



JÖNKÖPING UNIVERSITY
School of Engineering

IoT-Based Digital Twin Framework for environmental monitoring in the Indoor Environment: Design and Implementation

Main Subject area: *Computer Engineering*

Author: *Ahmad Adnan Abdullah, Essa Alshehada.*

Supervisor: *Adyasha Swain*

JÖNKÖPING *2022 June.*

This final thesis has been carried out at the School of Engineering at Jönköping University within Computer Engineering. The authors are responsible for the presented opinions, conclusions, and results.

Examiner: Anders Adlemo
Supervisor: Adyasha Swain
Scope: 15 hp (first-cycle education)
Date: 2022-06-02

Abstract

Purpose: This thesis aims to describe how to design and implement an IoT-Based digital twin framework for environmental monitoring in the indoor environment.

To fulfill the purpose of the study, the following research question is answered. How to create a digital twin solution utilizing AWS to establish interaction and convergence between the physical environment in a classroom and the virtual environment?

Method: As a research method, the research has conducted design science research (DSR). DSR is a new method, and it is an effective tool for enhancing engineering education research methods.

Results: The study describes in detail the steps required to create the framework. The framework enabled interaction and convergence between the physical and virtual environments in a particular location.

Implications: The research contributes to broadening the knowledge on using the Internet of things (IoT), digital twin (DT), and Amazon web services (AWS). The study provides future research with reference data and a framework to build upon.

Research Limitation: Due to time constraints, the study's scope and limitations are limited to the technologies that the participating company, Knowit, provides. Knowit AB is a Swedish IT consulting company that supports companies and organizations with services in digital transformation and system development. The study aims to create an AWS-based IoT framework, not improve the digital twin concept. The framework was implemented at Jönköping University. This work is also limited to temperature and light intensity as environmental parameters.

Keywords: Amazon web services (AWS), Cloud Computing, Digital Twin solution (DT), Environmental Data, Environmental Monitoring Sensors, IoT (Internet of Things), Smart Building.

Abbreviations

AWS: Amazon Web Services.
DSR: Design Science Research.
DT: Digital Twin.
HDMI: High-Definition Multimedia Interface.
IoT: Internet of Things.
JTH: Jönköping School of Engineering.
MQTT: Message Queuing Telemetry Transport.
OS: Operating System.
RPi: Raspberry Pi.
UI: User Interface.

Acknowledgement

We, the authors of this thesis, would like to express our gratitude and appreciation to those who have helped us along the way; without them, this research would not have been possible. Tobias Wahlström from Knwoit provided us with all equipment and guidance. Adyasha Swain, our Jönköping University advisor, for her assistance and direction throughout our study.

Abstract.....	ii
Abbreviations	iii
Acknowledgement	iv
I Introduction.....	I
1.1 BACKGROUND.....	1
1.2 PROBLEM STATEMENT.....	1
1.3 PURPOSE AND RESEARCH QUESTION.....	3
1.4 SCOPE AND LIMITATIONS	4
1.5 DISPOSITION	4
2 Method and implementation.....	6
2.1 DESIGN OVERVIEW OF THE FRAMEWORK.....	7
2.2 IMPLEMENTATION	7
2.2.1 Configure the RPi	8
2.2.2 Configure environmental Sensor and Z-Wave stick.....	8
2.2.3 Configure AWS	8
2.3 DATA COLLECTION.....	10
2.3.1 Data collection during implementation.....	10
2.4 DATA ANALYSIS.....	11
2.5 VALIDITY AND RELIABILITY	11
3 Theoretical framework	13
3.1 PREVIOUS WORK	13
3.2 DIGITAL TWIN	14
3.2.1 The History of the DT.....	14
3.2.2 Types of DT	15
3.2.3 Applications of DT	15
3.3 ENVIRONMENTAL MONITORING SENSOR	16
3.4 RASPBERRY PI	16
3.5 Z-WAVE STICK.....	16
3.6 MQTT	16
3.7 AWS IoT CORE	17
3.8 AMAZON TIMESTREAM	17

3.9	GRAFANA.....	17
4	Results	18
4.1	THE RESULT OF THE DESIGN AND IMPLEMENTATION	18
4.1.1	Sensor and openHAB.....	18
4.1.2	OpenHAB and AWS IoT Core.	19
4.1.3	AWS IoT Core and the Timestream	20
4.1.4	Timestream and Grafana.....	22
4.2	RESULTS ANALYSIS.....	23
4.3	ANALYSIS RESEARCH QUESTION	24
5	Discussion	25
5.1	RESULT IN DISCUSSION	25
5.1.1	Research question	25
5.2	DISCUSSION OF METHOD AND IMPLEMENTATION	26
6	Conclusions and further research	27
6.1	CONCLUSIONS.....	27
6.1.1	Practical implications.....	27
6.1.2	Scientific implication	27
6.2	FURTHER RESEARCH	28
6.2.1	The environmental monitoring sensors' placement.....	28
6.2.2	Utilizing the using of the AWS.....	28
7	References	30
8	Appendices	33
	APPENDIX 1	33
	APPENDIX 2	34
	APPENDIX 3	37
	APPENDIX 4	37
	APPENDIX 5	41

1 Introduction

The chapter provides a background and a clear motivation for the study and the problem area the study addresses. Further, the purpose and the research question are presented. The scope and delimitations of the study are also described. Lastly, the disposition of the thesis is outlined.

1.1 Background

Today the world is getting increasingly digital and interconnected, where more smart cities have become common in society. These smart cities and buildings are based on unique technology. These technologies refer to software and hardware making buildings "smarter." The concept of smart cities and buildings is built on the integration of the real world and the digital world generated by the Internet of Things, cloud computing, and digital twin to achieve people and thing perception, control, and intelligent services (Marmolejo-Saucedo, 2020).

Kevin Ashton, the British technology pioneer, was the first to use the term "Internet of Things" (IoT) in 1999 to describe a system in which sensors connected physical objects to the Internet. Ashton defined the Internet of Things as an interconnected computational device that makes it possible to collect real-time data about a specific environment in a specific place (Radouan Ait Mouha, 2021).

IoT is a linked network of computing devices, mechanical and digital machines, and objects. It enables data transmission without human-to-human or human-to-computer interaction (McClelland, 2020).

Cloud computing is a revolutionary concept with multiple definitions for many corporations, governments, and consumers. Cloud computing has numerous definitions since it does not refer to a single technology but rather to a concept comprising a set of several different combined technologies. According to the IEEE Computer Society, the most comprehensive definition of cloud computing is "A paradigm in which information is constantly stored in servers on the Internet and cached temporarily on clients that include desktops, entertainment centers, computers, notebooks, handhelds, etc." (Almarabeh et al., 2016).

Dr. Michael Grieves of the University of Michigan first proposed the notion of the digital twin in 2002, naming it "Conceptual Ideal for Product Lifecycle Management (PLM)" to reflect the relationship between actual and virtual space (Lo et al., 2021).

It uses a digital format to replicate the physical model for remote monitoring, viewing, and control. It is a living model of a physical system that adjusts to operational changes in real-time using data from many different IoT sensors and devices (Bhatt & Shah, 2014).

1.2 Problem statement

In terms of the evolution of the Internet of Things, it is estimated that by 2022, 85 percent of companies' information technology platforms will host a digital twin. Comparable to remote access and control of smart home equipment using IoT and digital twins' technologies, the framework will be the system that enables remote

monitoring and control of the indoor environment in a particular space (Marmolejo-Saucedo, 2020).

Automating environmental monitoring and control in Smart Buildings, where data is collected, transmitted, and managed without human intervention, requires interoperability and communication between devices. It also involves understanding and handling a large amount of environmental data. These requirements can be fulfilled by using the digital twin solution.

The digital twin's definition varies depending on the field it is used in. Still, the standard definition of a Digital Twin is "a virtual dynamic representation of a physical system, which is connected to it over the entire lifecycle for bidirectional data exchange." (*Digital Twin Development and deployment*, 2021). Figure 1 below visualizes what a digital twin is.

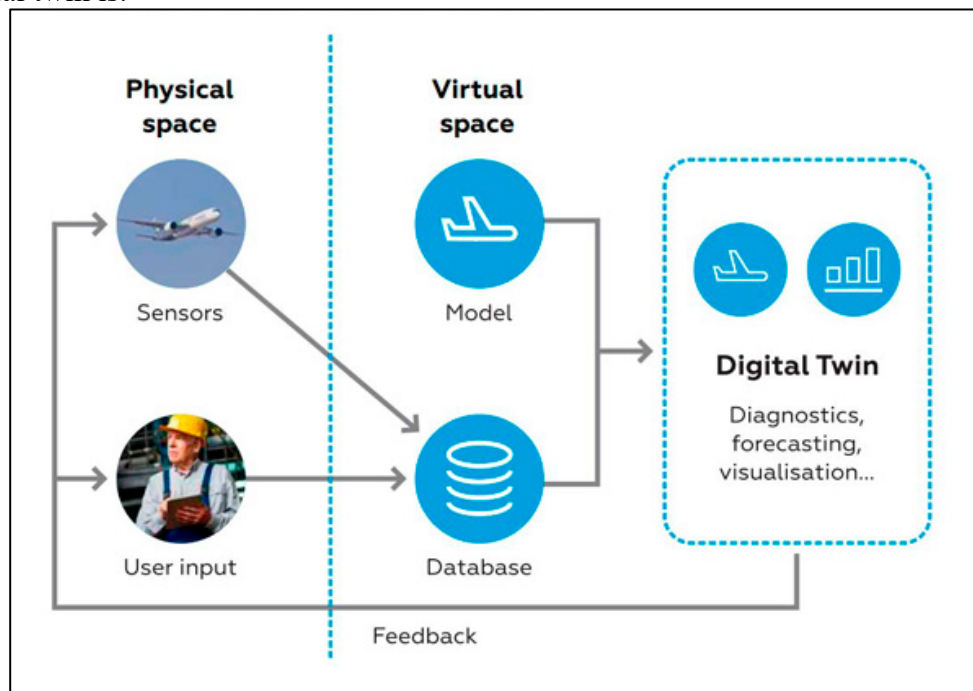


Figure 1. a visualization of Digital Twin's concept (*Digital Twin Development and deployment*, 2021).

Digital twins provide an intuitive way to store, organize, and access stockpiles of data generated by Internet of Things (IoT) devices in the Smart Building. The ability to comprehend how systems work and interact is improved by using a digital twin, as is the ability to examine parameters and interdependencies. A digital twin allows for low-cost and risk-free experimentation as a virtual environment. The key benefits of digital twins are system understanding, what-if analysis, and clarity. The goal of clarity in a digital twin is to help verify analyses, improve comprehension, and make it easier to discuss findings and concepts (*Digital Twin Development and deployment*, 2021).

It is required sensors, connectivity, data processing, and data visualization to count an IoT Solution as an IoT Solution. The reason to have a sensor is to capture environment monitory data, and connectivity is needed because the sensor must be able to send all the data to the cloud to process it in real-time. The benefit of having the data in the cloud is to have the ability to process and visualize data coming from different devices in real-time (McClelland, 2020).

