UNIVERSITY OF TARTU

Institute of Computer Science

Computer Science Curriculum

Enrih Sinilaid

Monitoring and controlling smart home appliances using IoT devices

Bachelor's Thesis (9 ECTS)

Supervisor: Chinmaya Kumar Dehury

Tartu 2021

Monitoring and controlling smart home appliances using IoT devices

Abstract:

Usage of different smart home appliances and systems is becoming increasingly more popular in many households. There are many key points for rising number of adopters. For some it is the price as these systems and appliances are not that new to the market anymore and thus are more reasonably priced. Also, with the development in both hardware and software areas processing and memory units have become both faster and smaller which makes designing and developing different smart home devices more viable for building and selling commercially. This ensures that this smart home systems market is not dominated by few companies and gives a chance for anyone to try them out price wise. The second key point is the versatility of different smart home devices that are out in the market, ranging from lights to home automation and security. This variability lets people start out with few cheaper products like smart lighting or media devices and see if this is something for them.

Smart home appliances are devices that could be a common sight at many households like lights, speakers, TVs, air conditioners and so on but what makes them different is the built-in functionality for connecting to internet and then be monitored and controlled remotely. This ability to be remotely controlled and monitored makes it possible to develop automations that could further enhance the way these devices are used.

The aim of this thesis was to try and connect different smart home devices to one central system that could let the user of this system to control and monitor different smart home appliances and devices using their smartphones or computers. Additionally, this thesis aims to develop and showcase how to try and automate these devices from the central system. This automation will try to control the energy consumption of these smart devices by the user's location so that if user is not present then some devices could turn off or start using power saving profile. The central system will be hosted by IoT device running OpenHab OS.

Keywords:

OpenHab, system, smart home appliances, IoT, central system

**CERCS:**

Targa kodu seadmete seire ja juhtimine kasutades asjade interneti seadmeid

Lühikokkuvõte:

Erinevate nutikate kodumasinate ja -süsteemide kasutamine on muutumas paljudes majapidamistes üha populaarsemaks. On mitmeid faktoreid, miks nende kasutusele võtjate arv kasvab. Mõne jaoks on see hind, sest need seadmed pole enam turul uued, seega ka enamikel seadmetel on hinnad palju käepärasemaks muutunud. Samuti on nii riist- kui ka tarkvara arengu tõttu protsessorid ja mäluüksused muutunud kiiremaks kui ka väiksemaks, mis muudab erinevate nutikodu seadmete kujundamise ja arendamise elujõulisemaks äride jaoks See tagab, et vähesed suuremad ettevõtted ei domineeri nutikodu süsteemide turgu, andes võimaluse ka väiksematel ettevõtetel oma lahendusi luua ja müüa, mis annab kõigile võimaluse neid seadmeid hinnatarkalt proovida.

Teine faktor on mitmesuguste turul olevate nutikate koduseadmete mitmekülgsus, alates tuledest kuni koduautomaatika ja turvalisuseni välja. See varieeruvus võimaldab inimestel alustada mõne odavama tootega, näiteks nutivalgustuse või meediumiseadmetega, ja vaadata, kas see on midagi nende jaoks.

Nutikad kodumasinad on seadmed, mis võivad olla paljudes majapidamistes tavalised nähtused, näiteks valgustid, kõlarid, telerid, konditsioneerid ja nii edasi, kuid mis muudab need erinevaks on sisseehitatud funktsionaalsus interneti-ühenduse loomiseks ning seejärel lasta kasutajal neid kaugelt juhtida ja seirata. See kaugjuhtimise ja seiramise võimalus võimaldab välja töötada automaatika, mis võiks veelgi täiendada nende seadmete kasutamist.

Selle lõputöö eesmärk oli proovida ühendada erinevad nutikodu seadmed ühte kesksüsteemi, mis võimaldaks selle süsteemi kasutajal juhtida ja jälgida nutitelefonide või arvutite abil erinevaid nutikaid kodutehnikat ja -seadmeid. Lisaks on selle lõputöö eesmärk välja töötada ja tutvustada, kuidas neid seadmeid kesksüsteemi abil automatiseerida. See automaatika püüab nende nutiseadmete energiatarbimist kasutaja asukoha järgi kohandada, nii et kui kasutajat pole kohal, saaksid mõned seadmed välja lülituda või hakata energiasäästuprofiili kasutama. Keskne süsteem hoiustatakse IoT-seadmel, milles töötab OpenHab operatsioonisüsteem.

Märksõnad:

Kaugjuhtimine, seiramine, OpenHab, IoT, nutiseadmed, targa kodu süsteemid ja seadmed, kesksüsteem, operatsioonisüsteem

**CERCS:**

Table of Contents

[1. Introduction 6](#_Toc70493748)

[1.1 Aim of the thesis 7](#_Toc70493749)

[1.2 Outline of the thesis 7](#_Toc70493750)

[2. Background 8](#_Toc70493751)

[2.1 Idea 8](#_Toc70493752)

[2.2 Home Automation Platform 8](#_Toc70493753)

[2.2.1 Platforms 9](#_Toc70493754)

[2.2.1.1 Home Assistant 9](#_Toc70493755)

[2.2.1.2 OpenHAB 10](#_Toc70493756)

[2.2.1.3 Apple HomeKit 11](#_Toc70493757)

[2.2.2 Chosen Platform 11](#_Toc70493758)

[2.3 User tracking system 12](#_Toc70493759)

[2.3.1 Trackers hardware 13](#_Toc70493760)

[2.3.2 Trackers software 13](#_Toc70493761)

[2.4 Smart Home Devices 14](#_Toc70493762)

[3. Requirements 15](#_Toc70493763)

[3.1 Functional requirements 15](#_Toc70493764)

[3.2 Non-functional requirements 16](#_Toc70493765)

[4. System Architecture & Configuration 17](#_Toc70493766)

[4.1 System Overview 17](#_Toc70493767)

[4.2 Home Automation Platform Configuration 17](#_Toc70493768)

[4.2.1 System 17](#_Toc70493769)

[4.2.2 Initial Setup 18](#_Toc70493770)

[4.2.3 Configuration 19](#_Toc70493771)

[4.2.3.1 Configuring openHAB 20](#_Toc70493772)

[4.2.3.2 Adding Devices & Services 20](#_Toc70493773)

[4.2.3.3 Automation 22](#_Toc70493774)

[4.3 User Tracking System Configuration 23](#_Toc70493775)

[4.3.1 System 23](#_Toc70493776)

[4.3.2 Initial Setup 23](#_Toc70493777)

[4.3.3 User Tracking Script 23](#_Toc70493778)

[4.4 Notes 23](#_Toc70493779)

[5. Conclusion 24](#_Toc70493780)

[6. References 25](#_Toc70493781)

# Introduction

With the progress in various information technology fields, many aspects of human society have changed. Access to information has never been quicker and easier as devices capable of accessing information anywhere with internet coverage have become very common. Today many devices around us can receive, store, process and transfer data to connected devices . This system is known as the internet of things (IoT). Over the years, the popularity in this field has been increasing rapidly. According to the GSM Association, it is expected that the number of IoT devices will grow to 25.1 billion by 2025 [1].

