Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Engineering -
Computer science 15.0 credits

MIDDLEWARE SERVICE FOR
CONNECTING DIFFERENT IOT
DATA MANAGEMENT SYSTEMS

Srdjan Scepanovic
ssc21005@student.mdu.se

Examiner: Sasikumar Punnekkat
Mälardalen University, Västerås, Sweden

Supervisors: Tijana Markovic, Miguel Leon Ortiz
Mälardalen University, Västerås, Sweden

June 30, 2023
Abstract

Internet of Things (IoT) is one of the most thriving areas in software engineering in the recent years. IoT is present in everyday activities as well as in various industries. IoT Data Management Systems (IoT DMSs) are one of the most important part of IoT as they process and control data in smart environments. This thesis work is focused on researching various IoT DMSs to gain more knowledge and get familiar with this topic. The main goal is to develop a software that acts as a middleware and supports multiple IoT DMSs and find out which techniques are suitable for finishing this task successfully. The literature review was conducted and multiple IoT DMSs were researched and their strength and weaknesses were identified. In addition to that, a software was developed in a form of web application with the capabilities to connect multiple IoT DMSs. This software supports communication and data processing between four different IoT DMSs. Smart room lab located on Mälardalens University was used and connected to this software in combination with all the devices and sensors that are connected in the Smart Room lab. The knowledge this research provides is useful to developers and users to use smart environment more efficiently. This area of software engineering (Internet of Things) has no limits and there are a lot of things that can be done to improve it and make the full use of it.
# Table of Contents

1. **Introduction**  
   1.1. Thesis overview ................................................. 2

2. **Background**  
   2.1. IoT data management systems (IoT DMSs) .............................. 3  
       2.1.1 open Home Automation Bus (openHAB) ............................ 3  
       2.1.2 Yggio ......................................................... 4  
       2.1.3 ThingSpeak ..................................................... 4  
       2.1.4 UbiDots ......................................................... 5  
   2.2. Smart room lab ..................................................... 6  
       2.2.1 Sensors in Smart room lab ...................................... 6  
       2.2.2 Smart outlets .................................................. 8  
       2.2.3 Devices in the Smart room lab .................................. 9

3. **Related Work** ..................................................... 10

4. **Problem Formulation** ............................................... 13  
   4.1. Limitations ......................................................... 13

5. **Method** ............................................................ 14

6. **Ethical and Societal Considerations** ................................ 16

7. **Description of the work** ............................................ 17  
   7.1. Environment setup .................................................. 17  
       7.1.1 Gathering data from Smart room lab ............................ 17  
       7.1.2 OpenHAB configuration ......................................... 18  
       7.1.3 Yggio configuration ............................................. 19  
       7.1.4 ThingSpeak configuration ...................................... 19  
       7.1.5 UbiDots configuration .......................................... 20  
   7.2. Middleware ........................................................ 21  
       7.2.1 Requirements .................................................... 22  
       7.2.2 Technical description .......................................... 23  
       7.2.3 Database Setup .................................................. 26  
       7.2.4 Connecting data from multiple IoT DMS ....................... 27  
   7.3. Testing ............................................................. 28

8. **Results** ............................................................ 30

9. **Discussion** .......................................................... 34  
   9.1. Research questions reflection .................................... 34  
   9.2. Threats to validity ................................................. 35  
   9.3. Room for improvement .............................................. 35

10. **Conclusions and future work** ..................................... 36

11. **Acknowledgments** ................................................... 37

References ............................................................... 39
List of Figures

1. openHAB dashboard .......................................................... 3
2. Yggio control panel .......................................................... 4
3. ThingSpeak panel for device management .............................. 5
4. Device representation in UbiDots ........................................ 5
5. UbiDots devices actions ..................................................... 5
6. Smart room lab ............................................................... 6
7. Motion sensor ................................................................. 7
8. Door/window sensor ........................................................ 7
9. Temperature sensor .......................................................... 8
10. Smart outlet ................................................................. 8
11. Method ............................................................................ 14
12. Smart room lab data sample .............................................. 17
13. Device representation in OpenHAB ...................................... 18
14. Detailed information about device in Yggio .......................... 19
15. ThingSpeak control panel .................................................. 20
16. UbiDots device configuration ............................................. 20
17. Architecture of the created software ................................... 21
18. User interface of created middleware .................................. 23
19. Communication between components ................................. 24
20. Created software’s sequence diagram .................................. 25
21. Database relational model .................................................. 26
22. Backend communication with IoT DMSs .............................. 28
23. Yggio overview .............................................................. 33
24. OpenHAB overview ......................................................... 33
25. ThingSpeak overview ....................................................... 33
26. UbiDots overview ........................................................... 33

List of Tables

1. Electric outlets and meter ID with corresponding devices. .......... 9
2. IoT DMSs general comparison .......................................... 30
3. Device Management Features Comparison ........................... 31
4. Detailed comparison of device management functionalities of IoT DMSs ........................................... 31
5. System Architecture Comparison ...................................... 31
6. Detailed architecture comparison for IoT DMSs ..................... 32
7. Interoperability Comparison .............................................. 32
8. Detailed interoperability comparison of IoT DMSs .................. 32
9. Ranking of important aspects of IoT DMSs ........................... 33
1. Introduction

Technology is taking over our world rapidly, especially in the last couple of decades. One of the most thriving areas is Internet of Things (IoT). IoT is growing quickly and is the main connection between the digital and physical world [1]. This span of technologies is providing services for devices to exchange data without human processing and supervision. IoT can be implemented in any area of business or industry [2]. The most often use of IoT is the smart environment. The most popular examples of this technology are smart cities, smart buildings, smart grids, smart rooms, etc.

Sensors are an essential part of the IoT. IoT connects different devices that are working together and exchanging data creating a smart environment. Due to the quick expansion of IoT technologies, security became one of the biggest concerns [3]. Certain actions should be taken so the security of the devices and the system is reliable. Many manufacturers do not put much attention to this problem because creating some complex safety software is expensive or not feasible. In that case, some type of home network protection is suggested [4].

One of the most important aspect of IoT is the systems that manage data and help process the data. These software are called IoT Data Management Systems (IoT DMSs). There are many different IoT DMSs and most of them have different structures and ways of processing and reading data. It is important to understand and know how to use these systems in order to implement the best match for the given project in a smart environment.

The purpose of this thesis work is to help developers process data easier and exchange data between different IoT DMSs. Software that acts as a middleware between the sensors and devices from the smart environment and different IoT DMSs is developed. This software is useful because it provides solutions for data processing due to its compatibility with different management systems. To make this work possible, a Smart room lab is used and the data is gathered and processed from there. The Smart room lab is located on the campus of Mälardalens University.

1.1. Thesis overview

To provide easier and more comfortable reading of the thesis work, the further text of this subsection explains how the thesis is organized.

- **Background (2)** - Describes basic knowledge that is necessary for better understanding of the whole thesis.
- **Related work (3)** - Presents related works and researches on the same or similar topics and highlights the similarities and differences between this work and other reviewed works.
- **Problem formulation (4)** - This section describes the purpose of this thesis work and its main goals as well as the proposed research questions.
- **Method (5)** - Shows and describes the work process and research methodology of this study.
- **Ethical and Societal Considerations (6)** - Considers ethical and societal concerns regarding this work and results, if there are any.
- **Description of work (7)** - This section describes what has been done to achieve results. In this case, implementation of software is described thoroughly.
- **Results (8)** - Shows and illustrates the results gathered for the proposed research questions.
- **Discussion (9)** - This section covers the discussion about the results, threads to validity and limitations.
- **Conclusion and future work (10)** - Here, the main conclusions about the work and research are drawn and presented.
2. Background

This section presents two essential components of this work: different IoT DMSs that will be considered and the Smart room lab that will be used including its components.