Cloud computing services like Amazon web services would be a significant benefit. Amazon Web Services (AWS) is Amazon's extensive, ever-evolving cloud computing platform containing platforms like AWS IoT Core and Amazon Timestream. (*Amazon Timestream* 2022; *Amazon Web Services (AWS)* 2020; *AWS IoT Core features* 2022) AWS IoT Core makes it possible to connect IoT devices to AWS Services. For IoT and operational applications, Amazon Timestream provides a fast, scalable, and serverless time series database service. (*Amazon Timestream* 2022; *AWS IoT Core features* 2022)

The use of DT solutions in other fields than the industry has increased in the last few years (Marmolejo-Saucedo, 2020). The global COVID-19 pandemic in 2020 was one of the driving forces behind the increased demand for DT in the pharmaceutical and healthcare industries as well as the manufacturing industries. (MarketsandMarkets, 2020)

Realizing a massive technology like DT presents its own difficulties. The difficulties associated with developing a DT are proportional to its scope and implementation complexity. The most significant obstacles DT needs to overcome to reach its full potential are its high implementation cost and the lack of standards and regulations for the implementation. As there is a multitude of DT models and architectures in the literature, there is a need to define a consistent framework for DT throughout Smart Buildings that include a shared and mutual understanding of interfaces, standardization for uniformity, and an efficient design of data flow to facilitate data accessibility without compromising its security. (Fuller et al., 2020)

Digital Twin: Origin to Future is research about DT history and future where the authors discuss DT's concept and applications as well as the future and challenges of DT. The authors figured out the most considerable difficulties and obstacles DT needs to overcome to reach its potential. These difficulties and obstacles are due to the lack of a standard implementation of the DT framework, which leads to high costs and time when it comes to implementing the framework based on DT's concept. (Singh et al., 2021)

A Smart Campus' Digital Twin for Sustainable Comfort Monitoring (Zaballos et al., 2020) is research using DT solutions in the indoor environment for Sustainable Comfort Monitoring. The research does not show a standard DT solution design and implementation method in AWS. In A Smart Campus' Digital Twin for Sustainable Comfort Monitoring, the authors used only Microsoft Azure and AWS solutions for image recognition.

Industry 4.0 and the Digital Twin for Building Industry (Mateev, 2020) is another research that provides a framework based on DT and IoT utilizing Microsoft Azure (Azure Digital Twins Service and Azure IoT Stack). The study does not provide a technique for designing and implementing a framework using AWS.

1.3 Purpose and research question

As mentioned in chapter 1.2, there is enough information like architectures and models in the literature about IoT and DT's technologies. However, there is a lack of standard method DT's digital twins' solutions in AWS. The main goal of this framework is to overcome the challenges that arise with the use and development of DT by providing a standard method for design and implementation that saves time and cost.

This gap allows research on creating and implementing a generalized and extensible framework based on IoT and DT technologies in the indoor environment for environmental monitoring using AWS.

The framework will be used to achieve interaction and convergence between the physical and virtual environments in a classroom at the Jönköping School of Engineering (JTH) using a DT solution in AWS.

To achieve this objective, the following research question has been posed:

1. How to create a digital twin solution utilizing AWS to establish interaction and convergence between the physical environment in a classroom and the virtual environment?

1.4 Scope and limitations

Due to time constraints, the study's scope and limitations are constrained to be relevant to Knowit, with whom the study is being conducted in conjunction. The study is concentrating on using AWS. The decision to focus on AWS was made because Knowit is interested in AWS and because AWS provides a high level of protection for customers' virtual data. The market for AWS has a promising future due to its price model, scalability factor, and quick reaction time compared to other cloud computing services.

The study aims to create an IoT based framework using AWS, not to improve the DT solution. Implementing was done in a lecture room at Jönköping University to address the research question in this study. Furthermore, because this work uses a specific sensor (Fibaro sensor), it is constrained by the sensor's output, in other words, temperature, and light intensity.

1.5 Disposition

This section will explain how the report is structured and briefly explain what is covered in these chapters.

2. Method and Implementation.
This chapter will describe and motivate the work process of the study. It creates a clear connection between the methods used to describe the implementation process for the framework and the research question. It explains in depth how the implementation is done and how the data is managed.
 3. Theoretical framework.
The theoretical framework reviews the relevant research and the literature while providing a quick overview of the different technical components used throughout the study and their background.
 4. Results.
The chapter will present the design and implementation of the framework objectively and coherently.
 5. Discussion.
-

This chapter will discuss the results of the study concerning previous studies. The chapter is divided into two subsections. The first subsection will discuss the results concerning the purpose and research question. The second one is method discussion will discuss the choice of methods and the execution.

6. Conclusions and further research.

This chapter will present the conclusions from the study and suggestions for further research.

2 Method and implementation

This chapter describes the method used in the study to get enough knowledge to answer the research question. The research method is Design Science Research (DSR), and it is a relatively new method that aims to build a new reality rather than describe an existing reality (Zhan & Walker, 2019).

DSR is a problem-solving technique with varying objectives depending on the application domain. A DSR's primary objective is the creation of a novel technical artifact. This artifact will most likely be delivered in the field of information systems in the form of digital innovation (Dresch et al., 2015). Figure 2 below shows the six steps in DSR and how they are related.

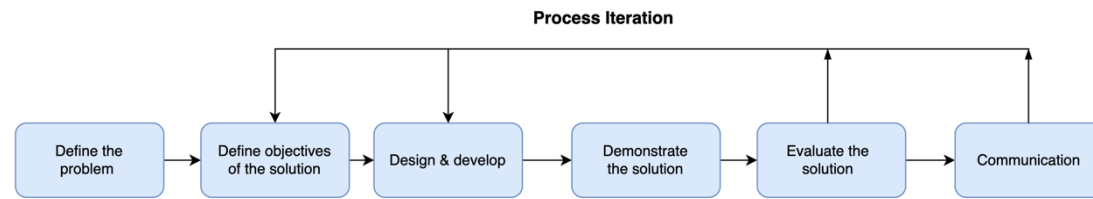


Figure 2 below shows the six steps in DSR.

The other objective of DSR is to produce not only a novel technical artifact, but also the necessary processes and procedures for its deployment and use in the problem context. It is a central research method in engineering, architecture, economics, and information technology domains because of its ability to solve problems and ways to improve or change existing solutions. DSR enables a thorough understanding of the problem, the development of a solution, and finally, the validation and refinement of the solution, and it consists of six steps (Grenha Teixeira et al., 2017; Peffers et al., 2007; Zhan & Walker, 2019).

Initial research for this thesis is the first thing to be done. During this phase, various data and information relevant to creating and implementing an IoT and DT solution are gathered. These findings are gleaned from peer reviewed research papers and articles in which existing data is extracted.

Following the collection and evaluation of various data is the next step. This step relates to creating a framework based on IoT and DT solutions using AWS in a given space. This requires knowledge of the theoretical and technical components of the framework mentioned in chapters 3.3 -3.9. In addition, the ability to create software that can be integrated with hardware to achieve a specific goal.

When the solution has been formulated and the framework has been established, it is time to demonstrate and evaluate the solution to determine whether it answers the research question the study seeks to answer and how it relates to other research. For this reason, this thesis employs DSR as its methodology. This is primarily since DSR is intended as a method for designing artifacts with the inherent purpose of solving problems via an iterative design process in which the artifact is evaluated. Thus, this methodology pairs well with software development, as it aims to reap the benefits of this iterative approach, which permits one to test, evaluate, and redesign their artifact in order to further its development. Moreover, the collected and generated data within the process is quantitative (Lapão et al., 2017).

2.1 Design Overview of the framework

The framework employs an overall system architecture shown in Figure 3. The framework consists of two main components, the hardware component and the software component. The hardware component includes the environmental monitoring sensor, Raspberry Pi (RPi), and a Z-Wave Stick wireless transceiver. These devices are all managed by RPi using an operating system called OpenHABian.

The software component includes the AWS, namely AWS IoT Core, Amazon Timestream, and Grafana.

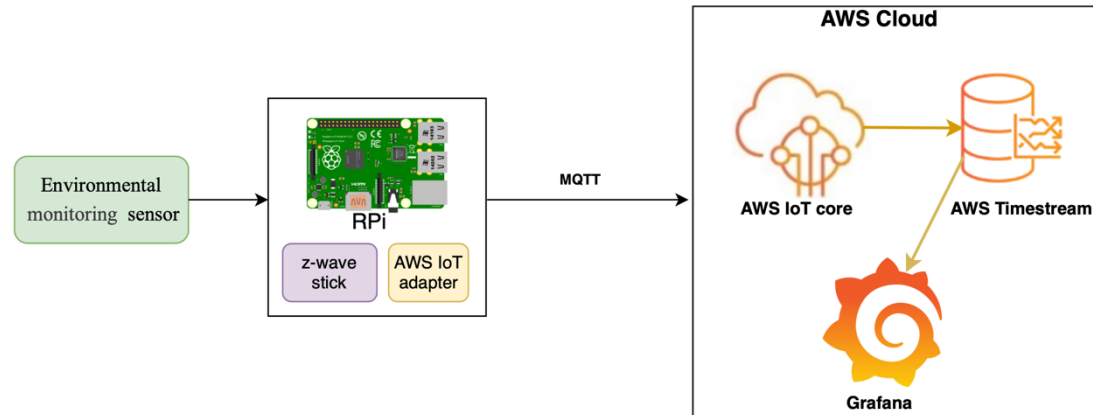


Figure 3 shows the flowchart of the framework.

The environmental recorded data is transmitted to the AWS IoT Core with the help of the RPi connected to the AWS using the AWS IoT adapter. The data transmission is done using the message queuing telemetry transport (MQTT) data protocol for receiving and sending data. Data storing is done in Amazon Timestream and Grafana for data visualization. A detailed description of the components is provided in chapters 3.3 – 3.9.

2.2 Implementation

Since the purpose of this study is to create and design a framework based on the DT concept using the AWS, the framework or the artifact was iterated and built around the RPi, using different hardware components and software to find the most reliable and accurate artifact. The iteration process consists of two different types of iterations. The first type of iteration is iteration for the choice of the hardware component, and the other type is about the configuration of these hardware components together to get the output data that answers the research question.

The first iteration of the artifact, shown in Figure 3, used the RPi model 2 and two environmental monitoring sensors of type LM-1ZW, (*Light Sensors*) one for temperature and the other for light intensity. This iteration provided readable output data, but RPi model 2 worked only with an ethernet connection, and it is not available in the marketplace widely, which is unsuitable for standardization's framework purpose. The other drawback of this iteration was that using two different sensors led to high costs and time for implementation. The second iteration of the framework used RPi model 3 B+ and Fibaro motion sensor. This iteration provided readable and good output data regarding the chosen hardware components for the framework. RPi Model 3 B+ has features like a Wi-Fi connection, a low price, and ready accessibility on the market. Fibaro sensor is a multi sensor that senses both light intensity and temperature,

has ready accessibility on the market, and a big community. These features are suitable for standardization's framework purpose.

The other type of iteration that occurs during framework implementation is configuration iteration. The framework implementation follows the flowchart shown in Figure 3. The implementation process consists of three stages.

2.2.1 Configure the RPi

The first stage in the implementation process is configuring the RPi, which serves as the processor and connector of the entire framework; it also employs the operating system to instruct and control the functions of the other hardware equipment. The operating system, OpenHABian, has many built-in functions, which can be extended to perform more advanced functions like showing the log data and events of the hardware components connected to the RPi. The primary purpose of this stage is to configure RPi by connecting other hardware components and installing the operating system, OpenHABian, as shown in Figure 4b. and Appendix 1. The first iteration was to configure the RPi to receive the environmental collected data from the sensor where the data rate was 400 times per minute and at the same time send data to the AWS 150 times per minute. This iteration produced undesirable results due to the RPi's inability to withstand the high pressure caused by the data transmission frequency. The data transmission frequency from the sensor to the RPi was reduced to 75 times per minute and 75 times per minute from RPi to the AWS in the second iteration, but the same issue arose again. Finally, the data transmission frequency has been reduced, so the RPi sends data to the AWS only if the collected data changes and a maximum of 60 times every minute per sensor. Conversely, RPi receives data from the sensor 75 times per minute. This iteration solves the problem of high pressure and provides the ability to add more sensors if needed. A detailed description of the configuration of the RPi is provided in Appendix 1.

2.2.2 Configure environmental Sensor and Z-Wave stick

The environmental monitoring sensor is the first component in the framework and the Fibaro motion sensor. It is connected to the RPi using a Z-Wave stick to establish the connection between the RPi and the Fibaro Sensor. The Z-Wave stick uses Z-wave technology that can connect many sensors simultaneously. This configuration's first iteration was that the sensor collected temperature and light intensity a hundred times per minute. This iteration provided unreadable and unexpected outcomes and odd behavior; after two weeks of continuous use, the battery ran out despite having a two year lifespan. The next iteration modified the configuration so that the sensor measures temperature and light intensity 75 times every minute instead of sending the data 400 times a minute. This iteration provided no unexpected output data or behavior. A detailed description of the configuration of the sensor and the Z-Wave is provided in Appendix 2.