One of many popular real-world applications for IoT is in smart home ecosystems. Through IoT, it becomes possible to monitor, control and automate smart home devices at some level, even for private consumers with no prior knowledge in this field. A large part of the popularity also stems from the fact that it is possible to create smart home systems with relatively more nominal cost than in the past. This is due to the development and the broader availability of older technologies. Adroid Market Research, a global market research firm, points out that smart home system's take-up is growing and that smart home systems' global market is expected to exceed USD 95 billion by 2025 [2].

A smart home is essentially a home where through wireless technologies, various appliances and devices can be monitored and controlled using mobiles or other networked devices. Data generated by sensors and smart home appliances make it possible to monitor conditions at home and its devices. Devices themselves can communicate with each other or with smart home central platform through WIFI and Bluetooth. This enables for smart home solutions where through IoT capabilities, more complex automations are possible.

Smart homes are not only more popular due to the capability and affordability of smart devices but also due to the fact that there are open-source smart home platforms. Some of these platforms are easy to implement and do not require huge investment. They can enable the user to tie in existing smart systems in the house to one central system. These platforms work by using wireless technologies and IoT capabilities to communicate with smart solutions from different ecosystems.

The flexibility of some open-source smart home platforms is not only limited to connecting together different smart systems in homes. With some research and time, users can try to build their own solutions on top of these platforms or implement advanced automations. This level of flexibility gives the users a certain degree of control of how their smart home systems handles information flow around the house. This enables users to have better control over their privacy and be sure that their private information will not reach any third party.

The degree of control over the private information gives certain open-source smart home platforms advantage over commercial smart home platforms. This is not the only advantage of choosing open-source platform for smart home. Open-source solutions often have better transparency than commercial solutions. This means that users could have better idea how open-source smart home platform works and how it works behind the scenes, thus possibly gaining better trust than their commercial counterparts.

Although smart home platforms importance in smart home is not low, it would be almost useless without any devices to control, monitor or automate. These devices can almost be anything in the house that could gather information, use information to accomplish some task or perform simple actions when triggered. These devices could be smart entertainment, climate, kitchen, security devices and the list does not end there.

With microcontrollers and single-board computers, even normal devices could act as smart devices. As an example, it is possible to monitor ordinary devices and appliances' energy usage and implement a switch for turning them off and on. This allows for creating a home system with efficient electricity usage and thus lowers the overall cost in electricity. Due to that, home owners could invest more into field of smart homes without having to worry about accumalating huge electricity bills.

## Aim of the thesis

The aim of this thesis is to create smart home system that could monitor and control smart home appliances through the use of IoT. To better demonstrate the IoT capabilities in the smart home system, a user tracking system will be developed. This system will collect users location data in real-time, which will be used to control devices around the user. This thesis uses an open-source home automation platform and various single-board computers to handle smart home devices and track users. As the tracking should differentiate between different users, users are located via their smartphones using either WIFI or Bluetooth. Communication between the home automation platform and user tracking devices is done by using MQTT messaging protocol.

## Outline of the thesis

This thesis consist of three stategs that are:

* Setting up home automation platform. In this stage an open-source platform OpenHAB is used for monitoring, controlling and automating various smart home devices across the house. As such this stage will consist of exploring OpenHAB functionalities and possibilities, setting up the OpenHAB, connecting devices into it and implementing automation and UI for controlling and monitoring smart home devices. This stage will also describe any problems or findings found.
* Developing user tracking system. This stage will cover how house user tracking system was developed. This will cover the research into different approaches and what approach was used in the end. As such there will also be steps taken in development phase and all risen problems will also be covered.
* Connecting home automation and user tracking system. In this stage the method of connecting the home automation and user tracking system will be described. Also this stage will also contain steps for creating automations using the data that user tracking system provides.

As for thesis structure it is as following:

* Background, section for information about the technologies and tools used in this thesis.
* System specification, section describing use cases and requirements for this system developd in this thesis.
* House automation system, section about setting up home automation platform, connecting smart home devices into it and automating said devices.
* User tracking system, section about developing system for tracking users in the house.
* Final assembely and automation, section about how location data is sent between systems and used in automating smart home devices.
* Summary, section describing work and suggestions for future research.

# Background

This section will give information about the system built in the thesis and introduce different components and technologies used in the implementation. This section will cover different open-source house automation platforms and explain why OpenHAB was used in the end. Next, there will be information about what tools and hardware will be used for the user tracking system. Finally, the smart home devices that are going to be used in this thesis will be introduced.

## Idea

This thesis aims to create an IoT-based smart home central system that could control, monitor, and automate different smart home devices in the house. This system will make use of a small single-board computer and open-source platform capabilities of good modularity, easy development and low cost. The user tracking system will be built with the purpose to gather data about users whereabouts in the house. This data will be used to automate smart devices around the house to respond when users are near them or have left their range. The main purpose of interaction between user location tracking and smart appliance control is to show what IoT-based smart home system is capable of.

Not only is the location of users monitored but also the smart home devices. The smart home system will use the data gathered from each smart home device and enable a manual control panel for house users to access. Said control panel would show each device's state, have an option to control these devices manually, and possibly show any data that is useful for the user, for example, time, date, the climate in and out of the house, energy consumption and more. This control panel will be able in both web and mobile application form.

An example of a possible scenario that should be possible with this system: Joe arrives at home. The system detects that and will turn on lights in the vestibule and cloakroom. Next, as it is a warm summer day, the system will turn the air conditioner on. As Joe reaches his room, the system will turn off lights in previous locations and turn on the Joes room lights. As Joe likes classical music, then the system will start playing a piece from Mozarts collection.

After some time, Joe leaves, and the system will turn off all the active devices and as indoor temperature is still relatively high, the air conditioner is set to run in an eco mode for more efficient power consumption.