2.1. IoT data management systems (IoT DMSs)

IoT DMSs that are tested and thoroughly researched are OpenHab, Yggio, ThingSpeak and UbiDots. Further text talks about these IoT DMSs and their basic attributes and purposes.

2.1.1 open Home Automation Bus (openHAB)

openHAB\(^1\) is based on Java and allows the users to remotely access the connected devices in a smart environment and check the status of devices [5]. This is an open-source system and it can operate on any device that can run Java Virtual Machine (JVM), such as mobile phones, personal computers, etc [5]. openHAB takes raw data from the devices, such as power consumption values, whether the devices are turned off or on, etc. openHAB is a system that can work with multiple software and programming languages, such as Java, Python, C++.

The most important components of openHAB are [5]:

- Things - represent all physical devices/objects and services that are part of one system in a smart environment.
- Channels - provide the possibility for devices to interact and communicate with each other. Channels, alongside things, are part of the physical layer.
- Items - repository for all Things with their attributes.

openHAB is a powerful tool that provides much information about the devices and conditions in the lab, such as the location of connected devices, current and historical data, data visualization if it is necessary, etc. The user can use this platform both for looking at the data configuration and controlling it. Figure 1 shows the dashboard of openHAB.

![Figure 1: openHAB dashboard](https://www.openhab.org/)

---

\(^1\)https://www.openhab.org/
2.1.2 Yggio

Yggio\(^2\) is a cloud-based IoT management system produced by Sensative\(^3\). It is a horizontal IoT platform designed for smart cities, smart buildings, and smart environments in general [6]. It acts as a broker between devices, sensors and users, it also helps data processing and exchanging information in general. Using Yggio, the user can classify all the data from the smart environment into different categories. The user can see the devices’ status, location and all the values from different sensors that are available. It provides different views of data and values such as graphs and real-time camera views. Yggio also provides a very flexible data model that works based on the digital twins concept. In this concept, every device is represented by a digital twin that contains all the important information about the device [7]. It supports multiple data formats, such as JSON, XML, CSV and YAML [6]. There are several communication and data exchange protocols regarding this system, some of the most used are MQTT, CoAP, and HTTP [6]. Figure 2 shows Yggio’s control panel.

![Yggio control panel](https://sensative.com/yggio/)

Figure 2: Yggio control panel

2.1.3 ThingSpeak

ThingSpeak\(^4\) is a cloud-based IoT analytics platform created by MathWorks. Like the other IoT DMSs, ThingSpeak’s main features are data gathering and storage from physical devices [8]. ThingSpeak is unique due to the high number of functionalities regarding mathematical aspects, such as averages, medians and probabilities. All of this helps manage smart environments. For example, power consumption tracking is easier with these functionalities. Another unique feature of this IoT DMS is the possibility to analyze data of the device using MATLAB\(^5\). This IoT DMS is subscription-based, which means it is not open-source and free to use. There is, however, a free trial that can be used, but has limited features and possibilities. One of the main features of this system is the power to adapt various types of devices and smart environments, both small-scale properties and big industrial use cases [9]. The security aspect is also considered as the platform

\(^2\)https://sensative.com/yggio/
\(^3\)https://sensative.com/
\(^4\)https://thingspeak.com/
\(^5\)https://www.mathworks.com/products/matlab.html
uses secure HTTP/HTTPS communication. This platform also supports multiple data formats, which makes interoperability easier. Figure 3 shows the device management page of ThingSpeak.

![ThingSpeak panel for device management](image)

**Figure 3: ThingSpeak panel for device management**

2.1.4 UbiDots

Ubidots is a cloud-based IoT DMS developed for easy data management and configuration of smart environments. This system provides easy to use web user interface that enables individuals and groups to maintain multiple smart environments. UbiDots is not restricted to one group of devices, and that is what makes UbiDots a powerful tool. This system provides much information about the device, regarding its real-time data, historical data, and value changes [10]. UbiDots uses secure HTTPS communication protocol, as well as SSL/TLS encryption. It is subscription-based with a free trial for student projects with limited features. Figure 4 shows devices representation. Figure 5 shows actions that can be performed on device, in this case they are limited due to free trial account.

![Device representation in UbiDots](image)

**Figure 4: Device representation in UbiDots**

![UbiDots devices actions](image)

**Figure 5: UbiDots devices actions**

---

6https://stem.ubidots.com/app/devices/
2.2. Smart room lab

This lab was created for the purposes of a project funded by The Swedish Foundation for International Cooperation in Research and Higher Education (STINT). There are many devices installed in this lab such as microwave, fridge, oven, lights, etc. The lab contains a lot of sensors that are gathering and interpreting the data from the inside of the lab. Some of the sensors are: door sensor, movement sensor, temperature sensor, etc. These sensors are essential for the Smart room lab so those sensors must be interpreting data from the lab correctly. So, the values from these sensors will be used and gathered to test different IoT management systems. The real life smart environment is perfectly simulated. Figure 6 shows a photo taken from a camera in a Smart room lab on the campus of Mälardalens University, with the blue pins indicating where are the four main sensors located.

![Figure 6: Smart room lab](image)

2.2.1 Sensors in Smart room lab

There are multiple sensors in the Smart room lab at the university which provide data. This subsection presents some of the sensors and provides additional information about all of them. One of the most important components of a sensor are communication protocols and security protocols. All of the sensors mentioned below use the same communication and security protocol. Regarding communication, the sensors use the Z-Wave communication standard. As for security, all of the sensors use the S0 security standard. All of the sensors mentioned in this subsection are part of the Cleverio smart home collection. This collection is contained of devices that are built for the smart environment.

In the further text of this section, the main four sensors are mentioned and explained.

---

7https://www.stint.se/en/
1. Movement (Motion sensor) - It uses the battery 1xCR123A/CR17335 and has an access range of up to 70 meters whereas the detection range of this sensor is up to 9 meters. This detection range is enough for this specific lab. With the detection angle of 90 degrees (°), it is necessary to have multiple movement sensors in the lab so it covers the whole area. It is located on three locations in the Smart room lab. Figure 7 shows the motion sensor in the Smart room lab.

![Figure 7: Motion sensor](image)

2. Door/window sensor - The information this sensor provides is whether the doors/windows are opened or closed. This sensor uses a 2x AAA battery and has ultra-low standby power (< 5μA) which makes the battery last longer. It has an access range of up to 70 meters and a detection distance of over 15 millimeters. The working temperature of this sensor is from -10° to 50°. There is only one sensor of this kind installed in the Smart room lab and it is located on the entrance doors. Figure 8 shows the door/window sensor in the Smart room lab.

![Figure 8: Door/window sensor](image)
3. Temperature sensor - This sensor monitors the current temperature inside of a Smart room lab as well as the humidity percentage. It uses the 1x CR2450 battery and has a working temperature from -40° up to 120°. The range of temperature measurement is from -10° up to 60° and the maximum humidity percentage measurement is up to 95%. Figure 9 shows the temperature sensor in the Smart room lab.

![Temperature sensor](image)

Figure 9: Temperature sensor

### 2.2.2 Smart outlets

This Smart room lab uses smart outlets to connect and remotely access devices that are installed in the lab. It is remote-controlled socket with an energy meter. It enables users to remotely turn on/off the devices that are connected to this receiver. It can be used with most of the devices since its maximum load is 3000 watts (W), 16 amperes (A). Voltage supported is from 100 to 240 volts (V). When in stand-by, it uses less than 0.5W. Figure 10 shows how the kitchen light is connected with the smart outlet.