2.2.3 Configure AWS

The software configuration can begin after the hardware components, such as the RPi, Z-Wave stick, and environmental sensor have been appropriately configured. The AWS configuration consists of three different steps. The first step is configuring the AWS IoT Core responsible for the registration of the RPi once it's running locally. The connection between the AWS IoT and the physical device (known as the thing in AWS)

must have a thing record, a virtual representation of the physical hardware. After registering the device, a Connection Kit, a software development kit with all the necessary libraries, and a sample project will be received. Once the Connection Kit has been configured and the code has been tested, running the code once will ensure a stable connection. Validating that the data has been received in the IoT Core is essential and is accomplished by displaying a list of the received data with the time it was received. The second step in the configuration process is to configure the Amazon Timestream. The data received in the IoT Core must be transmitted to Amazon Timestream to be processed. When processing the data, it is necessary to build an Amazon Timestream database using the standard AWS database settings, and it is important to encrypt the database with a key from the AWS Key Management Service (AWS KMS).

In the first iteration, the framework is connected to the Stockholm region since the study is conducted in Sweden, and it works with no problems when it comes to retrieving the data in the AWS IoT Core. Unanticipated complications arose when it came time to configure the Amazon Timestream service; specifically, it was discovered that the Stockholm region does not possess the Amazon Timestream. In the second iteration, a search had to be conducted to find the region closest to Sweden to finish configuring the Amazon Timestream. The search concluded that the Frankfurt region should be chosen because it is geographically closest to the Stockholm region and contains the Amazon Timestream service. It was necessary to delete all configurations made in the Stockholm region for the Frankfurt region because the regions in AWS are entirely separate from one another. The choice of the Frankfurt region caused latency (Saud Albazei, 2021) in the arrival of environmental data from openHAB to Amazon Timestream. However, the latency is considered ineffective because the environmental data is received in the Timestream at the same hour, minute, and second that is received in openHAB from the sensor. In other words, the latency is at 800 milliseconds, which does not affect the accuracy and validity of the environmental data because it is still considered real time (de Kleijn, 2020).

The Amazon Timestream was configured and served its intended purpose in its second iteration, but the measurement value always lost its decimal when inserted into the database. In the third iteration, this problem was solved by creating a new table in the database and sending the first value with decimals, and it was found that it is essential to send the first numbers with decimals if the sensor may send values with decimals. It was essential to check if the value contained the same between openHAB and Timestream. AWS Timestream requires a timestamp when data is completely inserted in the database because it is a time series database. The timestamp is generated when the data is sent from the IoT Core to the Amazon Timestream database; consequently, the time recorded in the database for the data is before its insertion. From openHAB data was sent 2022/5/27 at 10:53.11.908 and to Amazon Timestream 2022/5/27 at 10:53.12.95. This means that the data was sent to the database in 1050 milliseconds. Amazon Timestream keeps track of all insertion latencies; at 2022/5/27 8:53 UTC, the average ingestion latency was 60.8 milliseconds. This means that it totally took 1110 milliseconds to become completely inserted in the database.

The last step in the configuration process is configuring Grafana, which is a visualization tool. The configuration is accomplished by adding Timestream as a data source in Grafana. To establish a connection, it is critical to provide the correct

credentials and database. Sensor data can be visualized with the help of dashboards and panels in different diagrams and tables depending on the purpose of the visualization. The first iteration of Grafana's configuration had settings that made it take five minutes to update the charts. These settings produced undesirable results because five minutes is a long time where the collected data in Amazon Timestream has been changed several times before it is visualized in Grafana. During the second iteration, Grafana creates a line chart as soon as it receives the data. Each time Grafana requests new data in 100 millisecond intervals instead of five minutes as in the first iteration. The results in the second iteration were similar to the data shown locally in openHAB. A detailed description of the configuration of the AWS is provided in Appendix 3-4.

2.3 Data collection

As regards answering the research question in the study, data is collected through a mix of available information and enacted testing.

Firstly, gathering existing data is done by conducting a literature review, which is used to gain knowledge about which equipment is most effective and suitable for the framework. The equipment used in the framework fulfills the need to overcome the challenges for DT solutions when it comes to standardization, namely high cost and time. This review is based on data from peer reviewed articles and trustworthy websites about the equipment in these components.

Secondly, this data is utilized as a starting point and a point of comparison for any results from the testing. As a result of executing these tests in a controlled environment, data sets are generated, which play a crucial role in software development because they facilitate the implementation and design of the framework.

2.3.1 Data collection during implementation

Once all equipment has been chosen, implementation of the framework can be conducted as shown in Figure 3. This process requires each hardware shown in Figure 4a to be appropriately configured as shown in Figure 4b to the flowchart in Figure 3. The names of the components shown in Figure 4.a. are mentioned in table 1.

Table 1 shows the names of components used in the framework

Number	Equipment	Number	Equipment
1	An HDMI cable.	6	SD card
2	Fibaro sensor.	7	A keyboard
3	Ethernet cable	8	Z-Wave stick
4	RPi 3B +.	9	A mouse
5	AC adapter		

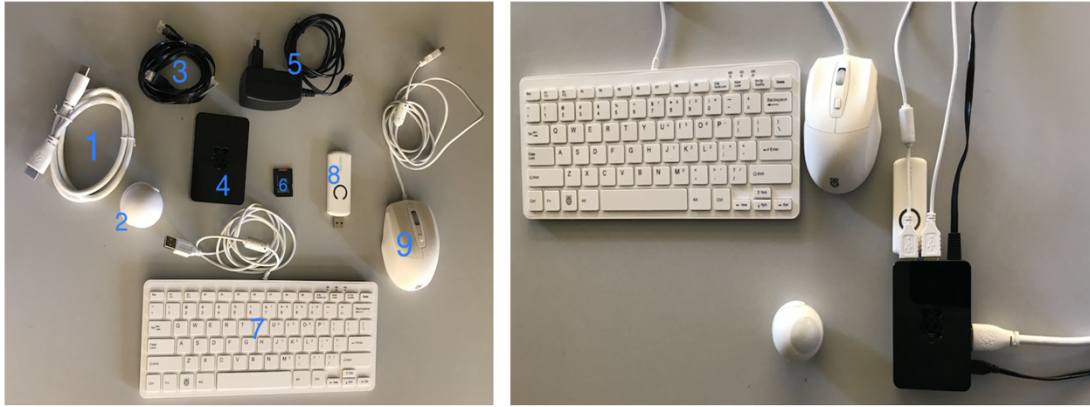


Fig. 4. (a) Pre-configuration components; 4(b) post-configuration system

Alongside the configuration to implement the framework, an array of verification tests to ensure that components have appropriately been configured and connected and the configured system behaved as intended. These verification tests are conducted to ensure that the data collected in real time and the values are relevant. One of these verification tests compares the collected data shown locally in the OpenHABian and the same collected data in the Amazon Timestream. Another test uses two sensors to collect data and compare the collected data. These tests are done in order to create the framework that achieves the interaction and convergence between the physical and virtual environment.

The data collected during the observation of the implementation process is the result of configuring the components, and with the help of the verification tests, it will be compared to see if any data is irrelevant to the implementation process. If any irrelevant data has been detected, the verification tests will be repeated multiple times to ensure no irrelevant data has been detected.

2.4 Data analysis

The collected and generated data for this thesis will be analyzed utilizing the method of dynamic analysis and quantitative research method. The dynamic analysis method is ideal for investigating the dynamic qualities of an artifact, such as those observed in dynamic systems (Dwivedy & Eberhard, 2006). Using this method, the software is tested and assessed by executing it in real time. Instead of attempting to identify every error in the component by evaluating it on its own, the objective is to identify and address errors while the entire framework is running in its intended environment.

2.5 Validity and reliability

The study's credibility can be divided into validity and reliability. To ensure the reliability of the data, the authors documented every step they took. Every step and configuration taken while implementing and testing to answer the research question is recorded. This will ensure that the person who tries to create and implement the framework will have the same results as the authors.

Selecting approaches was not random; instead, authors adapted the selection depending on the literature review they did and the existing knowledge in the field.

Since validity means measuring what is relevant in the context,(Gunnarsson, 2020) to ensure the study's validity, the environmental data is collected in real time and then

compared in Grafana to ensure there is no delay between sensing and sending data. The hardware and software components are selected to fulfill the objectives of designing the framework.

3 Theoretical framework

The chapter starts by presenting previous work from which the current study is inspired. Furthermore, a theoretical basis is given to provide a deeper understanding of the technologies being used in the research.

3.1 Previous work

From researching previous papers mentioned in chapter 1.2, (Fuller et al., 2020; Mateev, 2020; Singh et al., 2021; Zaballos et al., 2020) is known the definitions of DT, the applications of DT, and the challenges to implement DT. One of the challenges identified by prior research is the lack of a standardized method for creating and designing a framework that provides solutions for saving time and cost to make the use of digital twins effective.

Digital Twin: Origin to Future provides an overview of the current state of DT as well as its future applications and challenges. The purpose of this research is to examine the role and application of DT and how it can add value to existing systems in Smart City and other fields. DT can add value to any industry by decreasing time-to-market, optimizing operations, reducing maintenance costs, boosting user engagement, and fusing information technologies. The overview of DT discussed the various definitions of DT in accordance with their respective fields of study, in which the origins of DT and its early applications are explained and compared throughout the literature. The most interesting part of this work, which inspired the current study (IoT-Based Digital Twin Framework for Environmental Monitoring in the Indoor Environment: Design and Implementation), is the implementation challenges of a framework based on DT and IoT concepts. The key obstacle is the absence of regulations and standards, as there is no consistent, standardized, and generalized framework for DT throughout the Smart Building concept using AWS. This framework's practical design and straightforward implementation facilitate data accessibility without compromising its real-time nature.(Singh et al., 2021)

Digital Transformation Revolution with Digital Twin Technology is an additional study that discusses the role of DT technology in the digital transformation revolution in various fields. As a result of the industry 4.0 industrial revolution, the DT concept, which is anticipated to affect numerous fields in the near future, has entered these fields. DTs have begun to be utilized in civil fields as well as industrial and engineering fields due to the time and cost savings they provide. If there is a standard methodology that makes the application and implementation of DT possible and practical, DT as a promising technology will have a direct impact on daily life. According to the study's findings, there is a deficiency of general guidelines that make designing and implementing a standardized framework that can be used in various fields feasible and practicable (Erol et al., 2020).

Digital Twin: Enabling Technologies, Challenges, and Open Research is additional research addressing issues related to “What are the applications, challenges, and enabling technologies associated with IoT, data analytics, and Digital Twins?”. The study is answering the question by providing an overview of DT where its definitions, types, applications, and challenges are highlighted. As highlighted in this paper and observed in numerous new and emerging technologies, the lack of design and

implementation standardization is a significant challenge for Digital Twin. This causes disparities between Digital Twin projects in the Smart Building domain. Diverse definitions are a contributing factor; this, coupled with the lack of standardization, is a challenge that hinders the development of the Digital Twin technology.(Fuller et al., 2020)

Papers A Smart Campus' Digital Twin for Sustainable Comfort Monitoring (Zaballos et al., 2020) and Industry 4.0 and Digital Twin for the Building Industry, (Mateev, 2020) provide case studies for DT implementation. A Smart Campus' Digital Twin for Sustainable Comfort Monitoring proposes a DT modeling procedure that combines a set of advanced intelligent features, such as an IoT network and cloud computing, to transform university spaces into information sources for intelligent decision-making processes, thereby reducing operational costs and improving the quality of life. Microsoft Azure is the cloud computing platform used for DT modeling (Azure Digital Twins Service and Azure IoT Stack.). Industry 4.0 and the Digital Twin for Building Industry is a study that describes business cases and best practices in the design of Internet of Things (IoT) solutions for the construction industry that are powered by DT. This study produces a reference architecture and prototypes that demonstrate the application of DT in Smart Buildings. This research is technology agnostic; however, prototypes of sample solutions are considered within the context of Microsoft Azure (Azure solutions, covering Azure Digital Twins Service and Azure IoT Stack.)(Mateev, 2020; Zaballos et al., 2020)

These studies do not provide any implementation or modeling for DT using AWS, which arouses interest in conducting research about design and implementing framework based on DT and IoT concepts using AWS.

