BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

University of Zurich[UZH]

Secure IoT Network Prototyping for Artwork Tracking

Mohamed Islem Mdimagh
Zürich, Switzerland
Student ID: 20-706-735

Supervisor: Katharina O.E. Müller, Thomas Grübl, Prof. Dr. Burkhard Stiller
Date of Submission: May 4, 2024

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich, 04.05.2024

Signature of student

Zusammenfassung

Diese Arbeit untersucht die Integration von Internet of Things (IoT)-Sensoren in die Logistik von Kunsttransporten, um die Sicherheit zu erhöhen und eine Standardisierung in einem unregulierten Markt einzuführen. Sie beinhaltet die Implementierung einer Cloud-basierten Service-Infrastruktur, die IoT-Sicherheit und -Kommunikation während des Kunsttransports durch Long Term Evolution (LTE) unterstützt und über ein Virtual Private Network (VPN) gesichert ist. Der Schwerpunkt liegt auf der Entwicklung eines Prototyps für ein System zur Erfassung und Protokollierung von Kunstwerken, das den am Transport von Kunstwerken beteiligten Akteuren präzise Messungen liefert. Das System soll die Umgebungsbedingungen rund um das Kunstwerk in Echtzeit überwachen und so dessen Sicherheit und Unversehrtheit während des Transports gewährleisten. Die Arbeit beschreibt die Forschungs-, Design-, Implementierungs- und Evaluierungsphasen des Projekts und befasst sich mit den aktuellen Standards und Praktiken im Kunsttransport, der Effektivität des Prototyps und der zukünftigen Arbeit in diesem Bereich.

Das System, das ursprünglich für die Nutzung von Cloud-Diensten und Mobilfunkverbindungen konzipiert war, stiess auf Probleme mit den Netzwerkverbindungen in der Schweiz. Daher wurde eine lokale Lösung unter Verwendung des STMicroelectronics B-L462E-CELL1 Cellular Discovery Kit implementiert. Dieser lokale Prototyp misst Temperatur, Luftfeuchtigkeit und Beschleunigung und zeigt die Daten zur Echtzeitüberwachung direkt auf einem angeschlossenen Laptop. Trotz der Abkehr von der beabsichtigten Fernüberwachung hat der lokale Prototyp seine Robustheit und Zuverlässigkeit in verschiedenen simulierten Transportszenarien bewiesen und damit sein Potenzial für reale Anwendungen im Kunsttransport unter Beweis gestellt. Zukünftige Arbeiten zielen auf die Lösung von Konnektivitätsproblemen und die Implementierung des ursprünglichen Fernüberwachungsdesigns ab.

Abstract

This thesis explores the integration of Internet of Things (IoT) sensors into the logistics of art transportation to enhance security and introduce standardization in an unregulated market. It involves the implementation of a cloud-based service infrastructure, supporting IoT security and communication during art transport through Long Term Evolution (LTE) and secured via a Virtual Private Network (VPN). The focus is on developing a prototype artwork sensing and logging system that delivers precise measurements to stakeholders involved in the transportation of artworks. The system aims to monitor environmental conditions surrounding the artwork in real time, ensuring its safety and integrity during transit. The thesis outlines the research, design, implementation, and evaluation stages of the project, addressing current standards and practices in art transport, the effectiveness of the prototyped system, and future work in the field.

Initially designed to utilize cloud services and cellular connectivity, the system faced challenges with network connections in Switzerland. Consequently, a local solution was implemented using the STMicroelectronics B-L462E-CELL1 Cellular Discovery Kit. This local prototype measures temperature, humidity, and acceleration, and logs data directly to a connected laptop for real-time monitoring. Despite the shift from the intended remote monitoring, the local prototype demonstrated robustness and reliability in various simulated transportation scenarios, proving its potential for real-world applications in art transport. Future work aims to resolve connectivity issues and implement the initial remote monitoring design.