![Smart outlet](image)

Figure 10: Smart outlet
2.2.3 Devices in the Smart room lab

The information about the devices along with their data in the Smart room lab are necessary to fulfill the goals of this thesis work, such as their state, value and historical data. Table 1 presents all the devices that are currently installed and to which smart outlet they are connected in the Smart room lab. As it can be seen in the table, many smart outlets do not have a device connected to it. If there is any need, more devices can be installed and connected to the outlets that are not being used.

<table>
<thead>
<tr>
<th>Electric outlets and meter ID</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>Oven</td>
</tr>
<tr>
<td>25</td>
<td>Washing machine</td>
</tr>
<tr>
<td>26</td>
<td>Dishwasher</td>
</tr>
<tr>
<td>27</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>Microwave</td>
</tr>
<tr>
<td>29</td>
<td>Oven fan</td>
</tr>
<tr>
<td>30</td>
<td>Refrigerator</td>
</tr>
<tr>
<td>31</td>
<td>Drier machine</td>
</tr>
</tbody>
</table>

Table 1: Electric outlets and meter ID with corresponding devices.
3. Related Work

In this section, different related research are presented and explained and the main similarities and differences between the studied works and this research are mentioned.

In the work [3], the authors propose a model for the interoperability of devices in a smart environments. Authors mention that it is difficult to access the data of different IoT DMS due to different Application Programming Interfaces (APIs). The proposed platform is a "Semantic open IoT service platform". This platform provides the solution for the issues of connecting and exchanging data. To support the research, the authors developed an Android application to demonstrate advanced interoperability in a smart environment.

Interoperability between heterogeneous IoT DMSs in smart gas environment is introduced in [13]. The authors mention that traditional standard-based solutions do not fully support the level of interoperability that is necessary for this type of environment. Therefore, the authors propose a model-based engineering principle, more specifically "SmartHub", to enhance the interoperability. SmartHub system can gather data coming from various sources that use different communication protocols and data format standards. This model is still in the development stage. However, this system is restricted mainly for smart gas environment.

The research [14] introduces the "semantic interoperability layer" with its implementation in a smart city environment. The layer improves communication by having functionalities that do not depend on the specific syntax and format of the data. To operate on a semantic level, the data is interpreted with semantic representation by translating the file from non-RDF (Resource Description Framework) to RDF mapping syntax. RDF is a framework used for transforming data to the specific data format required by the system. Target service provides a template that can be provided in either JSON file format or XML depending on the content type the service is expecting. By implementing the "lifting and lowering operations" to semantic representation, the problems of integrating different services regarding different data formats are solved. The research is supported by experimenting with ticket booking systems in a smart city environments.

In the paper [15], the authors propose a new intelligent framework for interoperability in smart home environments. The framework is based on Simple Object Access Protocol (SOAP) technology that provides platform-independent interoperability between heterogeneous systems. To support this research, the authors implemented the framework with different home devices to prove the improvement in cooperation. The services and functionalities of this framework can be classified in three categories: 1. Services to elaborate data from heterogeneous sources, 2. Services to manage all data sources, 3. Services to enable communication among heterogeneous systems. The main components of this framework are the application interface, service stub, and database module and it is deployed using the Ethernet configuration of smart home environment. This framework also enables adding new devices when they are added physically.

Another paper that is researching interoperability in IoT and smart environments is [16]. The authors summarized the previous solution for interoperability in IoT and smart environments and pointed out the main issues and challenges of the proposed methods. This paper talks about interoperability as the most important and key aspect of IoT environments. Therefore, the list of challenges and issues must be addressed for creating future interoperability models according to the authors. The main issues that were pointed out were that often human interference is needed, privacy and security issues in smart cities.

In the [17], the proposed methodology for interoperability concerns is the use of IoT hubs in order to connect devices using web protocols. The authors used CKAN (Comprehensive Knowledge Archive Network) IoT DMS to integrate data, but also introduced the WoTKit. WoTKit is a web-centric toolkit focused on managing things that exhibit real-time behavior. This toolkit is run as a cloud service, its APIs offer an extensive set of IoT services making it easy to create web applications and other services for IoT. To achieve full interoperability between CKAN
Srdjan Scepanovic Smart Room

and WoTKit, the authors developed a certain API proxy architecture as a web application using Python. In this framework, the components of CKAN and WoTkit are represented as sub-classes that can communicate with each other.

The work [18], proposes a service-oriented, Quality of Service (QoS) aware integrated management methodology in a smart city environment. This approach enables the semantic interoperability between diverse IoT vertical deployments as well as the integrated management of different IoT systems and their resources. The authors propose VITAL-OS architecture. VITAL-OS enables the integration and management of multiple IoT services and data sets from numerous heterogeneous IoT systems. This architecture provides a set of data models and interfaces that enable connection and annotation from different IoT systems. Based on these data models and interfaces, the integrated management environment is created. This environment enables unified management of various IoT systems and datasets from a single entry point. The authors mention that it is relatively cheap solution and is compatible with small-scale smart environments, not just smart cities.

In the book [19], the authors talk about interoperability both between devices and IoT DMS. The authors propose multi-layer interoperability that enhances communication and data exchanging between heterogeneous IoT platforms. To achieve this multi-layer interoperability, INTER-IoT approach is introduced. INTER-IoT is based on hardware/software tools creating multi-layer interoperability among IoT systems layers (middlewares, databases, applications, semantics). This approach finds its implementation in various branches of IoT, health care, smart ecosystems, mobility infrastructures, etc. This approach has three main components that enable interoperability.

1. INTER-LAYER - The collection of methods and tools that provide interoperability among each layer of IoT platforms
2. INTER-FW - A global framework for programming and managing interoperable IoT management systems and IoT platforms.
3. INTER-METH - Engineering methodology based on Computer Aided Software Engineering (CASE) tool for systematically running the integration/connection between heterogeneous non-interoperable IoT platforms.

A middleware is a part of this approach and it connects multiple layers and each component of those layers as well as multiple IoT platforms. This approach provides long term solution that can be reused and applied on different IoT systems.

The authors of [20] talk about sensors and sensor values as the most vital part of the IoT system. Therefore, the accurate management of sensor values and data is crucial. The authors also propose a storage structure that integrates the relational database MySQL with the distributed database HBase. To reach this goal, SSM framework technology is used to enhance the development of this application. This approach improves IoT data management security and reliability. The most important component of this application is "Sensor data management". This function module enables users to search, query and delete the sensor values regardless of the sensor node types. The user can also edit the sensor values, type, storage location, etc. Another important function module is the "real time monitoring system" which provides a possibility to monitor data and sensor value changes in real-time. With the possibility of conducting data statistics and analysis, the authors conclude that this system fulfills all the needs of IoT platforms. Although, this approach is limited in the sense of interoperability between devices and sensors only.

Inovative hybrid approach to solve interoperability between IoT devices and IoT DMSs is presented in [21]. The idea is to introduce cognition into the IoT gateway. Therefore, the gateway acts as a cognitive dynamic system and provides holistic interoperability. The authors developed a cognitive IoT gateway framework. In this framework, each gateway is equipped with an adaptive radio for connecting with IoT devices. The gateways can register new devices that are added to the system and they can store information about each device in its database. The information stored is the ID of the device, along with annotated data from each device in a way that suits any IoT platform. The gateway senses the sensor data to decide what communication protocols
to instantiate, then sets up the communication protocol which suits the data format. The data is forwarded to an adaptive middleware layer, that filters it accordingly to the IoT management system.