3.2 Digital twin

Digital twin's (DT) definition varies depending on the context it is used, but the general definition of DT is a virtual vision of a product, service, or process that data scientists and IT professionals can use to run simulations before the machines are created and deployed. They are also changing how optimized technologies like IoT, AI, and data analytics are. The technology that underpins these DTs has a long way to go, and it is more likely to do so with people and procedures.(*What is a digital twin?*, 2019)

DT technology entails the creation of virtual replicas of physical objects or processes that simulate their behavior. The goal is to analyze its effectiveness or behavior in specific situations in order to enhance its effectiveness (Marmolejo-Saucedo, 2020).

3.2.1 The History of the DT

Despite DT technology gaining enormous popularity over the past few years, the concept is not entirely novel. Michael Grieves developed the concept of Product Lifecycle Management (PLM) at the University of Michigan in 2002.(Grieves, 2016) The proposed model consists of three parts: real space, virtual space, and a linking mechanism for the flow of data/information between the two; the model was subsequently dubbed "Mirrored Spaces Mode." DT first appeared in 2010 in a draft version of NASA's technological roadmap. DT was also referred to as "Virtual Digital Fleet Leader" in NASA road maps. NASA was the first organization to define DT as "an integrated multi-scale, multi-physics, probabilistic simulation of a system or vehicle

that uses the best available physical models, fleet history, sensor updates, etc., to replicate the life of its flying twin." The plan was to use DT to simulate the aircraft's physical and mechanical properties in order to predict any fatigue or cracks in the structure, thereby extending the aircraft's remaining useful life.(Singh et al., 2021)

3.2.2 Types of DT

Different types of DT can be distinguished based on criteria such as the DT's creation time, level of integration, and applications. Based on these criteria, various authors have devised their own classifications of DT types. Grieves and Vickers identify two types of DT based on when DT is created during the product's life cycle. The first type, Digital Twin Prototype, occurs prior to the creation of the prototype, i.e., during the designing phase. The second type is Digital Twin Instance, created once the product is complete, i.e., during the production phase. (Singh et al., 2021)

Kritzinger et al. (Kritzinger et al., 2018) classified DTs into three subcategories according to their integration level. The first type is the Digital Model, where in this type of DT, the data between the physical and digital objects are exchanged manually, so any changes in the state of the physical object are not immediately reflected in the digital object and vice versa. The second type is Digital Shadow, where the data from the physical object flows automatically into the digital, but the reverse is still done manually. Consequently, any changes to the physical object are reflected in the digital copy, not vice versa. The third type is Digital Twin, and in this form of DT, data flows automatically in both directions between the physical and digital objects. Figure 5 below shows the different types of DT based on the level of integration (1) Digital Model; (2) Digital shadow model; (3) Digital Twin.(Singh et al., 2021)

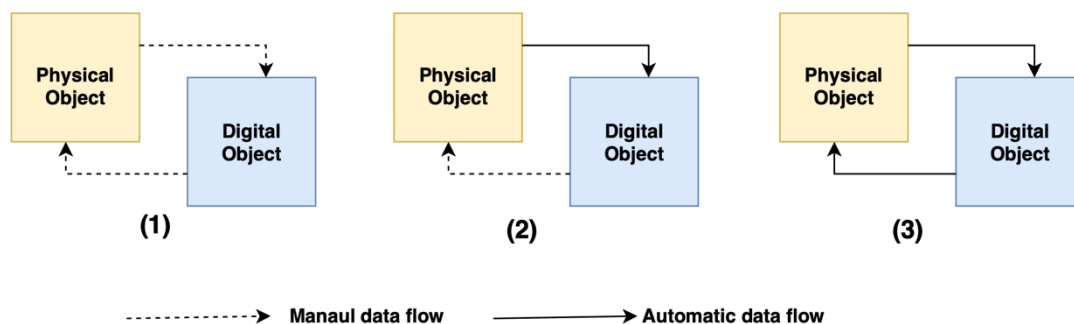


Figure 5 shows the different types of DT based on the level of integration (Singh et al., 2021)

According to Grieves, M. and Vickers, J.(Grieves & Vickers, 2017) DT can also be classified based on its applications. Prediction and interrogation are the two broad applications of a DT. As its name implies, a Predictive DT predicts the future behavior and performance of its physical counterpart, whereas an Interrogative DT is used to interrogate its physical counterpart's current or past state, regardless of its location.(Singh et al., 2021)

3.2.3 Applications of DT

In recent years, DT technology has gained immense popularity, and it has a bright future in many fields of study where numerous studies and research activities continue to be conducted. This resulted in the widespread application of DT in numerous fields,

including healthcare applications, industrial applications, and Smart City Management Systems (Erol et al., 2020).

3.3 Environmental monitoring sensor

The environmental monitoring sensor is the Fibaro Motion sensor. It is a multi-sensor that senses light intensity and temperature. It is compatible with any Z-Wave controller, and it supports protected mode (Z-Wave network security mode) with AES-128 encryption, which makes sending data securely. The Fibaro sensor is a battery-powered device with a battery life of two years designed to be mounted quickly and easily on any surface. The last feature is that Fibaro is made in the EU, which means that social and environmental issues are respected, and it is readily accessible on the market. (*Motion Sensor*, 2021).

3.4 Raspberry Pi

RPi 3B+ is the final revision of the RPi 3, and it has the features that are suitable and needed in the framework, namely the low power consumption, the small size, accessibility on the market, and the low price. Raspberry Pi 3 B+ has enough technical features for the study, namely the good networking support and, at the same time, a fast process (*Raspberry Pi beginner's guide 4th edition*, 2022). Raspberry Pi 3 B+ is the connector between the sensor and the AWS. It uses an operating system called openHABian.

3.5 Z-Wave stick

Z-Wave stick is a USB stick based that works with any platform, and it uses Z-Wave technology which is the most suitable technology for smart home automation. Z-Wave is secure because it uses the same encryption as online banking, and it is experienced where there are 100 million products worldwide using it (*Learn - Z-Wave*, 2022).

Z-Wave utilizes a mesh network architecture. Every (non-battery) device installed in the network acts as a signal repeater. As a result, the network becomes stronger as more devices are added to the home. While Z-Wave signals can easily pass through most walls, floors, and ceilings, devices can intelligently route themselves around obstructions to achieve seamless, robust, whole-home coverage. It has a range of 100 meters or 328 feet in the open air; building materials reduce that range. It is recommended for maximum efficiency to have a Z-Wave device approximately every 30 feet or closer (*Learn - Z-Wave*, 2022).

A detailed description of the Z-Wave stick and configuration process is provided in Appendix 2.

3.6 MQTT

MQTT is a publish/subscribe message transport protocol for clients and servers. It is light, open, and essential, and it is supposed to be simple to use. These qualities make it excellent for application in various circumstances, including confined environments where a minimal code footprint is required or limited network bandwidth. This communication is in the context of Machine to Machine and the Internet of Things (Andrew Banks, 2014).

3.7 AWS IoT Core

The AWS IoT Core is the AWS window for the outside world where the initial connection with IoT devices is configured. IoT devices can be connected securely to AWS IoT Core and it can be used even if these devices are offline. The AWS IoT Core provides the possibility to use a MQTT test client and a rule engine. The MQTT test client aids in verifying that data is received by AWS. Data can automatically be sent to Amazon Timestream by the rule engine as soon as it is received by the IoT core(*AWS IoT Core features* 2022). Each client connection is limited to 100 inbound and outbound publish requests per second by AWS IoT Core. Requests to publish that exceed the limit are discarded (*AWS IoT Core endpoints and quotas - AWS General Reference*, 2022).

3.8 Amazon Timestream

Amazon Timestream is fast, scalable, and serverless. Amazon Timestream is used for IoT and operational applications to store and analyze events faster than relational databases. Amazon Timestream saves time and money by keeping recent data in memory and moving older data to a cost optimized storage tier depending on user defined criteria. Amazon Timestream's purpose built query engine offers access to recent and historical data without specifying whether it's in memory or the cost-optimized tier. Amazon Timestream's built-in time series analytics tools can detect trends and patterns in data in near real time because it's serverless and scales dynamically to adapt capacity and performance. The underlying infrastructure is managed. .(*Amazon Timestream* 2022)

3.9 Grafana

Grafana is an all inclusive observability stack that makes it possible to monitor and analyze metrics, logs, and traces. This is made possible with the assistance of the software. Regardless of the location at which the data is stored, it is possible to query the data, visualize the data, set up alerts based on the data, and understand the data. It is helpful in the process of creating visually appealing dashboards, exploring those dashboards, and sharing those dashboards with others. Grafana provides the possibility to obtain data from the data source in intervals that are measured in milliseconds. (*Amazon Managed Grafana*, 2022)

4 Results

This chapter addresses the research question by analyzing the gathered data from the selection components, implementation process, and verification tests.

4.1 The result of the design and implementation

The choice of the hardware components used in the framework shown in figure 3 was not random but was made based on specific features in order to overcome the challenges facing the use of DT, which standardization regarding design and implementation. These features are low price, accessibility on the market, and popularity where the components have an extensive community.

The configuration process of the framework, shown in Figure 3, done in chapter 2.2, lead to dividing the implementation process into four stages. These stages, in turn, are divided into the following subtitles "sensor and the openHAB," "openHAB and AWS IoT Core," " AWS IoT Core and the Amazon Timestream," and " Amazon Timestream and Grafana." These stages are arranged according to the framework in Figure 3.

Figure 6 shows the classroom, where the sensors are placed at different locations when the iterations and verification tests have been conducted. The sensors are marked with a green circle.



Figure 6 shows the placement of the sensors in the classroom in JTH.

4.1.1 Sensor and openHAB

Figure 7 demonstrates the result of integrating the sensor with openHAB, where environmental data is collected by the sensor and transmitted to openHAB using Z-Wave technology. The sensor is a multi sensor that can sense both light intensity and temperature. The primary purpose of the current stage is to collect the environmental data using the sensor and visualize it locally in openHAB.

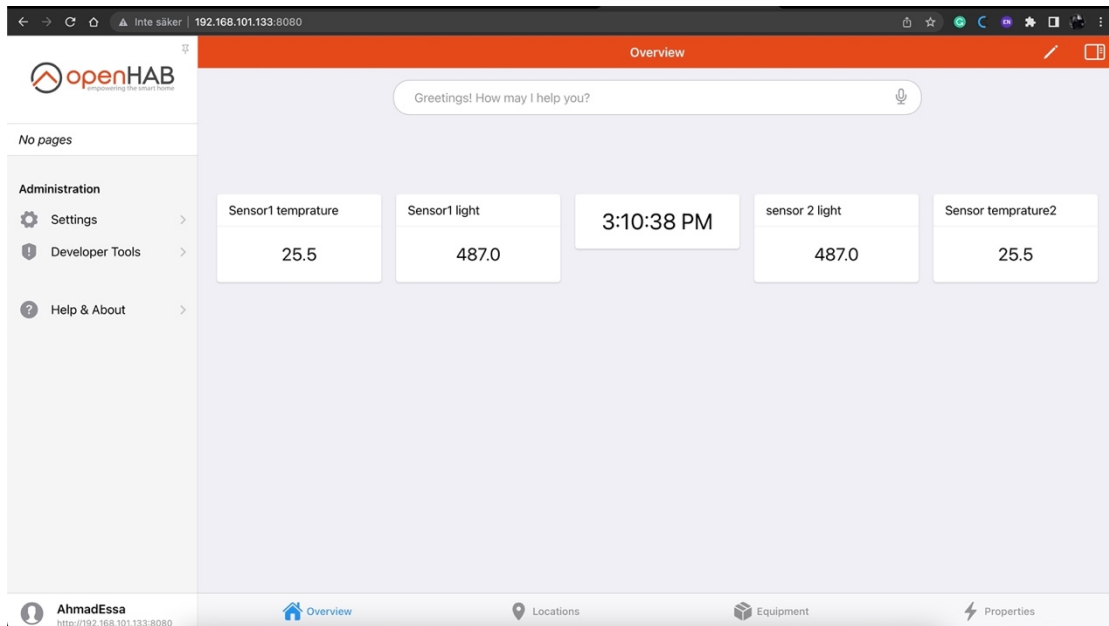


Figure 7 shows the collected data in openHAB dashboard.