Later, Joes wife reaches home and starts doing laundry. As she is working in the laundry room and going around the house collecting old bedsheets, the system automatically switching on and off lights in the rooms she is visiting. In Joe's room system only turns on lights and not the music as the room visitor is not Joe.

According to this example, the system should work with multiple users and should be able to differentiate between them to make customised automations possible. The following subsections will introduce the components and tools used for making this system possible.

## Home Automation Platform

A home automation platform is a system that is responsible for monitoring and controlling home devices like lightning, climate, entertainment systems, appliances and even home security. These devices are typically connected to a central hub that is managed by a house automation platform. There are many competing vendors for these platforms, and there is even several open-source systems. Home automation has a high potential to lead towards energy-efficient solutions that could positively impact the future [3].

### Platforms

For this thesis, the home automation platform must be able to handle multiple devices efficiently and have support for MQTT for simple and effective communication between devices and includes support for a large variety of smart home devices and appliances. These platforms should also have well-prepared documentation for understanding platforms' capabilities and limits and have an active user base and community for quickly getting answers to any specific question.

One other criterion for the home automation platform is to be IoT friendly and be efficient in its use of resources. As this smart home system will use a small single-board computer, the preferred platform would need to be optimised to run on it. Stability is also a significant factor, and there should not be any problem with storing and keeping safe the gathered data for a certain amount of time.

The following subsections will introduce few platforms that potentially cover the set criteria and could be used to build similar systems created in this thesis. All of the introduced platforms have their pros and cons. Based on these platforms, the one used in this thesis will be selected and further presented and discussed. One commercial platform is also introduced for better comparison.

#### Home Assistant

Home automation software Home Assistant [4] is a free and open-source platform for creating central control systems for smart home devices and appliances. This platform is designed around and written in Python, enabling Python's use in further custom development. Since November 2020, Home Automation has over 1700 modular add-ons that will allow it to use varios different smart home devices, services, and systems. [5]

Project Home Assistant started in September of 2013, and after few months in November, the core project was first published publicly on GitHub. The founder of this project is Paulus Schoutsen, and according to him, this project's goal was to be the platform for the home. Since the initial release, the project was more of a hobby project aimed at controlling Philips Hue lights [6]. Now, the Home Assistant community has risen in numbers, and in 2020 GitHub listed Home Assistant as second place in Python packages with the most active contributors.

As mentioned by the founder of Home Assistant, this project aimed to create smart home platform that could control everything from a central point [6]. His idea was to make smart home platform that focuses on smart home systems' usability and adaptivity. In his vision, the home automation system should not be something that would become cumbersome to use but instead be something that all people in the household would find helpful. This means that system should work flawlessly, blend with everyday workflow and should run at home. Overall the vision for Home Automation was to be able to develop smart home systems that would never get in the way or annoy but would be missed when not working. [7]

Home Assistant can be used in numerous ways. Recommended way would be to use their operating system in a dedicated system, for example, on Raspberry Pi. It is also available as a container based solution that, for example, could run on Docker. After initial configuration, the platform could already detect some smart home appliances on the home network. Other devices that were not automatically found can easily be added through UI. if there are integrations available to those devices.

Automation in Home Assistant can be done in multiple ways. The easiest way to implement automation is through UI using the Automation Editor feature. This builds automation that works by waiting for a trigger, then checking conditions and finally calling an action. For more advanced automating, Home Assistant has Templating feature that uses Jinja2 templating engine and syntax. Using this, users could benefit from a wider variety of operations and custom variables to build more indepth automations and system rules. Home Assistant is also capable of running python scripts. This means that users are not limited to use only the provided templating engine but could push their imagination to the limit and implement functionalities that are only limited by Python and hardware capabilities. [4]

#### OpenHAB

The open Home Automation Bus, or openHAB for short, is an open-source home automation platform designed to be a central piece for a smart home [8]. OpenHAB is entirely written in Java and is based on the Eclipse SmartHome framework. It is also very modular, and its base platform can be enhanced and extended through different add-ons. This allows for openHAB to be able to use different kinds of services or communicate with varios home automation solutions and devices. The openHAB supports close to 3000 various add-ons for both different services and devices as of writing this thesis.

OpenHAB project was initially released in 2010 by Kai Kreuzer and has since then become very popular. In 2013 openHAB core framework was contributed to Eclipse Foundation and became the Eclipse SmartHome project. This move allowed openHAB to truly become an open-source project with well protected and rigid intellectual property management. This enabled other companies and developers to use the openHAB core in their own solutions and ultimately helped with the openHAB community's growth and contributions. Based on the Eclipse SmartHome framework, new development branches of openHAB were created as versions 2.x and 3.x. [9]

The idea behind openHAB was to combat the situation where the user comes up with ideas and wishes on how to use different devices and systems but that were not supported out of the box or intended in their use cases. Thus this project was imagined as a central integration point between all of the different systems and services. As the vision behind this project was to consider users' wishes, handling information and privacy of users was also one of the key points. Considering that, openHAB gives their user the option to decide how they wish to control the data movement in and out of the local system. [10]

As openHAB was never intended to replace the existing solutions but rather enhance them, it is considered a system of systems. This means that all the sub-sub systems are to be configured and set up independently, leaving openHAB to only focus on these sub-systems' daily use. These sub-systems can further be broken down into items that are used in defining automation rules and UIs of the system. In openHAB, the notion of an item is a data-centred functional building block, and it does not matter whether this item is a device or some web service, which makes the concept of item abstract. This concept makes it easy to switch between different services and devices without changing the automation rules and UIs defined beforehand. [10]

OpenHAB automations are handled through a lightweight rule engine as such automations are referred to as rules in openHAB. These rules work using a trigger-based action activation model. These rules can be defined few different ways. One of the ways is to use the UI based simple rule creator. Their users can easily choose triggers and corresponding actions to be executed.

For more in-depth automation, users can write their own rule files. As the rule syntax is based on Xbase and is similar to Xtend, users can refer to Xtend documentation when writing rules. Also, the process of writing rules is made easier through openHAB VS Code Extension, which offers syntax checks, colouring and many more valuable features. The rule engine can also use JavaScript scripts which gives the users even more freedom when creating automations. [11]

#### Apple HomeKit

HomeKit is a communication protocol for smart home devices and appliances that are HomeKit enabled by manufactures. This software framework only works with applications that run on Apples operating system. These applications like Home allow users to configure, monitor, control and automate their smart devices. [12]

On 2014 Apple released iOS 8. Along with that they also released HomeKit. At releace the framework worked with third-party applications which enabled them to have interface for HomeKit devices. On 2016 Apple released their own official application called Home for managing all HomeKit enabled devices and made simple automations possible through that. Home application only worked with devices running iOS 10 and watchOS 3 but later on on 2018 this application was also released Apples computers, running macOS 10.14.