A number of similarities and distinctions can be seen between the reviewed papers and this thesis work. The main focus of the majority of reviewed studies is interoperability in IoT. The papers primarily discuss interoperability between devices in smart environments. Most of the papers that talk about IoT DMSs interoperability is limited to specific smart environments or use cases. This thesis work is distinguished by achieving a higher level of interoperability between two or more IoT DMSs that use different data formats and communication protocols.
4. Problem Formulation

The goal of this thesis is to develop a software that will act as a middleware between the smart environments and different IoT DMSs. The goal is to make managing smart environments and their data easier and more efficient. To accomplish these goals, specific research must be conducted. The following research questions were formulated to help accomplish the mentioned goals:

**RQ1:** Which software solutions exist for IoT data management and what are their advantages and disadvantages from different aspects?
**RQ2:** Which techniques are suitable for developing a software that will act as a middleware in a smart environment that can support multiple/diverse IoT management systems?

These two research questions are heavily dependant, since only the researched IoT DMSs can be used in development phase for testing and connecting those systems to created software. The best result would be to have as much IoT DMSs as possible researched and tested. Diverse application should be created that can be used in various use cases. The created middleware is expected to have a possibility to use most of the connected IoT DMSs in order to prove its value and fulfil goal to make managing smart environments easier.

4.1. Limitations

Only a few IoT DMSs are free and those few are the ones that are being implemented and tested. The subscription-based ones are tested with a free trial or the free restricted version of the system. There is a limitation regarding literature review. It is impossible to read all available papers and get familiar with all IoT DMSs. This is also connected with development phase, since only IoT DMSs that are researched, are later tested and worked with. This thesis work is done in a specific time frame, thus, there can be some rushed decisions and mistakes due to not having time to test and research IoT DMSs enough.
5. Method

It is necessary to define a clear method process that to be followed throughout the entire research. This method process consists of 6 steps: Literature review, System installation, Design, Implementation, Testing and Results. The further text explains each of the steps individually. Figure 11 shows the method process.

1. Literature review - The research is conducted to get familiar with smart environments and different IoT DMSs. It is necessary to find as many different IoT DMSs as possible and find out how they work and the possibility of their interoperability with each other. To achieve this goal, different online databases are used to gather as much information as possible.

2. System installation - The plan is to work with the Smart room lab on MDU and use existing sensors and devices which are already built in the lab.

3. Design - In this phase, the design of the application is created. In this phase, the work is about architectural design, user interface design and design of the functionalities.

4. Implementation - The implementation part consists of programming and different solutions to integrate various IoT management systems and create middleware service that supports multiple systems.

5. Testing - Testing the created software with real data from the Smart room lab. In this phase, it is tested how each of the IoT DMSs works with implemented service. The specific test cases are created to determine whether the application works as it is supposed to. If any of the tests fail, the design phase is conducted again.

6. Results - In this part, the results of this work and research are noted and represented.

![Method Diagram](image-url)

Figure 11: Method
By following the process that is defined reaching the goals of this thesis work is possible. Since this is a continuous type of work and time-limited, the approach that was used for writing this thesis and developing a software is incremental. The progress of the work was reported to supervisors weekly and based on new knowledge and supervisor’s comments, modification to the thesis is done accordingly. The phases of development is split in one week/two weeks sprints, depending on the amount of work and complexity of the tasks. This model of development is practiced on many other projects on Mälardalens University and it is proven to be successful in time-limited projects like this thesis work.
6. Ethical and Societal Considerations

The Smart room lab is a closed private room with restricted access. A Camera that is installed will not be used. However, since security is a high concern in IoT, it is possible that a data leak from the software could cause some issues. Information such as if the Smart Room lab is empty or unattended can be gathered from this software in a certain way. More specifically, if movement sensor is not reporting any new value for a long time, similar with door/window sensor. Some sort of encryption of the data should be used in order to protect users personal data that can be gathered from a specific smart environments. Sensitive information about identity, location of the smart environment and device’s statuses should be protected.
7. Description of the work

This section presents detailed information about the implementation of software that connects different IoT DMSs.

7.1. Environment setup

Before developing a middleware that supports multiple IoT DMSs, it was necessary to set up the environment. This process is consisted of:

- Data extraction from the Smart room lab
- OpenHAB configuration
- Yggio configuration
- ThingSpeak configuration
- UbiDots configuration

7.1.1 Gathering data from Smart room lab

The Smart room lab has all the necessary equipment and hardware to export the data from the devices and sensors that are installed. It is important to mention that most of devices and sensors in the lab were already installed before this thesis work started. The gathering of data was accomplished by configuring the software of devices and setting up the communication protocols that support various IoT DMSs. The data is first stored in the database and then sent to the specific IoT DMS. For the purpose of this thesis work, the data from the lab is connected with four IoT DMSs: OpenHAB, Yggio, ThingSpeak and UbiDots. Figure 12 shows a data sample that is collected from the Smart room lab.

![Smart room lab data sample](image_url)

Figure 12: Smart room lab data sample
### 7.1.2 OpenHAB configuration

The next step of this process is to send data and configure OpenHAB for this specific use case. There are a couple of options to send the data to OpenHAB, the methods used for this work are sending data via API and through the user interface of OpenHAB. There are several steps involved in configuring this system.

- **Create devices** - The first step is to create devices through the user interface. This process requires the user to fill out typical fields such as: Name, Label, Type and Category. It is also possible to select the semantic class of the item, and what group it belongs to. However, these fields are not mandatory.

- **Set the channel to the item** - In this step, a user connects the actual device from the lab or a specific smart environment and set it as the channel for a specific item on OpenHAB. This way, the system knows what data it should represent for the given item. Users can always unlink the channel from a certain device.

- **Set how data is represented** - When the type of device is chosen, there are some default settings on how data is shown. This is not usually reliable and data representation can and should be set manually.

- **Send the data** - Parse data to the data format OpenHAB requires and send it to the item. As it was mentioned before, OpenHAB supports multiple data formats such as JSON, CSV, XML, etc.

After this process, the OpenHAB shows the data as it currently is in the Smart room lab. Figure 13 shows how the devices from the Smart room lab are represented in OpenHAB.

![Device representation in OpenHAB](image)

Figure 13: Device representation in OpenHAB
7.1.3 Yggio configuration

To the contrary of OpenHAB, Yggio is a cloud-based platform and uses different communication protocols. Configuring the Yggio's environment is also a process that requires several steps:

- **Add device(s)** - A user can add one device or number of devices through the control panel of Yggio. It has the same fields as the OpenHAB method of adding devices but some additional information is necessary, such as the "secret" field that has to be set so the device can be accessed through API or third-party application. Another parameter is the location of the device.

- **Set device group** - Yggio supports devices from multiple locations so the information about group of the device is necessary.

- **Set translators** - Translators provide data representation based on type of the device that was set in the creation process.

- **Send data** - The data is sent to the device via API with an adequate data format.

The data is found in the "Devices" section of the Yggio control panel. The user has to click on the device name and on the page of that device there are all the information and values about the selected device. Figure 14 shows information about the the device in Yggio.

![Figure 14: Detailed information about device in Yggio](image)

7.1.4 ThingSpeak configuration

Another IoT DMS that was tested is ThingSpeak. The configuration of ThingSpeak was necessary to connect it with the created middleware and other IoT DMSs. This is a subscription-based platform, so the first step was to create an account and select a student free version with limited features. To send real data from the devices, there is also a process that has to be followed:

- **Create channel** - In this platform, the channel acts as data storage for one specific device. To create a channel, user needs to fill in the required attributes such as name, description and set the specific fields which will receive various data from the device. This can be done through a web user interface.

- **Generate API key** - The user has to generate an API key because sending data to ThingSpeak requires authorization. ThingSpeak’s web interface has the option for generating API keys, both for the whole system and for the specific channel (device).

- **Send the data** - Modify the software of the device and send the data using either HTTP or MQTT communication protocols. Figure 15 shows the device from Smart room lab on ThingSpeak control panel.
7.1.5 UbiDots configuration

The final IoT DMS that was used in this thesis work is UbiDots. This is a subscription-based software, so it was used with a free version that provides limited features. After creating a free account, the configuration for receiving data from devices had to be done.