Acknowledgments

I would like to express my sincere gratitude to Katharina O.E. Müller for her support and guidance throughout this project. Her expertise and dedication have greatly contributed to the success of this work.

I extend my thanks to Thomas Grübl for his assistance, which has enriched my understanding of the subject matter. I am also grateful to the Communication Systems Group, and Prof. Dr. Burkhard Stiller. Their expertise and feedback, which has significantly influenced the development of this thesis.

I would like to acknowledge the support of my friends and family throughout this journey. Their encouragement and understanding have been a constant source of motivation.

Contents

Declaration of Independence **i**

Zusammenfassung **iii**

Abstract **v**

Acknowledgments **vii**

1 Introduction **1**

1.1 Motivation 1

1.2 Description of Work 2

1.3 Thesis Outline 2

2 Related Work **3**

2.1 Artwork Tracking 3

2.2 Long Term Evolution 3

2.3 IoT Security 4

2.4 Discussion 4

3 Background Knowledge **5**

3.1 Internet of Things 5

3.2 Microcontroller 6

3.3 Wireless Communication Technologies 8

3.4 Cloud Computing 8

3.5 Message Queue Telemetry Transport 9
3.6 Artwork Tracking 10
3.7 Security 10

4 Design 13

4.1 Scenario 13
4.2 Goal 14
4.3 Actors 14
4.4 Technical Components 15
4.5 Overview 15
4.6 Testing 16

5 Implementation 19

5.1 Sensor Data 19
5.1.1 Setup 19
5.1.2 Source Code 21
5.2 Cellular Connection 24
5.3 Local Monitoring 26

6 Evaluation 33

6.1 Experiment 0 33
6.2 Experiment 1 34
6.3 Experiment 2 36
6.3.1 Experiment 2a 37
6.3.2 Experiment 2b 37
6.4 Experiment 3 38
6.4.1 Experiment 3 Fail 39
6.4.2 Experiment 3 Second Attempt 40
6.5 Discussion 40

7 Summary and Conclusions **43**

7.1 Summary 43

7.2 Conclusion 44

7.3 Future Work 44

Bibliography **45**

Abbreviations **51**

List of Figures **54**

List of Tables **55**

A Installation Guidelines **57**

Chapter 1

Introduction

The art market is a diverse ecosystem involving various stakeholders, from artists and collectors to auction houses, art dealers, and various intermediaries such as promoters, preservers, and curators. Despite this diversity, the core of the art world revolves around art objects, ranging from traditional sculptures and canvases to more unconventional pieces like bananas affixed to walls. Museums often showcase artworks from private collections, relying on the involvement of reliable logistics partners to safely transport and preserve these valuable pieces while monitoring environmental conditions such as temperature and humidity.

1.1 Motivation

Integrating Internet of Things (IoT) sensors into the logistics of art transportation presents a significant opportunity to enhance security and introduce standardization into what is currently an unregulated market. As such, it requires the implementation of a cloud-based service infrastructure, supporting IoT security and communication during art transport. This communication shall be enabled through Long Term Evolution (LTE) and secured through a Virtual Private Network (VPN).

During transportation, the environmental conditions surrounding the artwork are continuously monitored, with real-time updates broadcasted to a cloud gateway. To enable this, each truck has a designated ST-Board, which uses sensors to measure the environmental data and to facilitate communication from the truck to the cloud-based gateway.

The primary focus of this thesis lies in the communication between transportation and museum entities, with the use of ST-board-based technology. Therefore, the primary goal is to develop and implement a prototype artwork sensing and logging system that can deliver precise and accurate measurements to all involved stakeholders.

1.2 Description of Work

This Bachelor Thesis consists of three stages:

Research: The initial stage is dedicated to research, where related works are reviewed and essential background knowledge is acquired. This stage involves familiarizing with the artwork use case and defining requirements for both sensing and communication. Existing solutions, approaches, and their components will be evaluated, and their applicability, performance, and security will be considered. This stage also necessitates research into LTE communication and VPN, as they form the basis of communication. The goal is to gain sufficient insight into the theoretical aspects of this work, enabling the conceptualization of feasible approaches and improvements over existing solutions. Additionally, the capabilities of the ST-board, the available sensors, and potential additions will be explored.