The main idea behind HomeKit was to simplify the tasks around the home. For that HomeKit was developed to act as a set of tools, enabling Apple and other people to build solutions for their needs. These solutions could allow for high level management over accessories and devices.

Although HomeKit could be used by users to develop their own complex smart home solution for monitoring, controlling or automating HomeKit enabled devices, it does have a minus over open-source platforms. That minus is that the number of supported devices is fairly little. As of 2019 only 450 devices were marked as compatible. This is due to MFI program, that manufactures have to enroll in, in order to have permission to add HomeKit capabilities to their devices. This means that only certain manufactures can develop devices that work with HomeKit.

Users are not only limited by devices that work with HomeKit. HomeKit systems only work with HomeKit communication protocols. This means that smart IoT systems that use other communication protocols can not be integrated into HomeKit powered solutions for management and automation.

### Chosen Platform

The requirements and capabilities expected from the home automation platform used in this thesis were numeros. From three solutions only openHAB and Home Assistant met those requirements. Both are essentially very similar in their capabilities with no noticeable limitation for neither of them. However, openHAB was picked to be the platform to be used in this thesis.

One of the key differences between the two platforms, that decided which platform was going to be used in this thesis, was documentation quality. The documentation for automation syntax and different add-ons were more in-depth and extensive in openHAB. Both platforms have also included examples of possible use-cases for better understanding, but openHAB seemed to have more comprehensive and overall more examples than Home Assistant.

As for other requirements, openHAB is configured to run on a myriad of different platforms such as Windows, macOS and Linux and even in container-based environments like Docker. It will work with almost any hardware in the range of fairly limited single-board computers to server computers. According to their documentation, the system will behave well with 2 GB of random access memory and 16 GB of low-speed storage using SD card. This means that Raspberry Pis 2 to 4 are good enough, where Pi 4 would give the best experience.

Due to openHABs modular design with close to 3000 add-ons and well-established support for MQTT, it is fair to say that the openHAB platform is very IoT friendly. Their add-ons enable using a myriad of different IoT devices and services, which lets the users design solutions for practically any IoT device. The well thought out MQTT support makes it possible for other IoT systems and devices to communicate effortlessly with openHAB.

## User tracking system

This thesis aims to show what IoT is capable of accomplishing in a smart home environment. This thesis will develop a user tracking system that could gather data about users whereabouts around the house and transmit that data through MQTT to the home automation platform. The user tracking system will essentially allow for more complex automations and smart home system control thanks to IoT capabilities.

User tracking systems can be implemented in a variety of different forms. One way to divide them would be a systems that either could or could not differentiate between users. Both systems have their use-cases, methodology and implementations. For a system that could not acknowledge one user from another, it is usually implemented to sense movement or existence of entity around the system. For this, these systems would use sensors that could detect movement or heat emission from a foreign entity. An example of where these systems are most often found is in security to detect intrusion.

The tracking systems that could differentiate between users in its detection range are usually implemented to keep track of users' whereabouts. This is done for a couple of reasons. One of them is for security. If it is possible to know where somebody is, then it is possible to seal off access to areas they are not privileged for. Many offices use this approach to keep the personnel to their specified areas.

The second use case is for controlling and automating the environment around the user. This is mainly implemented in smart home solutions for more complex, personalised and convenient automations. An example of a home automation system with this trackin system would be when one user could enter the bathroom and bath would be preheated to their preferred temperature.

For this system to differentiate between users and track their whereabouts, the system would need to read some kind of identification data from the user when they enter a new area. The data used for identification could be something that is always on the user, for example, biometrics data or an external data carrier. In the case of using biometrics, then the system could either implement face recognition through cameras or fingerprint scanners at each entrance.

Using external identity data carriers, the system could be implemented in various ways depending on the carrier. The carriers could share the data two ways, either in contact or near range. For contact range, the carriers are not active and only passively share data once they come in contact with some sort of reader. This system could be implemented through the usage of personalised keycards and card readers at each entrance. As for the near range carriers, they could be active and connect to nearby scanners to transmit identification data from range. This means that users would not need to take action at each entrance but simply pass by, and the system would record their whereabouts by itself.

For this thesis, the tracking system will be able to differentiate between users using the external identity data carriers that the system from a close range could read. This will make it possible to implement automations near the users seamlessly and be more convenient as the user does not need to perform an identification action at each entrance. The system will consist of multiple trackers, one per room, to automatically detect the user when they are near or in the room.

### Trackers hardware

This thesis's user tracking system will be a series of stand-alone devices capable of wireless technologies like WIFI and Bluetooth. These devices are going to be divided around the house, one device per room. WIFI will be required to send data from each device to the home automation platform.

Wireless technology will also be used to track users. In order to discover users, tracking devices in different rooms will be using Bluetooth to find user carried Bluetooth devices. By doing so, if a user carried Bluetooth device is discovered, for example, their phone, the tracker can check the signal strength from it to the device. After one iteration of scanning, the device will send data consisting of discovered devices and their signal strengths to it via MQTT.

In order to realise this idea, the stand-alone devices need to be programmable and capable of running continuously. There are multiple different hardware options that could be considered. Two main options are to use either Arduino single-board microcontrollers or Raspberry Pi single-board computers. Both options have their strengths and weaknesses.

Arduino is an open-source company that designs and manufactures single-board microcontrollers [13]. From a wide variety of controllers that they manufacture, some are capable of using WIFI and Bluetooth. In essence, Arduino microcontrollers are perfect for interacting with sensors and handling data in small quantities. Depending on the use case, they can be very affordable. The main downside of Arduino microcontrollers is that they are capable of primarily small singular tasks, and some controllers are not easy to source.

Raspberry Pis are developed by Raspberry Pi Foundation in the United Kingdoms. The Raspberry Pi is a single-board computer that is capable of running operating systems and graphical output. Their use-cases are more broad compared to Arduino microcontrollers as they are not that limited in hardware and are also affordable.

The tracking system stand-alone devices in this thesis make use of Raspberry Pi Zero W single-board computers. Although Arduino also has microcontrollers that would fit into the tracking system use-case, Raspberry Pi Zero Ws were more accessible. Also, these Raspberry Pis enabled more tinkering and researching into different ideas of discovering user.