- Create device - On the UbiDots web user interface, the user creates a new device in the “Devices” section, sets the name and type of the device and it is ready to receive data from the actual device.

- Generate API key - To get the API key, the user navigates to the side menu and generates the API key in the “Authentication” section.

- Configure device - UbiDots cannot receive data straight from the device, it has to be either through some specific microcontroller such as Arduino or Raspberry Pi or through some other IoT DMS. In this thesis work, data is sent through other created middleware.

- Store data - When the data comes to UbiDots, it is stored in a specific device data storage in the system and the user can then access the functionalities UbiDots provides.

Figure 16 shows the device configuration on the UbiDots user interface.
7.2. Middleware

This subsection describes how the middleware that supports multiple IoT DMSs was developed. It provides information about the development phase: Gathering requirements (7.2.1), Technical description (7.2.2), Database setup (7.2.3), Connecting data from multiple IoT DMSs (7.2.4). Figure 17 shows the architecture of the project.

Figure 17: Architecture of the created software
7.2.1 Requirements

The requirements are one of the most important aspects of the development. They should be defined before the start of the development process. Functional and non-functional requirements are presented as well as the constraints for this work.

Functional requirements:

- Application must connect multiple IoT DMSs, OpenHAB, Yggio, ThingSpeak and Ubidots.
- The user can retrieve data from each of four IoT DMSs.
- The user can store the retrieved information gathered from IoT DMSs to the database.
- The user can send data from the application to IoT DMSs, modify and configure the IoT DMSs.
- The user can see historical data from each IoT DMS.
- The user can download specific data, such as, historical data and current data and store it in comma-separated values (csv) file.
- The user must have complete Create, Read, Update, Delete (CRUD) options.

Non-functional requirements:

- Performance - The application should process actions and respond to them in a given time frame.
- Scalability - The application should perform the same no matter the size of the data set or connected devices.
- Security - The application should ensure the privacy of the data along with the authorization.
- Usability - The application should be easy to use and not have a complex user interface.
- Maintainability - The application should be easy to improve or accept any changes.
- Reliability - The application should show correct values since it can cause many critical problems if reporting wrong values. Also, in a sense of being error-prone and not failing since IoT and smart environment management are high risk when it comes to crashing.
- The system must provide real-time data, to update and show the current data in the room every 100ms.
- Use of programming languages that support multiple data formats.

Constraints:

- Smart room lab firewall, which prevents gathering data when not in the room.
- Sensor types, all of the sensors are of the same type, which means if IoT DMSs does not support some sensor type, not a single sensor can be connected.
- Access to the Smart room lab, which is not consistent and reliable
- Time related constraint - The application must be built within the given time frame for this thesis work.
- Internet connection - The application can work only if the user is connected to internet.
- API dependant application - The application heavily relies on third party APIs and can not perform the actions if the given APIs resources are down.
7.2.2 Technical description

Middleware is developed in the form of a web application. The choice of the programming languages and frameworks was done based on previous knowledge and capabilities of those technologies. The backend of the application was done in PHP programming language using Laravel framework. Frontend application was done in Javascript using Quasar framework, which is part of Vue.js. XAMPP’s Mysql phpMyAdmin client is used for database configuration.

**Laravel**\(^8\) - A progressive PHP framework that is used for web development. It has many built-in options that help developers create and build projects. It is easy to configure and manipulate a database through the Laravel terminal. Some of the most used features of this framework are authentication, authorization, and permission management \(^{[22]}\). This framework works very well with multiple data formats and especially JSON data format and has multiple built-in options for JSON. Everything taken into consideration, Laravel provides powerful tools that are suitable for this thesis work.

**Quasar**\(^9\) - A Vue.js\(^{10}\) based javascript framework mostly used for building user interfaces of web, mobile and desktop applications \(^{[23]}\). Quasar also has many built-in features that help developers create reactive user interfaces. Using this framework, it is easy to represent data in many different forms, simply choose whatever is suitable for the specific case and implement it. Quasar does most of the Cascading Style Sheets (CSS) work as well. Figure 18 represents the frontend of the application created with Quasar.

![Figure 18: User interface of created middleware](image)

**PhpMyAdmin**\(^{11}\) - XAMPP’s interface that is used for the configuration of MySQL or MariaDB database server \(^{[24]}\). The web server of this client is Apache/2.4.54 (Win64) OpenSSL/1.1.1p PHP/8.2.0 and the database client version is libmysql - mysqli 8.2.0. In this case, it is used for the maintenance of the database of the application.

---

\(^{8}\)[https://laravel.com/]
\(^{9}\)[https://quasar.dev/]
\(^{10}\)[https://vuejs.org/]
\(^{11}\)[https://www.apachefriends.org/]

---

23
JavaScript Object Notation (JSON) - Data format that is mostly used in web development and acts as a communication tool between web applications and servers [25]. It is similar to the Extensible Markup Language (XML) notation and they can both serve the same purpose. JSON data is represented as a collection of objects and as a key-value pairs. It is easy to read for the human eye and there are many programming languages, such as PHP (Laravel) that can parse and read the data in JSON notation. The common thing about each IoT DMS used in this thesis work is their support for JSON format communication and data configuration. Therefore, the whole web application operates using JSON data format for the communication between different IoT DMS as well as the communication of the middleware itself with different platforms. Each of these IoT DMSs API also work based on JSON notation, more precisely, when targeting any API provided by IoT DMS, the response is returned in JSON data format.

Application Programming Interfaces (API) - APIs are fundamental components in the software engineering world. APIs are a set of protocols, routines or actions that are used to connect different software together or use the features of another software [26]. APIs can be either free to use or subscription-based. The APIs used in this work are free to use. There are different API categories, such as Web APIs (REST API) that uses HTTP communication protocol, operating system APIs, database APIs, etc. For the purpose of this thesis work, REST APIs are used due to the high compatibility with all four IoT DMSs that are connected through this middleware. Each IoT DMS provides a set of APIs that can be used in third-party software, for example, this middleware. The whole configuration, data gathering and sending as well as communication is achieved through REST APIs. This technology enables users to remotely modify external software and send the data both ways.

Hyper Text Transfer Protocol (HTTP) - A communication protocol used for data transfer over the internet. It is mostly used in web browsers and web servers [27]. HTTP is a request-response protocol, meaning that data is not sent until it is requested by some web entity. The request consists of a request line, headers and an optional body for messages. The request line is where the method is set, the methods are: GET, POST, PUT, DELETE. The response consists of status line, headers and message body. The a status line has the HTTP version and status code, for example, 200 - OK, 404 - NOT FOUND and a status message. There is a variation of this protocol and it is called HTTP secure protocol (HTTPS). This protocol uses encryption for data and makes data transfer more secure. HTTP is a communication protocol that is supported by OpenHAB, Yggio, ThingSpeak, UbiDots and also most of the other IoT DMSs. It was used in combination with APIs to clarify what was exactly needed in the response from IoT DMSs APIs. HTTP(s) was selected due to high compatibility with other technologies that are used for developing this middleware as well as compatibility with IoT DMSs.

This web application is dependent on the communication between these components which are mentioned in this subsection. Frontend communicates with the backend via HTTP requests and API routes and the backend is in direct contact with the database. Figure 19 shows how the components are connected to each other.

Figure 19: Communication between components
There are many other existing technologies that could be used for this thesis work, but due to a certain reasons those were not selected. CakePHP\(^{12}\) was considered for the backed, but due to Laravel’s more powerful documentation and previous experience, it was not selected. Standard Vue.js framework was considered, but it takes a lot of time to learn and develop a software. That is why Quasar was selected over Vue.js. Primarily, the technologies were selected based on compatibility with the IoT DMSs and IoT in general. More reasoning about the choice of technologies follows in the results and discussion sections. Figure 20 shows how the components work together and communicate with each other in the form of a sequence diagram.