4.1.2 OpenHAB and AWS IoT Core.

In this stage configured AWS IoT Core where it is connected with openHAB to ensure the environmental collected data is received in the AWS IoT Core with an ineffective latency. Figure 8 and 9 below shows the data collected from the sensors in openHAB and the AWS IoT core. Figure 8 below depicts sensor data displayed in the openHAB log view and when it was received in openHAB. Light intensity value was received at the time of 2022/5/27 11:57:33.605 and with the value of 151 Lux where Lux is a measurement unit of light intensity. It is crucial that the measured value retains its value and datatype when delivered to AWS IoT Core.

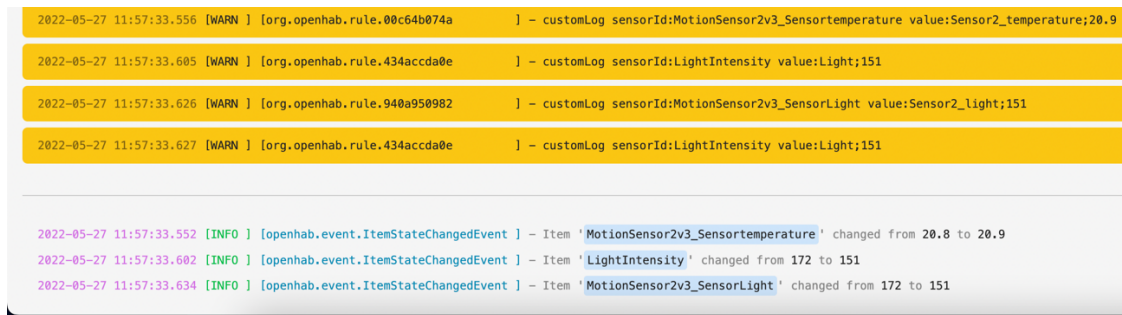


Figure 8 shows collected data in the openHAB logger.

The MQTT test client is used to keep track of the MQTT messages passing through AWS in the Frankfurt region. Light intensity data was received on 2022/5/27 at 11:57:33, and with a value of 151 Lux, we can see it at the bottom of the AWS IoT Core, as shown in figure 9. At this point, it is essential to ensure that the data received by the AWS IoT Core is identical to the data initially transmitted by the sensors.

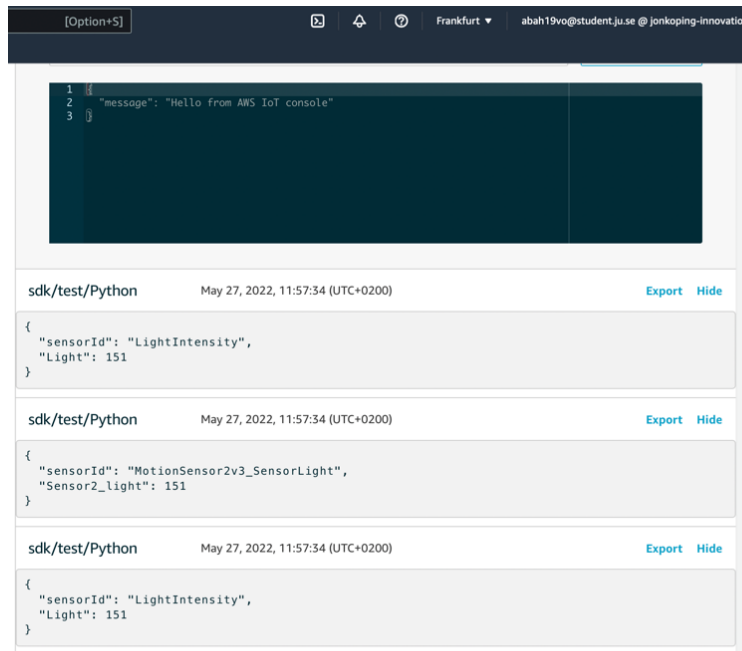


Figure 9 shows light intensity in the MQTT test client in the AWS IoT Core.

The MQTT test client was utilized to monitor MQTT messages traversing AWS in the Stockholm region. Figure 10 depicts temperature data received on 2022/04/12 at 13:06:30 at the bottom of the AWS IoT Core.

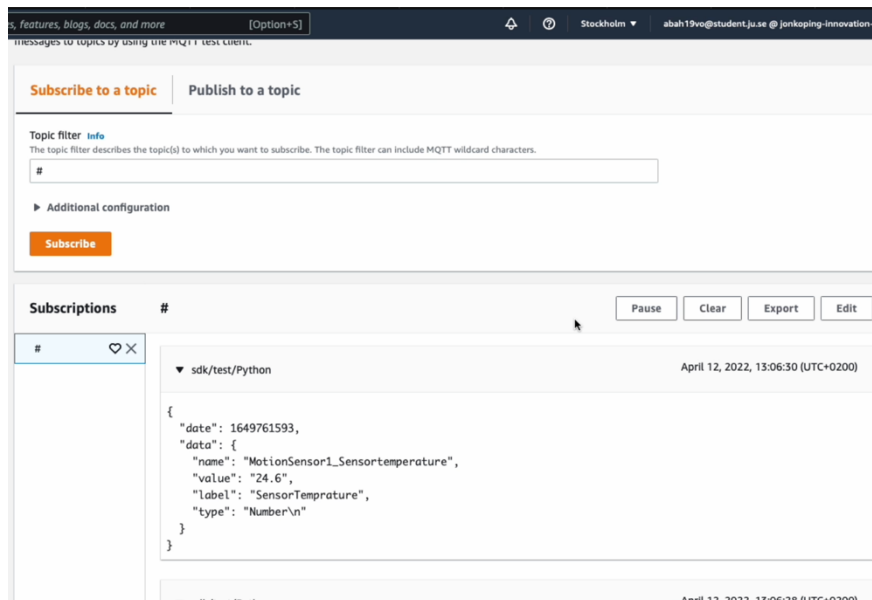


Figure 10 shows the temperature in the MQTT test client in the AWS IoT Core Stockholm region.

4.1.3 AWS IoT Core and the Timestream

In this phase, the connection between AWS IoT Core and Amazon Timestream is configured, and the collected data is then sent to Amazon Timestream for storage. Figures 11, 12, and 13 depict the outcome of integrating AWS IoT Core with Amazon

Timestream, in which environmental data was transmitted to Amazon Timestream throughout AWS IoT Core. Figure 11 depicts sensor data displayed in the openHAB log view when it was received. Light intensity from sensor 1 and sensor 2 was received on 2022/5/27 at 10:53.11.908 and 2022/5/27 at 10:53.11.896 in the Swedish time zone. The value is 96 Lux, which is based on a sensor's reading of light intensity. It is essential that the measured value maintains its value and datatype when sent to Amazon Timestream.

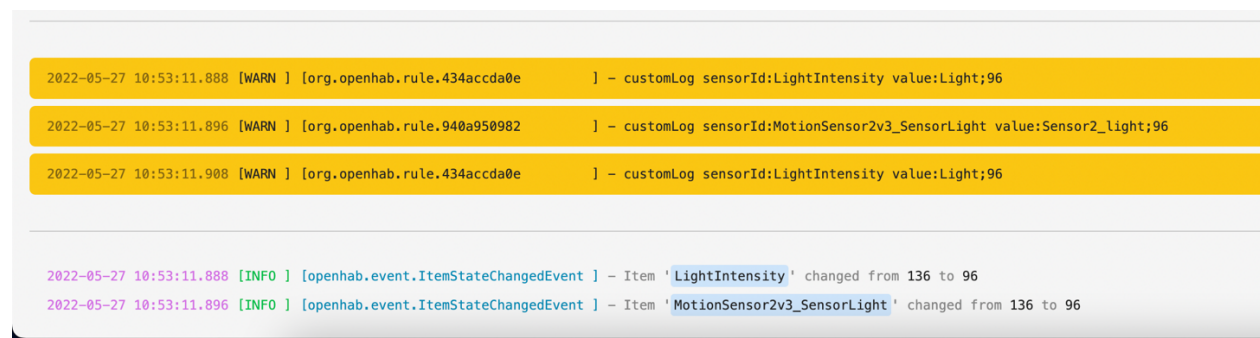


Figure 11 shows light intensity in the openHAB logger

Figure 12 below shows the data queried from Amazon Timestream. Where in the first row, the data matches the same time and same value, but the Timestream displays the UTC time zone. The data in the Amazon Timestream database corresponds to the data logged by the openHAB data logger in figure 11.

sensorId	measure_name	time	measure_value::double	measure_value::bigint
LightIntensity	Light	2022-05-27 08:53:12.950000000	-	96
MotionSensor2v3_SensorLight	Sensor2_light	2022-05-27 08:53:12.771000000	-	96
LightIntensity	Light	2022-05-27 08:53:12.660000000	-	96

Figure 12 AWS Timestream queried data.

The Ingestion latency p95 value represents the longest time it takes for 95 percent of the data to be inserted into the database in a time span. Figure 13 depicts a chart of ingestion latency p95 with one minute per value period. At 2022/5/27 8:53 UTC, the ingestion latency p95 value is 60 milliseconds, as depicted in Figure 13 below.



Figure 13 shows the Ingestion latency p95 in the Timestream.

4.1.4 Timestream and Grafana

The configuration process concludes with the configuration of the connection between Timestream and Grafana. At this stage, it is essential that Grafana visualize the collected data in real time, which means that Grafana will display a chart that is as identical to the openHAB chart as possible. Figure 14 demonstrates a User Interface (UI) visualization of the collected data locally in the openHAB. This figure shows how the light intensity value changes over time. The horizontal axis depicts the time in hours and minutes, while the vertical axis depicts light intensity. Collectively, they resemble a chart of light intensity over time in hours and minutes. The data in the chart below is updated as soon as the openHAB receives the data from the sensors.



Figure 14 demonstrates the change of light intensity in openHAB.

Figure 15 is a UI visualization in Grafana of the same data shown in Figure 14. The horizontal axis depicts the time in hours and minutes, while the vertical axis depicts light intensity. Collectively, they resemble a chart of light intensity over time in hours and minutes. Comparing Figure 14 and Figure 15 shows how they are comparable and similar, which means the collected data has been transmitted from the sensor via RPi, AWS IoT Core, and Timestream to Grafana without modification or delay.

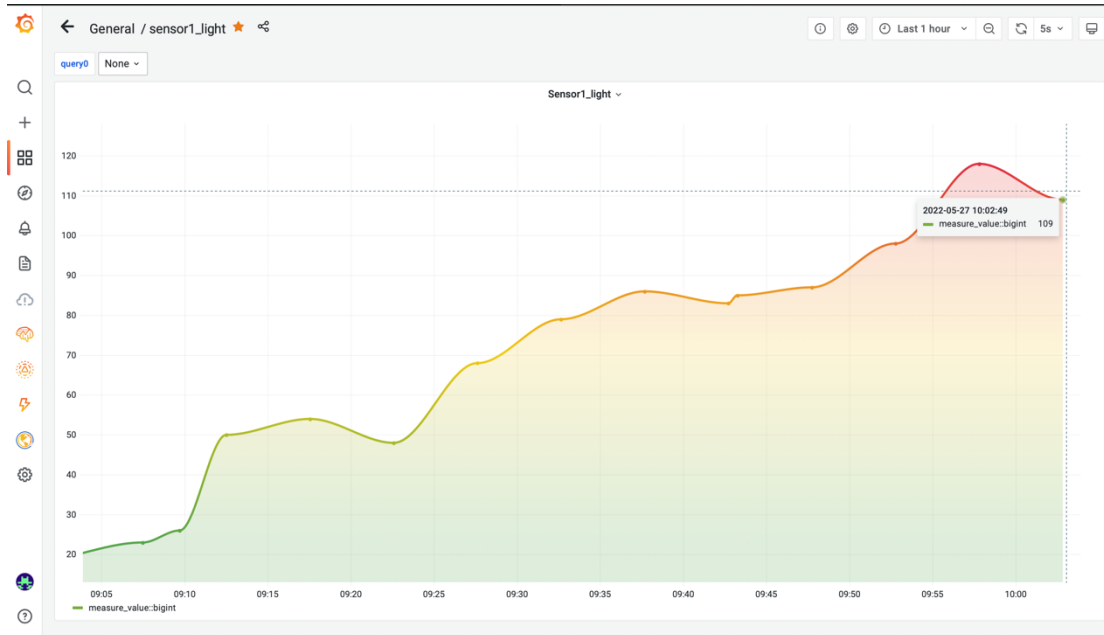


Fig 15 Grafana line chart.