Design: The second stage uses the knowledge gained from the first stage to design a prototype and plan experiments to thoroughly evaluate the logging functionality and behavior, as well as the communication paths and their stability. In this stage, the artwork transportation scenario with its actors is shown, the technical components are listed and described and an overview of the system is presented.

Implementation and Evaluation: The final stage involves implementing the prototype based on the design, considerations, and limitations determined in the first two stages. This stage also includes conducting experiments to assess the reliability of the implementation and evaluating its overall reliability, limitations, and potential expansions.

In addition to these stages, this thesis will address the following research questions:

- What are the current standards and practices in the art transport industry for monitoring environmental conditions?
- How effective is the prototyped system in real-world scenarios?

1.3 Thesis Outline

This thesis is structured as follows:

Chapter two delves into a review of various approaches aiming to address the artwork monitoring problem. In chapter three, the theoretical knowledge necessary for understanding the thesis is presented. Chapter four explains the scenario, including its system design and testing procedures. The implementation of the system is shown in chapter five. Chapter six conducts testing and evaluation of the implemented system. Finally, chapter seven provides a concise summary of the thesis and gives an outlook for future work.

Chapter 2

Related Work

This chapter presents an overview of the literature surrounding IoT monitoring systems, IoT in the art industry, LTE, and IoT security. Showing the current standards and practices in the art transport monitoring industry, and comparing this thesis to them.

2.1 Artwork Tracking

[20, 30, 19, 12] discuss IoT monitoring systems designed for tracking temperature and humidity. [30] focuses on monitoring the temperature within greenhouse environments, aiming to improve agriculture production. [19] is dedicated to monitoring the environmental data within a data center, while [12] specializes in monitoring the conditions within food storage facilities.

[24, 3, 10, 1, 6] delve into IoT monitoring solutions tailored for buildings within the art industry. [24, 3, 10] are dedicated to monitoring Heritage Artefacts, while [6] focuses on ancient wood structures. Additionally, [6] not only monitors artwork within local museum settings but also provides remote monitoring capabilities for artwork on the move. Furthermore, [39] addresses the need for a transport monitoring system aimed at protecting cultural relics during transit.

2.2 Long Term Evolution

Many IoT systems require connectivity to the internet through a cellular network. [4] talks about LTE-M and narrow-band IoT and provides an overview of their evolution. [21] provides an overview of LTE enhancements for machine-type communications, alongside an analysis of LTE capacity for machine traffic and an evaluation of device battery life.

2.3 IoT Security

Interestingly, not a lot of these papers address security concerns. In contrast, [9, 16] delves into the security aspects of IoT devices, showing vulnerabilities and corresponding solutions for each of the three layers of the IoT device architecture. [7] provides a comprehensive survey of machine learning and deep learning methods that can be used to enhance the security of IoT systems, discussing the opportunities, advantages, and challenges of these methods. [17] focuses on the vulnerabilities, providing a classification of related surveys, a unique taxonomy of IoT vulnerabilities, and a first look at Internet-scale IoT exploitations.

2.4 Discussion

Paper	Focus Area
[19, 29, 18, 11]	IoT Monitoring Systems
[23, 3, 10, 1, 6]	IoT in Art Industry
[38]	Transport Monitoring
[4, 20]	LTE
[9, 15, 7, 16]	IoT Security
This Thesis	Secure IoT Transport Monitoring System using LTE

Table 2.1: Paper Overview

As shown in Table 2.1, prior research has concentrated on various aspects of IoT, including monitoring systems, its application in the art industry and transportation, LTE utilization for IoT, and security concerns regarding IoT devices. This thesis delves into these domains, emphasizing the need for precise environmental monitoring to ensure the safety and preservation of artworks during transit.