![Project’s sequence diagram](image)

Figure 20: Created software’s sequence diagram

\(^{12}\)https://cakephp.org/
7.2.3 Database Setup

The core of every application is a database. For this specific project, it was necessary to create a database that stores information about devices from the Smart room lab and the IoT DMS that are being used. For this project, the MySQL database is used as it is compatible with the actions that are being performed by middleware. The database was created using Laravel. When creating a database, this framework creates several tables that are used for the built-in features. The key tables in this database are "devices" and "iot_dms" tables. Figure 17 represents the database relational model, the tables that are located in the red box are created by Laravel automatically.

The devices table consists of multiple columns that store information in a way that can be accessed by IoT DMS and the middleware itself. In this table, information about all the devices that are connected to middleware, including devices from all four IoT DMSs is stored.

Frontend of the application is using data from the database to represent information on the user interface. The frontend of the application is communicating with the database through Laravel API routes. The frontend application sends a request to the backend application through API and backend process the request and initiates the method that is set to that API route. Those methods are in charge of accessing the database and retrieving or storing data inside.
7.2.4 Connecting data from multiple IoT DMS

To satisfy the main requirements and the goal of this thesis work, it was necessary to configure the application to be able to connect data from multiple IoT DMSs. In this case, OpenHAB, Yggio, ThingsSpeak and UbiDots data have to be represented in a common user interface. To connect the devices from each of the IoT DMSs, it is very important to create a database that support the data format of all IoT DMSs and has fields that support all the information that is retrieved from the IoT DMS using APIs. To retrieve data from these IoT DMSs, APIs provided by all four IoT DMS official documentation are used. APIs from these IoT DMS provide different responses and therefore it was a challenge to configure the application to represent data about devices that are coming from different IoT DMS equally. Each of these platforms require authorization when sending requests through APIs. Every IoT DMS provides access tokens. An access token is similar to the password, the only difference is that it can expire and when generated through API, it is always different. Therefore, when targeting API through the application, an authorization header is necessary.

Since all of these IoT DMSs store and represent data differently, JSON files that are provided upon sending a request, provide different data formats and values. The main challenge of this process is to parse data accordingly and to store it in the database.

Retrieving data from Yggio - To connect and store data from Yggio, the "GET" method was used on an API route provided by Yggio Swagger Documentation. The first step before retrieving the data from Yggio is to generate an access token, by targeting authorization API using "POST" method and with parameters username and password.

Retrieving data from OpenHAB - Similarly to Yggio, OpenHAB has API documentation that help developers use data from the system. Retrieving data from OpenHAB is slightly different because OpenHAB is a locally hosted platform. So the request body is changed accordingly to the server OpenHAB is currently hosted on. For this case, OpenHAB is hosted on localhost:9000. So the route that is targeted with the GET method is: "http://localhost:9000/rest/items". OpenHAB does not require authorization for retrieving the data from the system.

Retrieving data from ThingsSpeak - To gather information about the devices and data from ThingsSpeak, it is also necessary to target an API route provided by ThingsSpeak API documentation. ThingsSpeak response is consisted either of a specific channel or a list of all channels that are connected to the platform, depending on the endpoint the user is targeting.

Retrieving data from UbiDots - To retrieve the data from UbiDots, it is necessary to create a Laravel method that initiates the GET API request to the UbiDots and handles the response it gets in return. To get the data from UbiDots, it is necessary to generate the API key because retrieving data from UbiDots requires an authorization header. When targeting an API route provided by UbiDots API documentation, the response is given in a JSON file and the devices are represented in an array of objects where each object represents a single device.

For each of the mentioned IoT DMSs above, after targeting the API for the data, it is necessary to create a Laravel method that will parse and handle the data accordingly. Depending on the JSON response from these IoT DMSs, the method either loops through array of objects or in some cases (OpenHAB and Yggio) through singular objects in JSON format, parses and saves the necessary values to the database. HTTPS protocol is used in each of these methods to assure secure communication.

13https://staging.yggio.net/swagger
14https://www.OpenHAB.org/docs/configuration/restdocs.html
15https://www.mathworks.com/help/thingspeak
16https://docs.ubidots.com/v1.6/reference/welcome
There are some key differences in the process of retrieving data from each of these systems. The main one is that for the retrieving of data from OpenHAB, the authorization header is not necessary. The second one is that responses from IoT DMS are not the same and do not have the same structure.

After storing the data from each IoT DMS to the database, the frontend is representing the data on the user interface based on the values in the database and completes the process of connecting data from multiple IoT DMS. Figure 22 shows the communication between the backend and the APIs provided by each IoT DMS.

![Diagram of backend communication with IoT DMSs](image)

Figure 22: Backend communication with IoT DMSs

### 7.3. Testing

Testing of this project is conducted based on BrowserStack\(^\text{17}\) testing checklist. This checklist is consisted of: Functional testing, Usability testing, Interface testing, Database testing, Performance testing, Compatibility testing and Security testing \([28]\).

- **Functional testing** - In this phase, the functionalities that the system is supposed to fulfill are tested. This is done based on defined requirements.
- **Usability testing** - In this part of testing, the user-friendliness of the application is tested. The authors of this checklist propose that it is essential that the application is easy to navigate in and that it has no complex user interface so that the most of the users can exploit full potential of the application. This testing was enhanced by other students which attend software engineering masters programme.
- **Interface testing** - In this phase, it is tested how the application is communicating with other components. Such as database, web server and in case of this project, IoT DMSs.

\(^\text{17}\)https://www.browserstack.com/guide/web-application-testing-checklist
• Database testing - Complete communication with the database is tested. Sending test queries to the database, document the response time and effectiveness of the queries. Test whether the response from the database is the same response as the actual data on the user interface as well as if the connection to the database is reliable.

• Performance testing - The authors of the checklist mention that testing speed, stability and scalability is one of the important aspects of testing. This can be measured by creating different data sets with diverse amount of sample and test whether the response time or efficiency is changed.

• Compatibility Testing - This phase is consisted of testing the application with different browsers and platforms. It is important to test this aspects since not all users use the same browser or operating system. This phase was tested on three different browsers, Mozilla Firefox\textsuperscript{18}, Google Chrome\textsuperscript{19} and Microsoft Edge\textsuperscript{20}.

• Security testing - The most important aspect in every area of software engineering, especially in IoT. In this part, the authorization and authentication is tested, it is important that no authorized users can perform any actions on the application. This was tested by targeting API resources from IoT DMSs without the authorization header or with old API key.

\textsuperscript{18}https://www.mozilla.org/en-US/firefox/new/
\textsuperscript{19}https://www.google.com/chrome/
\textsuperscript{20}https://www.microsoft.com/en-us/edge?form=MA13FJexp=e00
8. Results

This section provides information about the results that are gathered for both research questions and this thesis work in general. This section covers key aspects of this thesis work: identifying existing solutions for IoT data management and understanding their interoperability and the process of creating a middleware that makes interoperability between various IoT DMSs possible.