### Trackers software

For trackers to work, they needed to have software capable of utilising the hardware. As the trackers are running Linux based Raspian Lite operating system, then there were multiple different ways to implement user tracking and discovery. One of the ways was to use some familiar programming language and its modules to implement the Bluetooth discovering and data transfer over MQTT. The other option was to find an already existing open-source solution and develop software on top of that.

After searching for possible open-source solutions, there was only one that suited the use case called reelyActive. reelyActive was founded in 2012 with the idea to create a cloud-based active RFID system. Over the years, their team and project have grown, and as of 2017, the solution is capable of running on Raspberry Pi and supports identifying Bluetooth devices.

reelyActive seemed to be an excellent open-source solution that could be used in implementing the software for trackers. But due to aged and almost nonexistent documentation and support for seemingly only Bluetooth Low Energy technology, this open-source solution was not used.

As there were no suitable open-source solutions to implement trackers software, the only way was to implement a solution from scratch. The user tracking system software will be implemented using Python, as it has support for Bluetooth and MQTT.

## Smart Home Devices

In this thesis, due to limited amount of resources, Philips Hue smart light bulbs will be used as smart home devices. These smart lights were used as they are the only smart devices available in thesis authors home. In total 3 Philips Hue ambient and color smart lights will be used.

The Philips Hue line of smart LED lights is manufactured by Signify N.V. This line of LED lamps can have colour changing, light level, and temperature control capabilities and come in various forms. Apart from more traditional bulbs, they can also be in light strip, outdoor lamp, floodlamp, and many other forms. [14]

What makes Philips Hue lights smart is the capability for them to be controlled wirelessly. For that, there are few ways. With newer lights, it is possible to connect to them over Bluetooth and manage them through smartphone applications or computers. The second way is to have Hue Bridge, a central controller for Philips Hue lights. The Hue Bridge can be connected to home internet through an ethernet cable, and then Hue Bridge will handle each light by itself. This means that users do not need to connect to each ligh separately and easily control them through the internet.

This thesis uses only these lights as they are capable of different automations and thus sufficient for demonstrating IoT capabilities in monitoring and controlling smart home devices.

# Requirements

This section will give overview of functional and non-functional requirements. These requirements will describe what was focused on in setup and development process of automation and user tracking systems. Described requirements are based on presented idea in section 2 and goals set with thesis supervisor.

## Functional requirements

This subsection will describe the functionalities that both smart home and tracking system will provide to users. Description will follow the schema of functionality and then description of functionality.

* The smart home system should show data about connected devices. This means that users should be able to see any meaningful information about the devices connected to the smart home system through use of user interface. For example, the data could be about the power state of devices or information collected by sensors.
* User should be able to manually control the devices. The devices that are connected to smart home system and can be controlled should be controllable by user. This means that user should be able to use user interface to control devices in any meaningful way.
* User should be able to add new devices to the smart home system easily. This means that smart home system should have functionality to easily add any supported device. For example this system could find new devices by looking through devices connected to network.
* The smart home system should allow writing automation scripts and rules. This means that smart home system used in this thesis should allow for writing custom automation scripts for devices connected to it.
* The smart home system should be able to store data to it. This means that the smart home system should enable for automatically or manually storing information to it or to the cloud. This is for saving states of devices for automation and monitor purposes and creating graphs using selected data collected over time.
* Guests locations should be included in location based automations in the smart home system. This means that smart home system could make use of information regarding guests location and enable generic automations based on their location.
* The smart home system should have software switches for user tracking automation. This means that it should be possible to manually turn off automations that make use of user location data. The switches could be toggeled from user interface by user.
* The user tracking system should be able to frequently share information with the smart home system. This means that the smart home system should have access to user location that is updated frequently. This enables for user location based automations that could trigger whenever user is near to a certain room or location.
* The user tracking system trackers should be plugged in and out without any problem. This means that users could unplug the trackers from the electricity and replug them without facing any problem. The trackers should automatically start scanning for users when turned on.
* The user tracking system should be able to track guests locations. This means that user tracking systems trackers would not only track the users but also any unknown user and their devices. This would enable for the smart home system automations to work with guest locations aswell.

## Non-functional requirements

This subsection will describe the non-functional requirements that act as a critia for both the smart home system and the user tracking system. Descriptions will follow the schema of non-functionality and then description of non-functionality.

* The smart home system user location automation should trigger almost immediately when users location changes. This means that there should not be large time difference between user changing their location and location based automations triggering.
* The smart home system should allow for multiple automations to trigger simutainosly. This means that system could handle multiple automations at the same time which would prevent tasks from pailing up.
* The smart home system user interface should work on mobile devices. This means that using mobile devices the smart home user interface could be accessed. This could be implemented trough an mobile or web application.

# System Architecture & Configuration

Section description paragraph TODO

## System Overview

The home automation system that is built in this thesis, consists of two subsystems, the home automation platform and the user tracking system. The home automation platform wil be the central system that handles the smart devices and services, enables automations and user interaction. The user tracking system consists of stand-alone tracker devices that are each responsibe for certain area in the home. This tracking system could be viewed as a service for the home automation platform. Essentially the tracking system is responsible for collecting user location data and sending that to the home automation platform for user location based automations.

These two sub-systems communicate through MQTT network. The network will consists of MQTT broker, hosted by the home automation platform, and MQTT clients on each tracker. Each MQTT client on trackers will be publishing collected data to the MQTT broker. The home automation platform will also have MQTT client that is subscribed to the broker. This enables the platform to collect sent data and use that information in automation. The described MQTT network is represented on Figure 1.

(Figure 1)

## Home Automation Platform Configuration

### System

The home automation platform will consists of two part, the software and hardware. The requirements for hardware is set by the softeware. For this thesis the software used is open-source home automation platform openHAB as described in previous section. Based on openHAB documentation the platform is suitable to run on various different systems and hardware. One of the hardware that is supported by the openHAB is Raspberry Pi line of single board computers.

This thesis uses Raspberry Pi 4 for containing and running openHAB platform. Raspberry Pi 4 has different available configurations on market and one used in thesis has the following hardware specifications:

* CPU – Broadcom BCM2711, Quad core Cortex-A72, 64-bit, 1.5 GHz
* RAM – 4GB LPDDR4-3200 SDRAM
* WIFI – 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless
* Bluetooth – Bluetooth 5.0 and BLE

As this single board computer does not have any built-in storage then storage is handled through Micro-SD card as the board has available slot for it. There are other features on the board like graphical output ports over micro-HDMI, Gigabit Ethernet port and USB 3.0 and 2.0 ports but these do not have any purpose in this thesis.

