2021). *What's the range of a matter over thread device?* https://devops.com/thread-is-the-future-of-wireless-mesh/. Accessed: 2023-05-17.

Hofvander, M. (2022). *Design and Implementation of Bluetooth Mesh Outdoor Lighting*. MA thesis. Lund university, Lund, Sweden.

Jaap Haartsen (2018). *How we made bluetooth*. https://www.nature.com/articles/s41928-018-0186-x/. Accessed: 2023-05-17.

Jeanette Irekvist (2022). *Bluetooth: born in our backyard, raised by the world*. https://www.ericsson.com/en/blog/6/2022/ericsson-bluetooth/. Accessed: 2023-05-17.

Markets and Markets (2020). *Home automation system market with covid-19 impact analysis, by management, product, sofware & algorithm, and region (2020-2025)*. https://www.marketsandmarkets.com/Market-Reports/home-automation-control-systems-market-469.html. Accessed: 2023-03-16.

Nordic Semiconductors (2021). *Bluetooth low energy*. https://www.nordicsemi.com/Products/Bluetooth-Low-Energy/What-is-Bluetooth-Low-Energy?. Accessed: 2023-05-17.

Nordic Semiconductors (2022a). *Matter*. https://www.nordicsemi.com/Products/Matter. Accessed: 2023-05-17.

Nordic Semiconductors (2022b). *What is thread?* https://www.nordicsemi.com/Products/Thread/What-is-Thread?. Accessed: 2023-05-17.

Peyrott, S. (2023). *What is single sign-on authentication (sso) and how does it work?* https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/. Accessed: 2023-05-19.

Pierre Philip du Preez (2020). *Understanding ec diffie-hellman*. https://medium.com/swlh/understanding-ec-diffie-hellman-9c07be338d4a/. Accessed: 2023-05-17.

Runeson, P., E. Engström, and M.-A. Storey (2020). "The design science paradigm as a frame for empirical software engineering". In: pp. 127–147. ISBN: 978-3-030-32488-9. DOI: 10.1007/978-3-030-32489-6_5.

Semiconduct, N. (2020). *Introduction to the zephyr rtos*. https://webinars.nordicsemi.com/introduction-to-the-zephyr-rtos-3. Accessed: 2023-05-19.

Thread Group (2023). *What is thread?* https://www.threadgroup.org/thread-group. Accessed: 2023-05-17.

Ulrich S. & Eppinger, D. (2004). *Product design and development*. McGraw-Hill, cop.

Woolley, M. (2020). *Bluetooth mesh networking, an introduction for developers*. https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-networking-an-introduction-for-developers/. Accessed: 2023-05-17.

Zephyr (2023). *Generic attribute profile*. https://docs.zephyrproject.org/3.1.0/connectivity/bluetooth/api/gatt.html. Accessed: 2023-05-19.

12

Appendix

Website Design

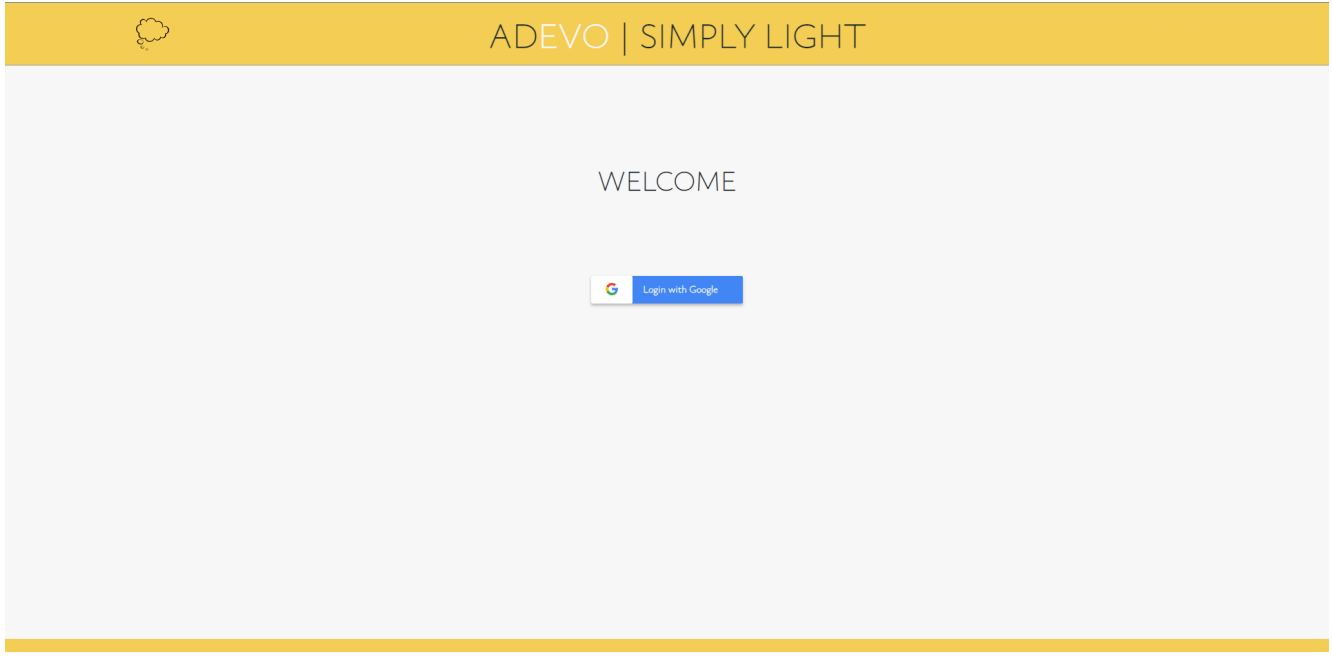

Figure 12.1 Login webpage

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2023

Document Number
TFRT-6215

Author(s)
Patrik Larsson

Supervisor
Mattias Wallinius, Adevo Consulting AB, Sweden
Maja Arvehammar, Adevo Consulting AB, Sweden
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden
Martina Maggio, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle

Designing and Implementing a Web Application for Bluetooth Mesh Device Provisioning and User Management

Abstract

As technology progresses, it seems to become increasingly inevitable for everyday objects in our homes to incorporate advanced features, to increase functionality, convenience, and energy management. This master's thesis presents the continuation of a Bluetooth lighting system, aiming to develop a comprehensive website solution for handling the provisioning and configuration of Bluetooth Mesh devices. Bluetooth Mesh is one of the latest features of Bluetooth, which connects devices into a mesh network, allowing messages to be received and relayed by each device, offering a reliable and scalable network that can cover large areas. This thesis focuses on implementing a user-friendly web-based platform that simplifies the setup and management of Bluetooth Mesh networks.

The website development process encompassed several stages, including systems specifications, system design, implementation, and evaluation. The project uses well-established web development frameworks, such as HTML5, CSS3, and JavaScript, to create an intuitive and responsive user interface. The website integrates with Bluetooth Mesh devices using standardized protocols, allowing for device discovery, provisioning, and configuration.

The devices in question are custom-built circuit boards that, through the website, are included in a mesh network and can, with either the rising or setting of the sun or the change of a simple switch, turn on and off an E27 lightbulb.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-63

Recipient's notes

Security classification

http://www.control.lth.se/publications/