4.2 Results analysis

With the use of dynamic analysis, the produced and developed artifact was exposed to iteration and testing. This is accomplished by iterating the choice of components and simulating the intended scenarios by conducting multiple tests, as stated in section 4.1, at various stages within a suitable environment. The hardware components were subjected to iteration to select the most suitable component for the framework, thereby achieving the framework's purpose. Each component's configuration was subjected to the same testing in order to identify any potential bugs and crashes that may occur during the implementation process.

The components' configuration was one of the outcomes of the implementation process, and these configurations were divided into four configurations. The first component's configuration is the sensor's configuration. While the first iteration produced undesirable and unexpected outcomes, the second iteration provided a readable and accurate output date. Since the new settings where the sensor measures temperature and light intensity 75 times, every minute does not affect the precision of each measurement but rather the rate at which data is transmitted to the RPi, the validity of the data measurements is unaffected.

The second outcome of the process is the RPi configuration, which is the entire framework's connector and processor. Due to the inability of the RPi to withstand the high pressure caused by the high data transmission frequency, the first and second iterations produced undesirable outcomes. The third iteration of the RPi configuration provided readable and accurate behavior and outcomes. The data transmission frequency in the third iteration does not affect the accuracy of the collected data because the data is sent to the AWS as soon as the data collected is changed.

The third result of the process is the configuration of the AWS IoT Core, which allows users to be connected to various regions where data is managed, and operations are running depending on the user's geographical position. The Stockholm region of AWS did not support Amazon Times during the first iteration of the framework, which posed

an unexpected challenge. To solve the problem, a second iteration was conducted in which a Frankfurt region was chosen, but all configurations were remade because the Stockholm and Frankfurt regions are entirely distinct.

In the second iteration, another unpredicted difficulty arose when the collected data value lost its decimal when the value contained decimals. The problem was solved in the third iteration by creating a new table in the database and sending the first value with non-changing decimals regardless of whether the value contains decimals.

The Grafana visualization was set up and used for its intended purpose as the fourth outcome of this procedure. In the first iteration of Grafana's configuration, the results were not updated as soon as data was updated in the Amazon Timestream, which led to inaccurate visualization. During the second iteration, Grafana creates a line chart as soon as it receives the data. Each time Grafana requests new data in 100 millisecond intervals. Grafana appears to be a good match for the openHAB visualization tool where the data is locally visualized. Figures 14 and 15 show the symmetry between the visualization of the same data in openHAB and Grafana.

4.3 Analysis research question

The purpose of analyzing the collected data is to answer the study's research question: How to create a DT solution utilizing AWS to establish interaction and convergence between the physical environment in a classroom and the virtual environment? The framework's design and implementation become more accurate by employing the DSR methodology and iteratively developing and improving an artifact throughout multiple phases. The iterations of the artifact mentioned in chapter 2.2 are divided into different iterations depending on the component's selection and configuration. The iterations of selecting the framework's components led to the RPi 3 model B+ and Fibaro sensor because they have the features needed to create and design the framework. RPi's configuration iterations concluded that the suitable data transmission frequency between the environmental monitoring sensor and the RPi is 75 times per minute and between the RPi and the AWS as soon as the collected data changes, up to a maximum of 60 times per minute. These iterations led to a connection between the RPi and AWS, where the data is received in AWS IoT Core with no data loss and with latency 800 milliseconds from the time it was received in openHAB. The connection between the RPi and AWS IoT Core enabled the connection between IoT Core and Timestream database where the data is saved with no data loss and the latency of 1110 milliseconds from the time received in openHAB. Timestream enables the connection to Grafana where all the data in collected by the sensors and stored in Timestream can be visualized.

5 Discussion

This chapter discusses the results of the research and further discusses the method used in the study.

5.1 Result in discussion

The purpose of the thesis is to conduct research in creating a framework for DT solutions utilizing AWS and IoT technology, and the goal of this research is to be presented in the form of the thesis.

In conjunction with the production of the framework, the configuration of the framework initially enables the collection of environmental data, which is followed by the transmission of the collected data to AWS, where it can be saved and managed in real time.

Configuring the framework enables the collection of environmental data and its transmission to AWS for real time storage and management. This process occurs simultaneously with the creation of the framework.

As a result, having the ability to measure and manage the environmental data required to accomplish the interaction and convergence of the physical and virtual environments is essential.

5.1.1 Research question

A research question has been posed to guide the implementation process toward the study's end goal in order to fulfill the thesis's purpose. The steps of creation and implementation, which were designed to work together, were intended to make it easier to find answers to the research question.

According to the research question, the answer to the research question can be divided into two parts, the first about the framework's design and the second about its implementation. Various factors must be considered when determining the optimal framework's components to achieve the framework's purpose. These shared factors include low cost, broad accessibility, and a large community. As regards determining the optimal components' configuration of the framework, there are a different few factor that need to be taken into consideration. The first factor is the time of monitoring of the data to determine if the data has been managed in real time. There are two separate variations for this process, these ones being if the data has been collected by the sensor in the real time or if it is transmitted from the RPi to the AWS IoT Core in the real time. The optimal configurations were determined to be a collection data rate is 75 times per minute and a transmission data rate of once the collected data changes and up to a maximum of 60 times per minute. A connection between the RPi and AWS has been established, with no data loss and a latency of 800 milliseconds from when openHAB receives the data. Since the RPi and AWS IoT Core were connected, data from the IoT Core and Timestream database could be saved without any data loss and with a latency of 1110 milliseconds from the time it was received in openHAB. Data collected by sensors and stored in Timestream can be viewed in Grafana thanks to the Timestream to Grafana connection.

5.2 Discussion of method and implementation

Regarding the method, its strength was that it provided a course of action that suited and accommodated the thesis purpose by not only allowing for the development of the artifact or configurations in this case, but also for employing an iterative approach that was conducive to answering the research question.

In addition, this allowed for flexibility throughout the process, which made it possible to deviate from the initial plan when problems arose. Changing to something more appropriate and practicable given the diminishing time frame. The inclusion of the demonstration or testing phase within DSR provided the opportunity to examine whether the design matched the intent by examining its functionality in conjunction with potential edge cases, and then, after determining whether it corresponded to the desired solution or not, to communicate any necessary changes.

Consequently, the study has been implemented effectively and accomplished its intended aim. As a result, the research question can be answered due to the implementation of the approach. By adopting a similar structure of design, testing, and evaluation, the data analysis technique of choice in dynamic analysis aligned appropriately with the DSR procedure.

To establish the validity of the study, all produced data was compared with what can be perceived and re-created to validate its authenticity and to ensure that no unknown variables affecting the outcome remained.

In order to ensure the validity of the measurement tool, it has been compared to a similar model and the selection was depending on the knowledge gained from the search done in the research's start. However, it would have been beneficial to compare its performance to that of another model, possibly of a different brand. Similarly, for the purpose of ensuring the reliability factor, dependable and reproducible data sets were sought. In addition, documentation on the project's setup and implementation, as well as its configurations.

6 Conclusions and further research

This chapter summarizes the study's findings and makes suggestions for those who desire to perform more research on this topic.

6.1 Conclusions

The problem statement of this thesis is the lack of information about how to create a framework based on IoT and DT solutions utilizing AWS, with the goal of achieving the optimal interaction and convergence between the physical and virtual environments in a classroom.

In conclusion, the study has contributed to more knowledge in creating the IoT based DT framework for environmental monitoring in the indoor environment. The gained knowledge about configurations of different components in the framework would optimize the interaction and convergence between the physical and virtual environments in a given space. In this study, the configurations were concluded the data collection rate and data transmission rate.

6.1.1 Practical implications

According to (Erol et al., 2020; MarketsandMarkets, 2020; Marmolejo-Saucedo, 2020) mentioned in chapters 1.2 and 3.1, the fact that IoT and DT technologies have a promising and bright future in the field of smart cities indicates that their use will rise in the future in cloud computing.

Specifically, individuals, businesses, and institutions interested in IoT, and DT's solutions and cloud computing may be able to use the results of this study to learn more about designing and implementing a framework using these technologies.

Furthermore, the practical implications for others who wish to accomplish anything like this thesis are substantial. Using the framework the study built and executed, one may add additional sensors for environmental monitoring, such as air quality or overcrowding, and send the collected data to a monitor that displays the data to determine whether the area has a healthy environment or is overcrowded. Thus, a safer environment is maintained throughout the pandemic.

There are numerous further practical implications, but they can also be interpreted as scientific implications, discussed in the next chapter.

6.1.2 Scientific implication

Regarding scientific implications, this thesis can have some implications. For instance, the output data from the framework could be used to demonstrate how the light intensity changes throughout the day so that the placement and number of lamps can be controlled and managed to save resources in a specific space, such as a classroom.

In addition, the framework can be used to monitor temperature fluctuations throughout the day. The heating and air conditioning can be adjusted based on the output readings

from the framework. Thus, it is possible to determine when it is necessary to increase the heating or decrease the air conditioning in a specific space to save energy.

Lastly, since the framework allows for the addition of different sensors, sensors for air quality and humidity can be added. Since the environmental factor has a significant impact, the results can be used to examine the well being of individuals. People may be students, employees, or store customers.

6.2 Further research

This chapter provides examples of research and study questions that could be posed to enhance the framework's configuration and functionality. Time and scope constraints prevented the authors from responding and implementing them.

6.2.1 The environmental monitoring sensors' placement

The environmental sensor is the essential component in the framework where all environmental data is collected. Since this study is not about how these data are accurate rather about how accurate the collection and transmission are in a real-time.

The Fibaro sensor used in the study to collect the environmental data has a specific configuration when it comes to placement to cover the biggest area of a place. Authors get sometimes false temperature and light readings because the sensor was out of range.

In the beginning, a lot of time was spent trying to find the best placement for the sensor using the detection feature built in the sensor to reduce the irrelevant temperature and light readings. The best way to reduce outside temperature and light readings is to position two sensors in opposing locations to cover the entire area.

However, this could be an area that would benefit from further research. Consequently, the following research question could be posed:

- How should an environmental monitoring sensor be positioned geometrically in a classroom to optimize interaction and convergence between the physical and virtual environments?

6.2.2 Utilizing the using of the AWS

The AWS has many properties that could be used to further improve the creation of the framework based IoT and DT technologies. For example, the properties of machine learning (ML) can be used in many ways to enhance the customers' experiences, improve productivity, and optimize the data management process.

The use of machine learning in an environmental monitoring system can improve life in society by highlighting the connection between the environment and health. Transforming environmental monitoring data into information and disseminating actionable insights to the community promptly is essential for informing the public about the state of the environment.

However, this could be an area that would benefit from further research. Consequently, the following research question could be posed:

- How can machine learning be used to optimize the control and management of resources in a Smart Building's environmental monitoring system?