The Raspberry Pi 4 single board computer will be running Linux system setup by openHAB called openHABian. openHABian is based on standard Raspberry Pi OS Lite and designed as a headless system. It has plenty of features that makes setting openHAB 2 up easy. One of the features is that on the first boot, openaHABian will set up all the required tools and install the latest packages. This makes it easy for new users that have no prior experience with Linux based systems as they only need to provide ethernet access to Raspberry Pi and leave it to set itself up.

Other than that openHABian also comes with Linux packages and pre-installed settings that enable users to set up more advanced system configurations. Few of these packages are for setting up security measures, provide IoT friendly communication through MQTT, enabling databases and data visualization and provide secure remote connectivity through VPN. All of these packages and more can be enabled and configured through openHABian Configuration Tool. Through this tool user could additionally configure system setting, enable backuping and update openHAB and openHABian.

### Initial Setup

With hardware and software introduced in previous section the home automation platform can be set up.

The home automation platform setup on Raspberry Pi single board computer requires the following:

* Raspberry Pi 4
* openHABian image file
* Flash tool
* 16 GB Micro-SD card
* Micro-SD card reader
* Ethernet access

To get the openHAB platform running on Raspberry Pi 4 first the openHABian image file is needed. It can be acquired on openHAB project github page which can be accessed easily through documentation page about installation. There are multiple versions of openHABian images, but image used in this thesis is version 1.6.1, which has version 2 of openHAB called openHAB 2. This image was at the time of installation the newest available stable release. Since then there have been new realeases with updates to openHAB tools and packages, even new version of openHAB has been introduced which is openHAB 3.

After the image file has been acquired, it can be flashed onto a Micro-SD card. For flashing there are plenty of tools available, but one used in this thesis is balenaEtcher[[1]](#footnote-1) by balena. This tool is fairly easy to use as it only need few inputs from the user. Using this tool first image file was selected and then the destination. It should be noted that when using balenaEtcher, the openHABian image file should be unpackaged before flashing to a storage device.

Other requirement for flashing image on a Micro-SD card is to have a Micro-SD card with 16 GB of storage and a way for the computer, that is handling the flashing process, to access it for reading and writing purposes. The 16 GB of storage is not mandatory for the Micro-SD card as image does not take up large amount of space, but is recommended by the documentation. For computer to access Micro-SD card it needs to have a SD card reader that is compatible with Micro-SD cards. As the computer used for flashing purpose already had an inbuilt reader then no additional tool was necessary.

Next step after the flashing process has finished is to set up openHAB on Raspberry Pi 4. For this the Micro-SD card with flashed image needs to be installed on the Raspberry Pi 4, connected to the internet either through WIFI or Ethernet cable. Using the Ethernet cable is easier as connecting with WIFI involves modifying configuration file with WIFI SSID and password before the first boot. After the Micro-SD card is installed and a way for connecting to internet is provided then the single board computer could be booted for the first time.

After the Raspberry Pi 4 is booted for the first time it will take some time to set everything up automatically. This process length is entirely based on the internet connection, Micro-SD card write and read capabilities and the Raspberry Pi computers processing capabilities. In the installation documentation it is mentioned that this process could take around 15 to 45 minutes. It should be noted that Raspberry Pi 4 in this thesis took about 20 minutes to finish this process.

When the process is finished then openHAB web server is set up. Web servers user interface can be accessed on any computer on network through web browser with router designated IP address for Raspberry Pi 4 on port 8080 as the address. For example, when IP address for Raspberry device is set by router to be 192.168.1.3 then the user interface can be accessed on web browser at 192.168.1.3:8080.

### Configuration

This section will focus on configuring home automation platform openHAB. Through this process adding and connecting various devices and services to openHAB is possible. User interface will be implemented and configured to enable user to monitor and manually control added devices and services. This section will also describe the automation process and how user location data will be used.

There are a couple of ways to configure openHAB depending on what the user is trying to achive. If the user aims to configure the openHAB system like settings concerning updating, restoring from backups and creating backups or changing hardware features then openHABian has a tool called Configuration Tool. This can be accessed through connection to command line console on the system that is running the openHABian image. The described tool is represented on Figure 2

(Figure 2)

Other than that, the configuration can be aimed to add new devices, services, packages, implement user interfaces and add automation. This can be done in two different ways. First option is to use the openHAB web user interface. Through tool called Paper UI users could install packages and modules for adding various devices and services and configure them. This tool also allows to monitor and control added devices and services and add automations to them. The described tool and actions are represented on Figures 3, 4 and 5.

(Figure 3)

(Figure 4)

(Figure 5)

The second option is to modify configuration files on the openHAB system directly. Through this option users could do everything that was possible with Paper UI on openHAB web user interface. This option also allows users to write more advanced automations and develop JavaScript based scripts, thus making it more flexible than the first option. The downside of this option is that modifying and writing custom configuration files may be more difficult than using Paper UI tool and could require more indepht knowledge. The upside is that this option allows for more advanced system setups and configurations.

In this thesis the first option is used to discover and add the devices and second option is used for installing packages and implementing user interfaces and automation. This will be further explained in following subsections. The following subsections will also cover the process of configuring openHAB.

#### Configuring openHAB

After openHABian system has booted for the first time, the openHAB platform does not need any additional configuration to start functioning. Although it is not needed, some advanced features will not work without configuring the system. One of the advanced features is MQTT capability, that is by default disabled. As the system built in this thesis requires this functionality then it should be enabled.

To configure the openHAB the openHABian Configuration Tool was used. This tool can be accessed through command line console on the openHABian system. In order to access the openHABian system a SSH connection was established using software called PuTTY[[2]](#footnote-2), which is shown on figure 6. To open a PuTTY session in openHABian system the IP address of Raspberry Pi 4 and SSH port was required as the destination. The IP address was found using the network routers interface. As for the SSH port, it is by default 22.

(Figure 6)

After SSH connection is opened, PuTTY will display command line console. As the openHABian system is protected, then the console will request username and password, which is by default ‘openhabian’ for both. After successful login welcome screen is displayed. As the openHABian image is based on Linux kernel then the console can be operated with Linux commands. With command “sudo openhabian-config” the configuration tool can be accessed, as shown on figure 7 and 8.

(Figure 7)

(Figure 8)

In the tool menu the MQTT can be enabled under the “Optional Components” tab. The Mosquitto MQTT broker is used by the system built in this thesis and can be enabled bysellecting it in the menu. Additionally to MQTT WIFI was also enabled and that was done under the “System Settings” tab.