Chapter 3

Background Knowledge

This chapter describes the theoretical background used in this thesis. It covers key aspects such as IoT, microcontrollers, wireless communication, cloud computing, Message Queue Telemetry Transport (MQTT), and their application in artwork tracking. Additionally, it addresses IoT security.

3.1 Internet of Things

IoT was first mentioned by the British technology pioneer Kevin Ashton in 1999 to describe a system, where sensors connect objects of the physical world to the internet [22]. Today, the term is used to describe scenarios where internet connectivity and computing capabilities extend to a range of objects, devices, sensors, and everyday items [22]. For instance, new IoT products such as internet-connected appliances, home automation components, and energy management devices are steering us towards a ”smart home”, offering enhanced security and energy efficiency [22]. Other personal IoT devices, like wearable fitness and health monitoring devices, are revolutionizing the delivery of healthcare services [22]. This technology is particularly promising for individuals with disabilities and the elderly, as it provides improved independence and quality of life at a reasonable cost [22].

As shown in Figure 3.1 the number of IoT devices worldwide is projected to nearly double from 15.1 billion in 2023 to over 29 billion by 2030 [38]. On one side, the increase in IoT devices is expected to offer economic and social benefits, including cost savings, value creation, productivity enhancements, and overall economic growth [36]. On the other, this growth could lead to a more sinister landscape filled with surveillance, privacy and security breaches, and consumer lock-in [22].