7 References

- Almarabeh, T., Majdalawi, Y. K., & Mohammad, H. (2016). Cloud Computing of E-Government. *Communications and Network*, 08(01), 1-8.
<https://doi.org/10.4236/cn.2016.81001>
- Amazon Managed Grafana. (2022). Amazon web services. Retrieved 2022/07/17 from
https://aws.amazon.com/grafana/?sc_channel=EL&sc_campaign=Demo_Deep_Dive_2021_vid&sc_medium=YouTube&sc_content=Video10138&sc_detail=MANAGEMENT_GOVERNANCE&sc_country=US
- Amazon Timestream (2022). Retrieved 2022/02/25 from
<https://aws.amazon.com/timestream/>
- Amazon Web Services (AWS) (2020). Retrieved 2022/02/25 from
<https://searchaws.techtarget.com/definition/Amazon-Web-Services>
- Andrew Banks, R. G. (2014). *MQTT Version 3.1.1*. OASIS Standard. Retrieved 2022/04/06 from <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>
- AWS IoT Core endpoints and quotas - AWS General Reference. (2022). Retrieved 2022/06/16 from <https://docs.aws.amazon.com/general/latest/gr/iot-core.html>
- AWS IoT Core features (2022). Retrieved 2022/02/25 from
<https://aws.amazon.com/iot-core/features/>
- Bhatt, C., & Shah, T. (2014). The Internet of Things- Technologies, Communications and Computing. *CSI Communications*, 38, 7-9.
- de Kleijn, S. (2020). Real-time Machine Data. *IXON*.
<https://www.ixon.cloud/knowledge-hub/how-real-time-data-can-improve-machines>
- Digital Twin Development and deployment. (2021). Retrieved 2022/02/25 from
<https://www.anylogic.com/features/digital-twin/>
- Dresch, A., Lacerda, D. P., & Antunes, J. A. V. (2015). Design Science Research. In (pp. 67-102). Springer International Publishing. https://doi.org/10.1007/978-3-319-07374-3_4
- Dwivedy, S. K., & Eberhard, P. (2006). Dynamic analysis of flexible manipulators, a literature review. *Mechanism and Machine Theory*, 41(7), 749-777.
<https://doi.org/https://doi.org/10.1016/j.mechmachtheory.2006.01.014>
- Erol, T., Mendi, A. F., & Dogan, D. (2020, 2020). Digital Transformation Revolution with Digital Twin Technology.
- Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital Twin: Enabling Technologies, Challenges and Open Research. *IEEE Access*, 8, 108952-108971. <https://doi.org/10.1109/ACCESS.2020.2998358>
- Grenha Teixeira, J., Patrício, L., Huang, K.-H., Fisk, R. P., Nóbrega, L., & Constantine, L. (2017). The MINDS Method. *Journal of Service Research*, 20(3), 240-258. <https://doi.org/10.1177/1094670516680033>
- Grieves, M. (2016). *Origins of the Digital Twin Concept*.
<https://doi.org/10.13140/RG.2.2.26367.61609>
- Grieves, M., & Vickers, J. (2017). Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In F.-J. Kahlen, S. Flumerfelt, & A. Alves (Eds.), *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches* (pp. 85-113). Springer International Publishing. https://doi.org/10.1007/978-3-319-38756-7_4

- Gunnarsson, R. (2020, 2020). *Validitet och reliabilitet – INFOVOICE.SE*. Retrieved 2022/06/15 from <https://infovoice.se/validitet-och-reliabilitet/>
- Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11), 1016-1022. <https://doi.org/https://doi.org/10.1016/j.ifacol.2018.08.474>
- Lapão, L. V., Da Silva, M. M., & Gregório, J. (2017). Implementing an online pharmaceutical service using design science research. *BMC Medical Informatics and Decision Making*, 17(1). <https://doi.org/10.1186/s12911-017-0428-2>
- Learn - Z-Wave. (2022). Retrieved 2022/5/27 from <https://www.z-wave.com/learn>
- Light Sensors. climax. <https://www.climax.com.tw/new/lm-1zw.php>
- Lo, C. K., Chen, C. H., & Zhong, R. Y. (2021). A review of digital twin in product design and development. *Advanced Engineering Informatics*, 48, 101297. <https://doi.org/https://doi.org/10.1016/j.aei.2021.101297>
- MarketsandMarkets. (2020). Digital Twin Market by Technology, Type (Product, Process, and System), Application (Predictive Maintenance, and Others), Industry (Aerospace & Defense, Automotive & Transportation, Healthcare, and others), and Geography—Global Forecast to 2026. In: MarketsandMarkets Pune, India.
- Marmolejo-Saucedo, J. A. (2020). Design and Development of Digital Twins: a Case Study in Supply Chains. *Mobile Networks and Applications*, 25(6), 2141-2160. <https://doi.org/10.1007/s11036-020-01557-9>
- Mateev, M. (2020). Industry 4.0 and the digital twin for building industry. *Industry 4.0*, 5(1), 29-32.
- McClelland, C. (2020, January 10, 2020). *What is an IoT Platform*. Retrieved 2022/02/10 from <https://www.iotforall.com/what-is-an-iot-platform>
- Motion Sensor. (2021). Retrieved 2022/04/07 from <https://www.fibaro.com/se/products/motion-sensor/>
- Peffer, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3), 45-77. <https://doi.org/10.2753/mis0742-1222240302>
- Radouan Ait Mouha, R. A. (2021). Internet of Things (IoT). *Journal of Data Analysis and Information Processing*, 09(02), 77-101. <https://doi.org/10.4236/jdaip.2021.92006>
- Raspberry Pi beginner's guide 4th edition. (2022). <https://magpi.raspberrypi.com/books/beginners-guide-4th-ed>
- Saud Albazei, S. (2021). What to Consider when Selecting a Region for your Workloads \textbar Amazon Web Services.
- Singh, M., Fuenmayor, E., Hinchey, E., Qiao, Y., Murray, N., & Devine, D. (2021). Digital Twin: Origin to Future. *Applied System Innovation*, 4(2), 36. <https://doi.org/10.3390/asi4020036>
- What is a digital twin? (2019). IBM. Retrieved 2022/04/19 from <https://www.ibm.com/topics/what-is-a-digital-twin>
- Z-Stick Gen5+. (2021). Retrieved 2022/04/07 from <https://aeotec.com/z-wave-usb-stick/index.html>
- Zaballos, A., Briones, A., Massa, A., Centelles, P., & Caballero, V. (2020). A Smart Campus' Digital Twin for Sustainable Comfort Monitoring. *Sustainability*, 12(21). <https://doi.org/10.3390/su12219196>

Zhan, X., & Walker, S. (2019). Craft as Leverage for Sustainable Design Transformation: A Theoretical Foundation. *The Design Journal*, 22, 503-523.
<https://doi.org/10.1080/14606925.2019.1613040>

8 Appendices

APPENDIX 1

Preparing the RPi and the OpenHAB dashboard install OS (OpenHABian)

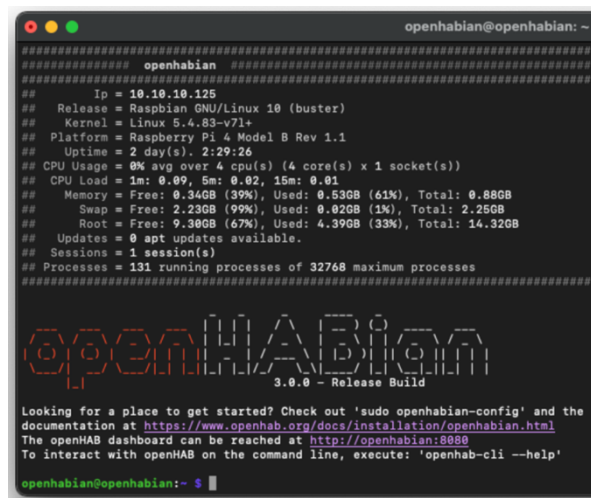
Requirements for installation:

1. microSD card (4GB minimum, 8GB recommended).
2. a computer with a microSD card drive.
3. a monitor with an HDMI interface.
4. an HDMI cable.
5. a USB keyboard and mouse.
6. A stable internet connection.

Guidelines for installation:

1. Download the latest "openHABian" SD card image (image means a copy of OS).
2. Write the image to the SD card using the Raspberry Pi Imager or the Etcher.
3. Insert the SD card into your Raspberry Pi.
4. Connect RPi to Ethernet if there is no Wi-Fi or configure Wi-Fi.
5. Power on and wait approximately 15-45 minutes.

When openHABian has been installed and configured a welcome screen is shown as in the Figure below.



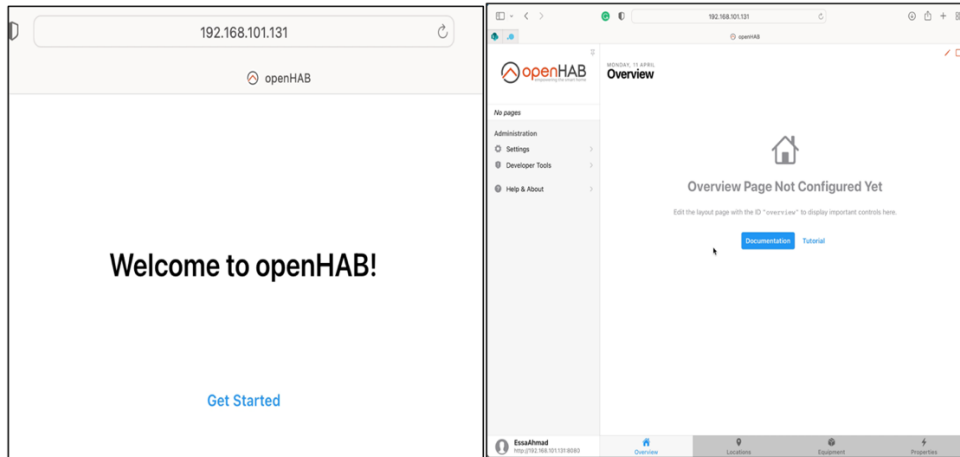
```
openhabian@openhabian: ~  
===== openhabian =====  
##  
##   Ip = 10.10.10.125  
##   Release = Raspbian GNU/Linux 10 (buster)  
##   Kernel = Linux 5.4.83-v7l+  
##   Platform = Raspberry Pi 4 Model B Rev 1.1  
##   Uptime = 2 day(s). 2:29:26  
##   CPU Usage = 0% avg over 4 cpu(s) (4 core(s) x 1 socket(s))  
##   CPU Load = 1m: 0.09, 5m: 0.02, 15m: 0.01  
##   Memory = Free: 0.34GB (39%), Used: 0.53GB (61%), Total: 0.88GB  
##   Swap = Free: 2.23GB (99%), Used: 0.02GB (1%), Total: 2.25GB  
##   Root = Free: 9.38GB (67%), Used: 4.39GB (33%), Total: 14.32GB  
##   Updates = 0 apt updates available.  
##   Sessions = 1 session(s)  
##   Processes = 131 running processes of 32768 maximum processes  
##  
===== openhabian =====  
openhabian 3.0.0 - Release Build  
Looking for a place to get started? Check out 'sudo openhabian-config' and the  
documentation at https://www.openhab.org/docs/installation/openhabian.html  
The openHAB dashboard can be reached at http://openhabian:8080  
To interact with openHAB on the command line, execute: 'openhab-cli --help'  
openhabian@openhabian:~$
```

configure the OpenHAB dashboard

Once the RPi is ready OpenHAB dashboard configuration can be completed by following the steps below.

1. Connect the computer to the same Wi-Fi the RPi is connected
2. Access the openHAB dashboard via <http://openhabian:8080> or <http://the-computer-IP:8080>.
3. create an account to manage the openHAB dashboard.
4. choose the language, region, location, and time zone.
5. install the necessary add-ons in the openHAB dashboard.

When all the configuration steps have been completed, a welcome screen is shown as in the figure below.



APPENDIX 2

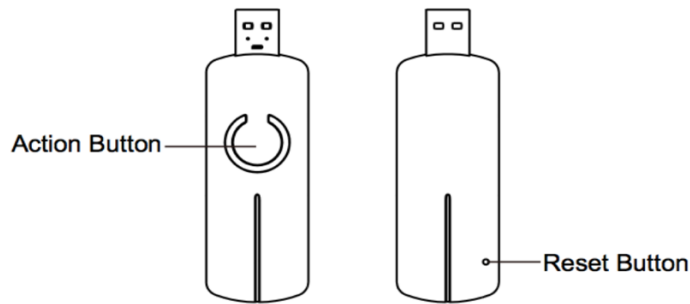
Configuration Z-Wave stick and Motion sensor

Configuration of the Z-Wave stick

Z-Wave stick acts as a controller for the automation system in the indoor environment, and it keeps environmental monitoring sensors connected to the RPi. It will be used in the framework because it provides improved battery life, range, and bandwidth. Up to 232 devices supported by z-wave technology can be connected to it, and it is straightforward to pair new devices to the Z-Wave stick (*Z-Stick Gen5+*, 2021). It can be configured by following the steps below.