After that the tool can be exited and SSH connection closed. It should be noted that if enabling WIFI then system will be assigned new IP address. This means that in order to access web based openHAB user interface or when establishing SSH connecton then the new IP address should be used.

#### Adding Devices & Services

openHAB documentation states that, each device connected to openHAB is different and thus, needs base components in order to represent all of them [15]. The main base components responsible for adding and managing devices and services are the Add-ons, Things and Items. Each of these base components has it own function.

Add-ons are the base components, that are responsible for integrating support for devices and services, integrating external systems, handling data storage and transformation, extending automation engine and enabling voice features. Each add-on for device or service contains tools for openHAB to monitor, control and automate them. [16]

Things are the base components that represents all the entities like devices, services, etc., that are managed by the system. They are connected to the system through the add-ons, which enable the system to manage them. System can access the devices or services functionality through Channels that every corresponding Thing has. [17]

Item is the base component that can represent all the properties of the automation system. They can be strings, number, switches, sliders, colour pickers or other Item types. Item can be connected to a Thing to have control over its Channel. Items can be used in both automation and when defining user interfaces as they enable to interact with Things with corresponding devices and services, that they are connected to. [18]

In order to add a device to the system, first the add-on for that device needs to be installed. This can be done by either modifying the *addons.cfg* file by adding the add-on package name to bindings list, as shown on figure 9 or by installing through Paper UI, as shown on figure 10. If this process is done by modifying the *addons.cfg* file then the package name can be found on Add-ons page[[3]](#footnote-3) by searching for the device or service. After successful search the result page can be opened and the required package name is the last word in the url, as shown on figure 11, that is separated by slashes.

(Figure 9)

(Figure 10)

(Figure 11)

It should be noted, that if add-on was installed through Paper UI, then after a system restart these add-ons are removed and need to be added again. This is not the case when add-ons are installed by modifying add-ons configuration file and as such this method is used in this thesis for installing add-ons.

In this thesis, as the only smart devices, that we had access to, were Philips Hue smart lights, then the add-on was called Philips Hue Binding with package name “hue”. For services, we used add-on for MQTT with package name “mqtt”. It was added to bindings list in *addons.cfg* file.

After support for devices and services is enabled for openHAB through installing corresponding add-ons, then the next step is to add devices and services. This can be done by either specifying them in *default.things* Things file manually or by automatically searching for them in Paper UI. If done manually by modifying Things file for devices and services then it is adviced to follow instructions given by corresponding add-on on add-ons page.

In this thesis, the devices were added through the usage of Paper UI as this process seemed quicker and easier. To add the devices on the Paper UI, we first went to “Inbox” menu and clicked “Scan” button, as seen on the figure 12. After few moments devices started to appear. The Philips Hue lights, that we had, were connected to Hue Bridge. As such both the Bridge and each light were added by clicking on the blue circle with white checkmark and then on “ADD AS THING” button, as shown on figure 13 and 14.

(Figure 12)

(Figure 13)

(Figure 14)

The MQTT service was added by modifying *default.things* Things file and adding additional configuration file *mqtt.cfg*. This configuration file was for holding core parameters for MQTT service. The Things file was for defining MQTT service as entity so that openHAB could access its functionalities. For MQTT service a broker and a client were defined, where Channel for client was added for storing data from subscripted topic. Figures 15 and 16 show the contents of described files.

(Figure 15)

(Figure 16)

#### Automation

Acqording to openHAB documentation, automations in openHAB have their own base component known as Rules. Each rule works through lightweight rule engine and invokes a script when triggered. Rules can be defined either through Paper UI or by writing rule files, where file could hold multiple rules. Rule syntax is based on Xbase and follows structure of:

* Rule name – an unique name for each rule
* Trigger condition – an event, that triggers the rule execution
* Script block – a container for logic, that should be executed on trigger

For excecuting a rule, there are different categories of triggers:

* Item event based – The Item based triggers react to Item updates
* Thing event based – Thing based triggers react to device or service status changes
* Group event based – Group based triggers react to Item state changes, that are in certain group
* Time event passed – Time based triggers react at specified times
* System event based – The System based triggers react to system start events

Rules after triggering can do various things. They could change Item states, send command to Things, accomplish mathematical calculations and even trigger other scripts. [11]

The automation, that is implemented in this thesis is for controlling devices that are in the same room as the user. In essence, as the only smart devices, that we have for this thesis, are the Philips Hue smart lights, then if the users location is near a room, where a light is then light in that room will turn on. Similary if user leaves from the room, then the light in there turns off.

This automation is accomplished through a rule, that triggers whenever the MQTT client, that is subscribed to a certain topic, receives new data. This data is in json format, containing information about:

* which room this data came from
* Bluetooth signal strength between each users device and the tracker
* if any guest is in the room

This rule uses the data to determine if any user moved to another room by comparing the signal strengths between current room and last room. If the current room is different from the last room and the signal strength is better, then user is written into the new room. This rule also writes user out of any room, if their Bluetooth signal can not be determined.

After all the user locatons are determined, then rule will check if any rooms user count reached zero or became greater than zero. In case it reached zero, then Philips Hue lights in that room are turned off. But if user count changes from zero to greater value, then lights in that room are turned on.

Additionally there is a switch that disables or enables this rule, which can be manually controlled by users through user interface.

The described rule will be shown on figure 17.

(Figure 17)

## User Tracking System Configuration

This section will focus on user tracking system. The following subsections will cover the overview of the system, the setup process and finally explain the script used for user tracking.

### System

The user tracking system will consist of multiple individual trackers. Each tracker will be placed in different rooms and will independently track user whereabouts through Bluetooth scanning. This information is then sent to home automation platform openHAB through MQTT. For trackers to accomplish such actions, they need to have support for WIFI and Bluetooth. Other than that they also need to be able to run the script that scans for users, compiles the data and sends it over the MQTT.

As mentioned in Background section of this thesis, there are Arduino microcontrollers and Raspberry Pi single board computers available for such tasks. For this thesis Raspberry Pi Zero W will be used for each tracker. Main reasons for picking Raspberry over the Arduino was that they allow for better experimenting with different softwares and scripting languages and are easier to source.

According to the official Raspberry product specification [19], the used Raspberry Pi Zero W’s have following specifications:

* CPU – Single-core, 1 GHz
* RAM – 4GB LPDDR4-3200 SDRAM
* WIFI – 2.4 GHz 802.11 b/g/n wireless
* Bluetooth – Bluetooth 4.1 and BLE

Each Raspberry Pi Zero W will be running Linux based Raspberry Pi OS Lite that is a port of Debian. This operating system does not have a graphical output and works as headless system.