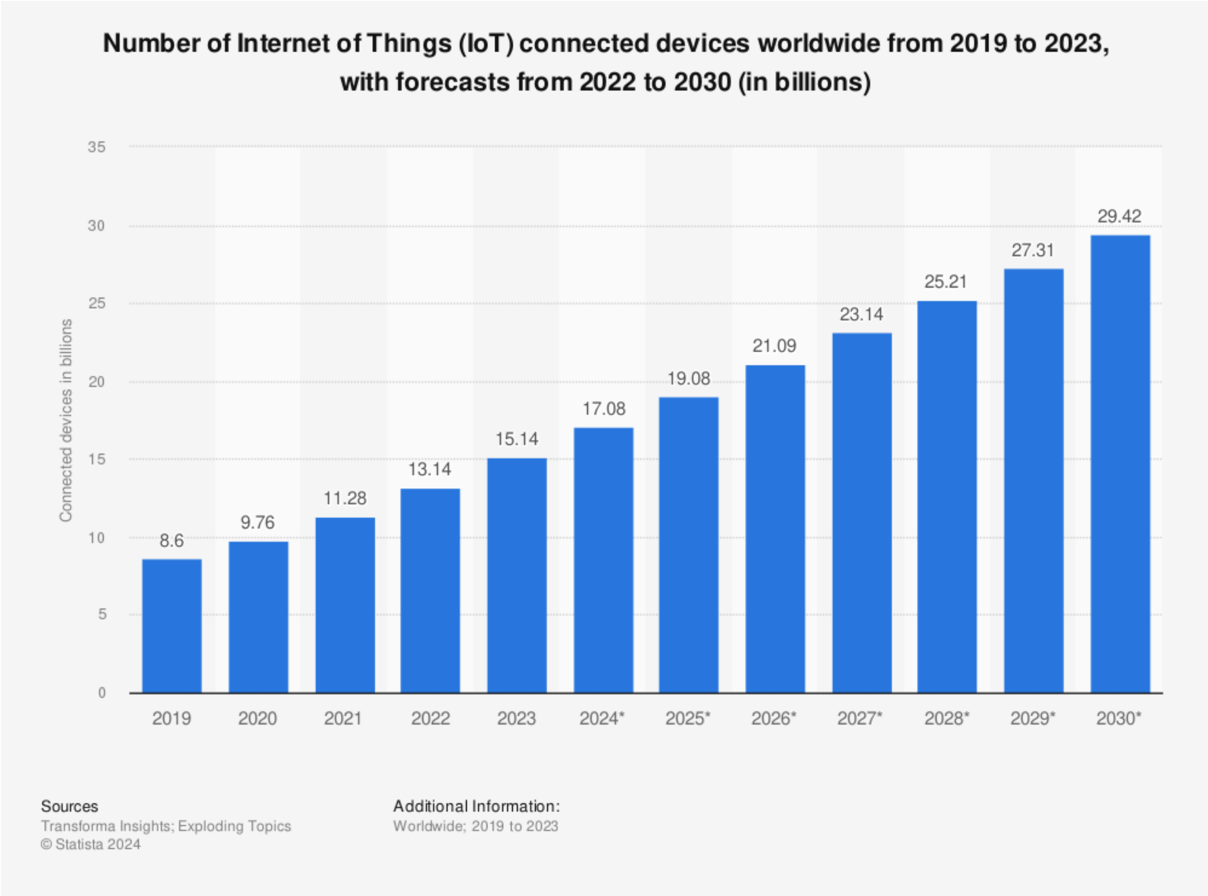

Figure 3.1: Number of IoT connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030, source: [26]

3.2 Microcontroller

What do mobile phones, control devices for automobiles, and monitoring and control systems for industrial automation have in common? - They are all embedded systems [15]. Unlike general-purpose personal computers, embedded systems have specific requirements with pre-defined tasks to perform [15]. The most popular type of an embedded system is a microcontroller, a compact computer system encapsulated within a single integrated circuit [14].

STM32 microcontrollers are a range of microcontrollers that are divided into nine sub-families, and each one has its own set of features [18]. What they all do have in common is their Cortex-M core [18]. Cortex-M-based processors are a span of scalable, compatible, energy efficient, and easy-to-use processors designed for the low-cost-embedded market, suitable for IoT, connectivity, and smart metering applications [18]. They also consist of a static RAM, flash memory, and debugging interface [18]. There are some advantages for embedded developers to use the STM32 platform [18]. Because of its Cortex-M-based processors and ARM's position in the embedded market, numerous tools are accessible to developers with ARM-based tool chains being completely free [18].

The STMicroelectronics B-L462E-CELL1 Cellular Discovery Kit plays an important role in bringing IoT applications to life, especially those that require cellular connectivity and embedded sensor functionalities [29]. The B-L462E-CELL1 board combines STM32 microcontroller technology with LTE cellular connectivity, allowing for the collection of sensor data and its transmission to remote cloud-based servers for further analysis and processing [29]. Within the STM32 family, the B-L462E-CELL1 board utilizes the STM32L4 series microcontroller, which is known for its ultra-low-power capabilities [29]. This microcontroller unit enables the connection of various sensors, such as the HTS221 which is shown in Figure 3.2. An ultra-compact sensor for measuring relative humidity from 20 to +80% rH, with an accuracy of +- 3.5% rH, and temperature from +15 to +40°C, with an accuracy of +- 0.5°C [27, 29]. Another sensor is the LSM303AGR, which is an ultralow-power high-performance system-in-package with a 3-axis digital linear acceleration sensor, which measures the acceleration on the x-, y- and z-axis and returns the data in mg [28]. Furthermore, the integration of LTE cellular connectivity, by utilizing its built-in eSIM or an extern SIM by using the SIM card slot shown in Figure 3.2, provides a method for transmitting sensor data to cloud-based servers. This feature is essential for applications that require real-time or periodic transmission of sensor data to remote servers for monitoring, and analysis[29].