For the first research question, multiple IoT DMSs were researched and used to get familiar with their functionalities and capabilities. A lot of information was gathered by doing the literature review as well as the implementation part of the thesis. There are four IoT DMSs that were thoroughly researched and used:

- OpenHAB
- Yggio
- ThingSpeak
- UbiDots

This subsection highlights their similarities and differences regarding their functionalities, architecture and usability. By comparing these IoT DMSs, deeper knowledge is gained about existing solutions for data management in IoT and how these systems can be used in real-life use cases. This process also helped answer the second research question and develop a middleware that supports multiple IoT DMSs. Table 2 shows the general comparison between the four used IoT DMSs [29], [6], [5], [8]. As it can be seen in Table 2, these platforms serve a specific purpose and have their own advantages and disadvantages in specific fields.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>OpenHAB</th>
<th>Yggio</th>
<th>ThingSpeak</th>
<th>UbiDots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hosted</td>
<td>Locally-hosted</td>
<td>Cloud-based</td>
<td>Cloud-based</td>
<td>Cloud-based</td>
</tr>
<tr>
<td>Application area</td>
<td>Smart home auto-</td>
<td>Various indus-</td>
<td>Various indus-</td>
<td>Various indus-</td>
</tr>
<tr>
<td>support</td>
<td>mation devices</td>
<td>tries</td>
<td>tries</td>
<td>tries</td>
</tr>
<tr>
<td>Devices support</td>
<td>Home automatio-</td>
<td>Different de-</td>
<td>Different de-</td>
<td>Different de-</td>
</tr>
<tr>
<td>Data model</td>
<td>n home devices</td>
<td>devices in vari-</td>
<td>devices in vari-</td>
<td>devices in vari-</td>
</tr>
<tr>
<td>Communication protocols</td>
<td>Multiple com-</td>
<td>ous areas</td>
<td>ous areas</td>
<td>ous areas</td>
</tr>
<tr>
<td>Authentication</td>
<td>munication pro-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td>Requires deeper</td>
<td>Requires deeper</td>
<td>User-friendly</td>
<td>User-friendly</td>
</tr>
<tr>
<td>Programming languages</td>
<td>technical knowl-</td>
<td>technical knowl-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>availability</td>
<td>edge</td>
<td>edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming languages</td>
<td>Restricted to</td>
<td>Supports multi-</td>
<td>Supports multi-</td>
<td>Supports multi-</td>
</tr>
<tr>
<td>support</td>
<td>Java Virtual</td>
<td>ple program-</td>
<td>ple program-</td>
<td>ple program-</td>
</tr>
<tr>
<td></td>
<td>Machine (JVM)</td>
<td>ming languages</td>
<td>ming languages</td>
<td>ming languages</td>
</tr>
</tbody>
</table>

Table 2: IoT DMSs general comparison
These IoT DMSs all provide device management and data storage functionalities. In the next part of this subsection, the focus is on IoT DMSs device management functionalities. Table 3 summarizes IoT DMSs device management functionalities [5], [6]. As it can be seen in Table 3 each of these IoT DMSs provide different functionalities but the most of those functionalities are common for each IoT DMS. Table 4 provides more detailed information about the functionalities these IoT DMSs have.

<table>
<thead>
<tr>
<th>IoT DMS</th>
<th>Device Management Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yggio</td>
<td>Centralized management, multiple communication protocols, data formats, REST API, IoT platform integration</td>
</tr>
<tr>
<td>OpenHAB</td>
<td>Powerful home automation functionalities, device discovery, strong documentation, add-ons</td>
</tr>
<tr>
<td>ThingSpeak</td>
<td>Basic device management, data collection, REST API, MQTT communication protocol support, analytics</td>
</tr>
<tr>
<td>Ubidots</td>
<td>Simple user interface, MQTT/HTTP support, REST API, integrations, data visualization, narrowed device management</td>
</tr>
</tbody>
</table>

Table 3: Device Management Features Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Yggio</th>
<th>OpenHAB</th>
<th>ThingSpeak</th>
<th>Ubidots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Registration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Device Provisioning</td>
<td>Yes</td>
<td>Yes</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Device Configuration</td>
<td>Yes</td>
<td>Yes</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Device Control</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Device Grouping</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Real-time Device Monitoring</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Data Filtering and Processing</td>
<td>Yes</td>
<td>Yes</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Local &amp; Remote Connectivity</td>
<td>Both</td>
<td>Local</td>
<td>Remote</td>
<td>Remote</td>
</tr>
<tr>
<td>Security &amp; Authentication</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Device Management API</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Multi-protocol Support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Platform Integration</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4: Detailed comparison of device management functionalities of IoT DMSs

Another important and unique aspect of all researched IoT DMSs is the system architecture. Every platform has its own unique architecture. All of these systems operate on different principles and it is important to have that knowledge when deciding which IoT DMS is the most suitable for the specific use case. Table 5 summarizes IoT DMSs architectures and Table 6 provides more detailed information about architectural features of IoT DMSs.

<table>
<thead>
<tr>
<th>Platform</th>
<th>System Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yggio</td>
<td>Microservices architecture, API gateway, authentication, IoT device management service, data processing</td>
</tr>
<tr>
<td>OpenHAB</td>
<td>Modular architecture, runtime core, device integration</td>
</tr>
<tr>
<td>ThingSpeak</td>
<td>Cloud-based architecture, REST APIs, MQTT support, data analytics and visualization</td>
</tr>
<tr>
<td>Ubidots</td>
<td>Cloud-based architecture, REST API, MQTT/HTTP support, integrations, data visualization</td>
</tr>
</tbody>
</table>

Table 5: System Architecture Comparison
Table 6: Detailed architecture comparison for IoT DMSs

The most important aspect of IoT DMSs regarding this thesis work is interoperability due to the middleware that had to be created. In the further text, an analysis based on the interoperability level of IoT DMSs is presented. Table 7 summarizes interoperability characteristics of each IoT DMS and Table 8 provides more detailed information about IoT DMSs interoperability features.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Interoperability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yggio</td>
<td>High interoperability, support for multiple communication protocols and data</td>
</tr>
<tr>
<td></td>
<td>formats, integration with various IoT platforms</td>
</tr>
<tr>
<td>OpenHAB</td>
<td>High interoperability in home automation, modular architecture, extensive library</td>
</tr>
<tr>
<td></td>
<td>of bindings, community-driven development</td>
</tr>
<tr>
<td>ThingSpeak</td>
<td>Limited interoperability, supports RESTful APIs and MQTT for data ingestion,</td>
</tr>
<tr>
<td></td>
<td>focus on data storage and processing</td>
</tr>
<tr>
<td>Ubidots</td>
<td>Good interoperability, support for MQTT, HTTP, RESTful APIs, built-in integrations</td>
</tr>
<tr>
<td></td>
<td>with IoT platforms</td>
</tr>
</tbody>
</table>

Table 7: Interoperability Comparison

Table 8: Detailed interoperability comparison of IoT DMSs

The further text summarizes and visualizes the strengths and weaknesses of each reviewed IoT DMS. The graphs are generated based on the aspects presented in the previous section. The most important aspects of IoT DMSs for this thesis are highlighted and ranked 1-5. The ranking is conducted based on the criteria in Table 9. Figures 24, 23, 25, 26 show an overview of IoT DMS strengths and weaknesses.
<table>
<thead>
<tr>
<th>Aspect</th>
<th>Criteria</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devices support</td>
<td>Small number</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Most devices</td>
<td>5</td>
</tr>
<tr>
<td>Application area</td>
<td>Specific use cases</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Any area</td>
<td>5</td>
</tr>
<tr>
<td>Data model diversity</td>
<td>One format</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Multiple formats</td>
<td>5</td>
</tr>
<tr>
<td>Communication protocol diversity</td>
<td>One protocol</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Most protocols</td>
<td>5</td>
</tr>
<tr>
<td>Availability</td>
<td>Subscription</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Free to use</td>
<td>5</td>
</tr>
<tr>
<td>Device management options</td>
<td>Poor options</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Powerful tools</td>
<td>5</td>
</tr>
<tr>
<td>Ease of use</td>
<td>Complex UI</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Simple UI</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 9: Ranking of important aspects of IoT DMSs
9. Discussion

This section reflects on the results gathered throughout this research work, also, contains information about the constraints and limitations, what could have gone better and threats to validity.