### Initial Setup

With hardware and operating system introduced in previous section, each tracker can be set up.

The user tracking system on each Raspberry Pi single board computer requires the following:

* Raspberry Pi Zero W
* Raspberry Pi OS Lite image
* Flash tool
* Micro-SD card with 4 GB or more storage
* Micro-SD card reader
* WIFI access

To get the Raspberry Pi OS Lite running on Raspberry Pi Zero W, first the image file is required. For this a tool like Raspberry Pi Imager [[4]](#footnote-4) can be used for acquireing the image file and flashing the image onto a Micro-SD card. As with

It can be acquired on openHAB project github page which can be accessed easily through documentation page about installation. There are multiple versions of openHABian images, but image used in this thesis is version 1.6.1, which has version 2 of openHAB called openHAB 2. This image was at the time of installation the newest available stable release. Since then there have been new realeases with updates to openHAB tools and packages, even new version of openHAB has been introduced which is openHAB 3.

After the image file has been acquired, it can be flashed onto a Micro-SD card. For flashing there are plenty of tools available, but one used in this thesis is balenaEtcher[[5]](#footnote-5) by balena. This tool is fairly easy to use as it only need few inputs from the user. Using this tool first image file was selected and then the destination. It should be noted that when using balenaEtcher, the openHABian image file should be unpackaged before flashing to a storage device.

Other requirement for flashing image on a Micro-SD card is to have a Micro-SD card with 16 GB of storage and a way for the computer, that is handling the flashing process, to access it for reading and writing purposes. The 16 GB of storage is not mandatory for the Micro-SD card as image does not take up large amount of space, but is recommended by the documentation. For computer to access Micro-SD card it needs to have a SD card reader that is compatible with Micro-SD cards. As the computer used for flashing purpose already had an inbuilt reader then no additional tool was necessary.

### User Tracking Script

## Notes

Section dedicated for describing problems that were faced when building a central "hub"

1. Problem with probably SD card (hub not booting correctly after shutting down)
2. Connectivity issue to wifi (hub not connecting via WIFI when connection to router lost sometimes)
3. Etc
4. Issues faced with Bluetooth(setting up PI0s and Bluetooth broadcasting by devices)

# Conclusion

# References

|  |  |
| --- | --- |
| [1] | Research and Markets, “IoT Middleware Market - Growth, Trends, and Forecasts (2020 - 2025),” Research and Markets, August 2020. [Online]. [Accessed 15 March 2021]. |
| [2] | Adroit Market Research, "Smart Home Automation Market," November 2020. [Online]. Available: https://www.adroitmarketresearch.com/industry-reports/smart-home-automation-market. [Accessed 10 December 2020]. |
| [3] | “Home automation,” 1 January 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Home\_automation&oldid=1004185639. [Accessed 3 March 2021]. |
| [4] | Wikipedia, “Home Assistant,” 30 March 2021. [Online]. Available: https://en.wikipedia.org/wiki/Home\_Assistant. [Accessed 31 March 2021]. |
| [5] | “​Home Assistant lets you automate your smart home without giving up privacy,” 10 May 2018. [Online]. Available: https://www.the-ambient.com/features/home-assistant-automation-privacy-582. [Accessed 31 March 2021]. |
| [6] | P. Schoutsen, “Perfect Home Automation,” 19 January 2016. [Online]. Available: https://www.home-assistant.io/blog/2016/01/19/perfect-home-automation/. [Accessed 31 March 2021]. |
| [7] | Home Assistant, “Documentation,” Home Asisstant, [Online]. Available: https://www.home-assistant.io/docs/. [Accessed 31 March 2021]. |
| [8] | openHAB Foundation, “Introduction,” openHAB Foundation, [Online]. Available: https://www.openhab.org/docs/. [Accessed 31 March 2021]. |
| [9] | K. Kreuzer, “openHAB 2.0 and Eclipse SmartHome,” 16 June 2014. [Online]. Available: http://www.kaikreuzer.de/2014/06/16/openhab-20-and-eclipse-smarthome/#ug. [Accessed 31 March 2021]. |
| [10] | openHAB Foundation, “Who We Are,” openHAB Foundation, [Online]. Available: https://www.openhab.org/about/who-we-are.html. [Accessed 31 March 2021]. |
| [11] | openHAB Foundation, “Textual Rules,” openHAB Foundation, [Online]. Available: https://www.openhab.org/docs/configuration/rules-dsl.html. [Accessed 31 March 2021]. |
| [12] | Wikipedia, “HomeKit,” 10 January 2021. [Online]. Available: https://en.wikipedia.org/wiki/HomeKit. [Accessed 17 April 2021]. |
| [13] | Wikipedia, “Arduino,” 17 March 2021. [Online]. Available: https://en.wikipedia.org/wiki/Arduino. [Accessed 11 April 2021]. |
| [14] | Wikipedia, “Philips Hue,” 7 April 2021. [Online]. Available: https://en.wikipedia.org/wiki/Philips\_Hue. [Accessed 11 April 2021]. |
| [15] | openHAB Foundation, “Configuration Overview,” openHAB Foundation, [Online]. Available: https://www.openhab.org/docs/configuration/. [Accessed 27 April 2021]. |
| [16] | openHAB Foundation, “Add-Ons,” openHAB Foundation, [Online]. Available: https://www.openhab.org/addons/. [Accessed 27 April 2021]. |
| [17] | openHAB Foundation, “Things,” openHAB Foundation, [Online]. Available: https://www.openhab.org/docs/configuration/things.html. [Accessed 27 April 2021]. |
| [18] | openHAB Foundation, “openHABian,” openHAB Foundation, 7 December 2020. [Online]. Available: https://www.openhab.org/docs/installation/openhabian.html#features. [Accessed 10 December 2020]. |
| [19] | RASPBERRY PI FOUNDATION, "Raspberry Pi 4 Tech Specs," RASPBERRY PI FOUNDATION, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/?resellerType=home. [Accessed 10 December 2020]. |

1. https://www.balena.io/etcher/ [↑](#footnote-ref-1)
2. https://www.putty.org/ [↑](#footnote-ref-2)
3. https://www.openhab.org/addons/ [↑](#footnote-ref-3)
4. https://www.raspberrypi.org/software/ [↑](#footnote-ref-4)
5. https://www.balena.io/etcher/ [↑](#footnote-ref-5)