1. Installing the necessary bindings like z-wave binding.
2. Switch on the Z-Wave stick using the action button.
3. Connect the Z-Wave stick to the RPi.
4. Add devices "sensors" to the Z-Wave stick by following these steps:
 - 4.1 remove the Z-Wave stick from the RPi.
 - 4.2 Tap the action button on the Z-Wave stick twice quickly.
 - 4.3 Now the Z-Wave stick blinks rapidly to indicate the Z-Wave is ready to pair.
 - 4.4 Switch on the Sensor desired to connect.
 - 4.5 If the pairing is done LED on the Z -Wave stick becomes blue solid for 2 seconds.
 - 4.6 If the pairing is failed LED on the Z-Wave stick becomes red solid for 2 seconds.
 - 4.7 Once all paring is done, tap the Z-Sticks button once, and the LED should stop blinking.
 - 4.8 Repeat steps 4.2 – 4.6 if more devices are desired to be connected.
5. Connect the Z-Wave stick again to the RPi.

The figure below shows the buttons of the Z-Wave stick using in the configuration and connection process.



The table below shows a detailed description of the different colors and functions of the Z-Wave stick and the buttons needed to complete the pairing process.

LED Color	Unplugged / Plugged	Description of function
No LED	Unplugged	No activity, z-wave management mode
Slow Blue blink	Unplugged	Pair/inclusion mode activate
Fast Amber blink	Unplugged	Unpair/exclusion mode active
Red blinking LED	Unplugged, RESET button held	Indicates that it has entered factory reset mode. Will only reset if RESET button is held for a full 20 seconds (followed by the confirmation LED indicator). Factory reset does not happen if released earlier than 20 seconds.
Solid blue for 2 seconds	Unplugged	Confirmation of success/communication for pair, unpair, or factory reset.
Solid Amber Solid Green	Plugged into USB	Normal activity Amber LED = Charging battery Green LED = Fully charged
Blue Flash	Plugged into USB	Indicates that Z-Stick Gen5 has communicated a Z-Wave command.

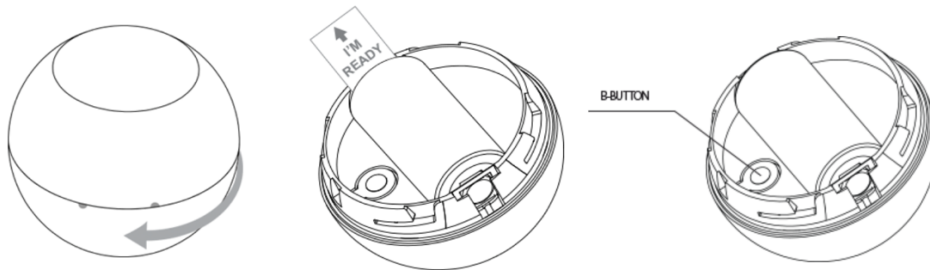
Configuration of the environmental Sensor

The connection between the Sensor and RPi can be established in two phases. The first phase is to connect the Sensor to the z-wave stick. The following steps guide establishing the Sensor's connection to the z-wave.

1. Open the Sensor and remove the plastic strip to turn on the Sensor.
2. Bring the Sensor directly within the range of the z-wave stick when it is unplugged into the RPi.
3. Press once on the button on the front side of the stick (z-wave stick) till it blinks with blue light (learning modes).
4. Place the stick near the Sensor and then press three times on the button on the backside of the Sensor till the Sensor is recognized.

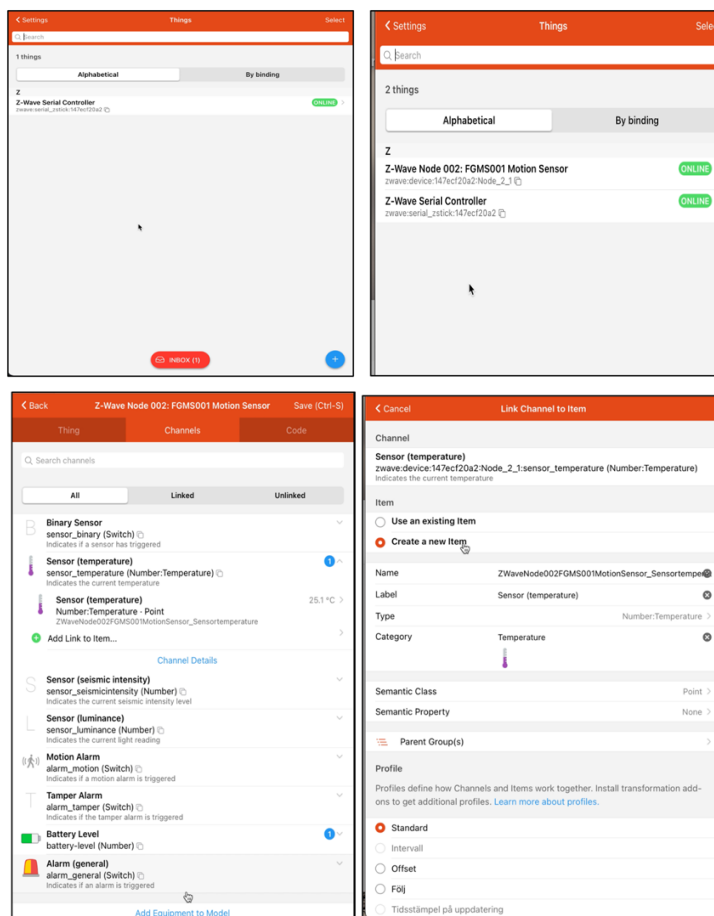
- Both the Sensor and the stick will blink to show connected.

The figures below show how to start on the Fibaro sensor.



The second phase is to configure the Sensor in the openHAB dashboard, and the guidelines for configuration are the following.

- Connect the stick by plugging it into the RPi.
- Open the notification in the inbox.
- Follow the link in the notification to the Sensor's feature interface.
- Choose the light intensity and temperature features.
- Link the chosen features with an item.
- Open the openHAB dashboard and connect the item to the user interface to show the measured environmental data.



APPENDIX 3

Configuration the connection between RPi and AWS

The connection between RPi and AWS consists of two distinct components. The first step is to register the device with AWS IoT Core, and the second is to run the Connection Kit using the security credentials.

Registering the IoT device in the AWS IoT Core

The IoT device in AWS IoT core can be configured by following the steps below.

1. Select onboard a device on the "AWS IoT/Connect to AWS IoT" page.
2. Because the RPi in the thesis is running OpenHABian, which is a Linux distribution, you can use Linux as a platform.
3. Since the OS of the device has Python and Git installed and a TCP connection to the public Internet on port 8883, python would be the excellent choice between to choose as the AWS IoT Device Software Development Kit.
4. Name the device and then download the connection kit.

Run the Connection Kit

To run the connection kit, open the terminal in the directory the Connection Kit was installed in and do the following steps below.

1. The following command will unzip the device's connection kit.

```
unzip connect_device_package.zip
```

2. The following command will grant access to the executable file.

```
chmod +x start.sh
```

3. Run the start script below. Messages from the local device will appear in the IoT core user interface.

```
./start.sh
```

APPENDIX 4

A Configuration the connection between AWS IoT core and Amazon Timestream.

The connection between AWS IoT Core and Amazon Timestream consists of three distinct components. The first step is to register the database, the second is to create a database table in Amazon Timestream, and the third is to configure AWS IoT Core rule to send the data to the database table.

Creating a database in the Timestream.

The Amazon Timestream database is configured by following the steps below.

1. On the "Amazon Timestream/ Databases" page, click Create a Database.

2. For the configuration, choose a standard database.
3. Give the database a name.
4. Assign a Key Management Service (KMS) key to the database.
5. Select the Create Database option.

The user interface of create a database page below.

The screenshot displays the 'Create database' page in the AWS Timestream console. The breadcrumb trail at the top reads 'Timestream > Databases > Create database'. The main heading is 'Create database' with an 'Info' link. Below this, the 'Database configuration' section explains the purpose of the page. Under 'Choose a configuration', the 'Standard database' option is selected, indicating a new database with custom configuration. The 'Sample database' option is also visible. The 'Name' field is populated with 'myDatabase'. The 'Encryption' section shows the 'KMS key' dropdown set to 'aws/timestream'. The 'Tags - optional' section indicates no tags are currently associated with the database. At the bottom right, there are 'Cancel' and 'Create database' buttons.

Creating a database table in the Amazon Timestream.

The Amazon Timestream database table can be configured by following the steps below.

1. Click Create a Table on the "Amazon Timestream/ Tables" page.
2. Choose the database in which the table will be created.
3. Give the table a name.
4. Determine how long data will be stored in memory before being transferred to the magnetic store.

5. Define the amount of time data will be stored in the magnetic store before being deleted.
6. Choose the option to create a table.

[Timestream](#) > [Tables](#) > Create table

Create table

Table details

Database name
Choose the database where this table will be created.

Table name
Specify a table name that is unique within this database. You can not change this name once you create it.

Must be between 3 and 256 characters long. Must contain letters, digits, dashes, periods or underscores.

Data retention
Specify how long your data is retained in each storage tier. Data moves from the memory store to the magnetic store as it ages. Data that exceeds the magnetic store retention will be deleted.

Memory store retention
Specify how long data will be stored in the memory store before it is moved to magnetic store.

The value must be a number. Minimum 1 hour, maximum 12 months.

Magnetic store retention
Specify how long data will be stored in the magnetic store before it is deleted.

The value must be a number. Minimum 1 day, maximum 200 years.

Magnetic Storage Writes

Settings
☐ Enable magnetic storage writes

Tags - optional
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags are associated with this table.

You can add 50 more tag(s).

Creating a IoT Core rule.

The IoT core rule can be configured by following the steps below.

7. On the "AWS IoT/ rules" page, click Create.
8. Assign a name to the rule.
9. Locate the SQL statement's measure values.
10. Create a Timestream action method.
11. Make a dimension out of the sensor id.
12. Make your rule.

Create a rule

Create a rule to evaluate messages sent by your things and specify what to do when a message is received (for example, write data to a DynamoDB table or invoke a Lambda function).

Name

Description

Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see [AWS IoT SQL Reference](#).

```
1 SELECT temperature FROM 'iot/topic'
```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*required)



Write a message into a Timestream table
DigitalTwinDB

Remove Edit ▶

Add action

Error action

Optionally set an action that will be executed when something goes wrong with processing your rule.

Add action

APPENDIX 5

A Configuration the connection between Amazon Timestream and Grafana.

Grafana is the final component of the framework, and it serves as the data visualization tool. A connection must be established between Grafana and Amazon Timestream in order to retrieve data.

The following steps guide the process of implementation and connection:

1. Connect the Grafana to the AWS by providing the credentials details as shown in the figure below

The screenshot shows the Grafana Settings page for a connection named 'Amazon Timestream-1'. The page is divided into two main sections: 'Connection Details' and 'Timestream Details'.

Connection Details:

Authentication Provider	Access & secret key
Access Key ID	
Secret Access Key	
Assume Role ARN	arn:aws:iam:*
External ID	External ID
Endpoint	https://query-(cell).timestream.(region).amazonaws.com
Default Region	Choose

Timestream Details:

Default values to be used as macros

Database	Choose
Table	Choose
Measure	Choose

At the bottom of the page, there are four buttons: 'Back', 'Explore', 'Delete', and 'Save & test'.

2. When the connection is successful it is time to select the database, table and measure values.
3. In the Grafana Dashboard a line graph can be selected to visualize temperature values.
4. When it is done the following will be showed