9.1. Research questions reflection

In this thesis work, different IoT data management systems and software were investigated and researched, with the main focus on their interoperability and the web application development that acts as a middleware.

**RQ1 -** Which software solutions exist for IoT data management and what are their strength and weaknesses from different aspects?

This research provided key insight into four different IoT DMSs, OpenHAB, Yggio, ThingSpeak and Ubidots. After this research, it can be identified that each IoT DMS has its own strength and weaknesses, as well as areas where they can be implemented in full power. These four IoT DMSs were evaluated on different categories: device support, application area, data models, communication protocols, availability, device management functionalities, usability, architecture, interoperability level. It was concluded that OpenHAB stands out with its flexibility, but requires deeper technical knowledge. Yggio is cloud-based platform that has powerful device management options with wide area of implementation. ThingSpeak has the best visualization and analysis tools out of four, but lacks device management options. UbiDots has the simplest user interface with wide area of implementation, but is not as flexible as the other IoT DMS.

This knowledge is useful to users and developers and it helps making decisions on whether to use a certain IoT DMS in a specific use case. This information also helps when creating a middleware like the one that is created in this thesis work. Without knowing the system architecture, how they manage data and devices and the interoperability level, it would be impossible to create such a middleware. All four researched IoT DMSs have their specific purpose and application, but also many common things that can be used to create software that will enable interoperability between them.

**RQ2 -** Which techniques are suitable for developing middleware service in a smart environment that can support multiple/diverse IoT management systems?

For the middleware that supports multiple IoT DMSs, a web application was created. This web application supports two-way communication between itself, the Smart room lab and multiple IoT DMSs. This web application enables users to easily connect devices to different IoT DMSs through its interface. The user can create devices on other platforms and send the data through the application and afterward use the functionalities of each IoT DMS without previously setting the communication directly through the device. This middleware design is quite different than other software solutions that were reviewed due to its high level of interoperability between IoT DMSs, not just devices in IoT smart environments. It simplifies the usage of and control of smart environments and helps users that are unrelated to software engineering to use different IoT DMSs functionalities. The essential techniques:

- **Laravel (PHP framework) for the backend** - Selected due to its various built-in options to manipulate with database and its compatibility with JSON data format that is used for the communication of IoT DMSs with each other and with the middleware.

- **Quasar (Javascript Vue framework) used for the frontend of the application** - Selected due to the built-in options that enable developers to build user interface in a short time frame, which was necessary because this thesis work is time-restricted. Works well with Laravel and JSON data format.

- **PhpMyAdmin used for database configuration** - Selected because of its simple user interface and straight connection with Laravel.

- **JavaScript Object Notation (JSON) for data gathering** - Selected because it is supported by each of four investigated IoT DMSs so the interoperability between them is achieved easier.
• REST Application Programming Interfaces (REST APIs) for sending and retrieving data from IoT DMSs - Selected because each of four IoT DMSs provide detailed REST API documentation for developers to use and interact with those systems.

• Hyper Text Transfer Protocol (HTTP(S)) in combination with API for communication between middleware and IoT DMSs - Selected due to its compatibility with all four IoT DMSs and its security. The security is high concern in IoT and this communication protocol provides that.

In general, these technologies work well with each other and that helps developers in the implementation process. Another important aspect is that most of the IoT DMSs support these technologies thus creating interoperability between IoT DMSs is significantly easier.

9.2. Threats to validity

This subsection pays close attention to threats to the reliability of the results this research provided. Further text lists the threats to the validity of this research.

• Selection bias - The analysis of different IoT DMSs may not be completely reliable due to the fact that selected IoT DMSs are not representatives of all IoT data management solutions. This has been reduced by careful selection of IoT DMSs with different architectures and area of implementation.

• Generalization - The finding of this study may not be reliable for every use case because it is done considering specific requirements with specific constraints. This was reduced by creating functionalities that are independent of the smart environment hardware.

• Construct validity - This study was impacted by the constraints and limitations related to the Smart room lab and multiple IoT DMSs. Certain actions are taken to mitigate this and it can also be referred to as a future work problem.

• Time-related threats - Due to IoT technology’s high rise and usage in the past years, it can be expected even higher expansion of this technology and the changes that happen in the upcoming year may affect the results of this research.

9.3. Room for improvement

Due to the high number of specific limitations and constraints, there is room for improvements in this research. As expected, the limitation in the form of limited access to the Smart room lab proved as one of the highest. Generating data sets and retrieving data from the lab was bounded to certain time slots, this also impacted the testing phase. In the combination with Smart room lab’s firewall, that restricts users from gathering data remotely, outside of the Smart room lab and its network, it was very hard to gather real-time data from the lab as it was necessary to be present in the lab to do that. If this issue is solved, with continuous and consistent access to the lab, the work would be easier and the results would be improved. The type of sensors also had an impact and it can influence the results of this research, as mentioned, all of the sensors installed in the Smart room lab are part of the same collection that uses Z-Wave communication protocols and there are some IoT DMSs that do not support this type of communication. This issue can be solved by equipping Smart room lab with all the necessary sensors of certain type that is supported by each tested IoT DMS. Availability of IoT DMSs also created an issue, due to subscriptions that are necessary to use full capacity of most of the IoT DMSs. Working with the free version of the platforms made it harder to use the full capacities of the created middleware. By using full paid version of these IoT DMSs, there are many improvements that can be done. Regarding specific issues while developing this middleware, it is necessary to point out that API services from Yggio were unavailable for an entire month, this reduced productivity and made it harder to develop planned functionalities for this specific IoT DMS.
10. Conclusions and future work

In this thesis, different IoT DMSs were researched and compared with the main focus on their strengths and weaknesses from different aspects. Also, a middleware was created in the form of a web application that supports multiple IoT DMSs. Through this research, four IoT DMSs were reviewed, OpenHAB, Yggio, ThingSpeak and UbiDots. These four platforms were also used and integrated with the developed middleware. This research provides valuable insight for developers and researchers that are in the process of selecting an appropriate IoT platform for their specific use case in smart environments. This research also provides a guide to developing an application that enables users to integrate their smart environment with different IoT DMSs and use their features as necessary. The research presented the architecture of such middleware and suggests certain technologies that make it easier to develop a web application of this type. However, the limitations must be taken into consideration when interpreting the results.

Future research and work can address explained limitations and provide a higher level of interoperability in smart environments. With constant access to the Smart room lab and firewall issues fixed, as well with the full features of used platforms, there would be much more to work with. More specifically, it can be tested how other IoT DMSs work with the created middleware and to add more functionalities to improve their interoperability. The middleware itself can be improved to enable users and developers to control smart environments completely, and use complete power of migrated IoT DMSs.

Overall, this work provided a deeper understanding of IoT and smart environments in general and can help specific groups to use smart environments and create their own applications to ease the use of IoT platforms.
11. Acknowledgments

This work has been partially supported by the H2020 ECSEL EU project Distributed Artificial Intelligent System (DAIS)\textsuperscript{21} that has received funding from the ECSEL JU under grant agreement No 101007273. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands and Turkey. Special thanks to Sensative\textsuperscript{22}, for granting the full access to Yggio for the purposes of this thesis work. The work was more productive and every functionality of Yggio could be tested out and migrated with the created software.

I want to express gratitude to my supervisors Tijana Markovic and Miguel Leon Ortiz, for their guidance throughout this thesis work, support and patience. I am grateful to Mälardalens University and my home institution, University Mediterranean for the opportunity to gain amazing experience in this exchange period in Sweden and amazing conditions on both universities.

The most important, special thanks to my family and close friends for unconditional support throughout my whole study period, it had helped a lot and I dedicate this success to them.

\textsuperscript{21}https://dais-project.eu/
\textsuperscript{22}https://sensative.com/
References