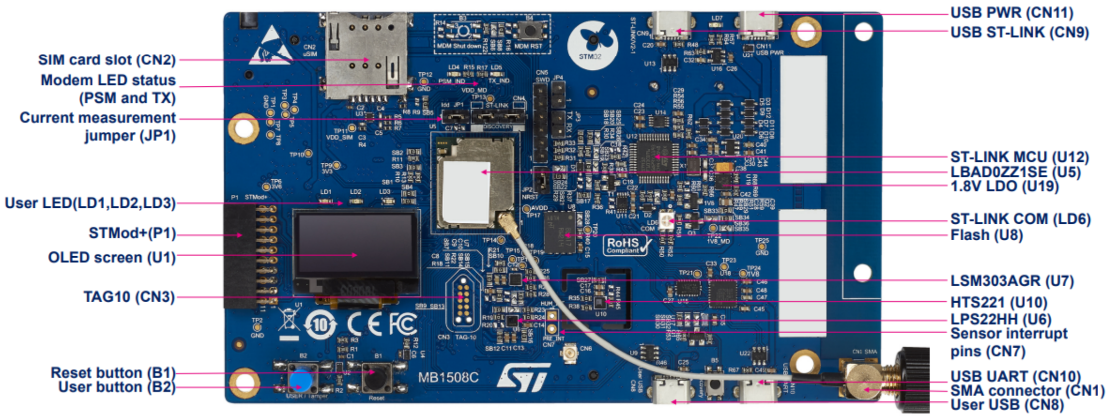

Figure 3.2: B-L462E-CELL1 Discovery kit (top view), source: [29]

Figure 3.3 illustrates that the B-L462E-CELL1 board can also be battery-operated, making it useful for systems on the move where access to power sources may be limited or unavailable.

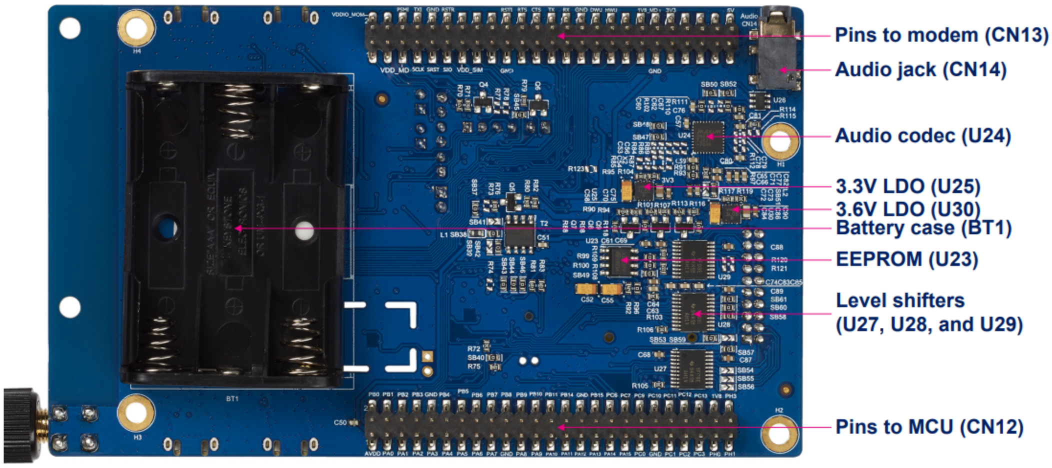

Figure 3.3: B-L462E-CELL1 Discovery kit (bottom view), source: [29]

3.3 Wireless Communication Technologies

LTE is a standard for wireless broadband communication for mobile devices and data terminals, developed by the 3rd Generation Partnership Project (3GPP) [37, 34]. It is the evolution of the Universal Mobile Telecommunication System (UMTS), which evolved from the Global System for Mobile Communications (GSM) [37].

LTE provides high data rates, low latency, and a packet-optimized radio access technology that supports flexible bandwidth deployments [37]. It is an ideal technology for supporting high data rates for services such as Voice over IP (VoIP), streaming multimedia, video conferencing, or as a high-speed modem for mobile devices [37]. LTE uses both Time Division Duplex (TDD) and Frequency Division Duplex (FDD) modes [37].

In the context of IoT, LTE provides a reliable connection for IoT devices to complete data transfers to the cloud and other devices [33]. It also enables cellular LPWA (Low Power Wide Area) networks (e.g., Cat M and NB-IoT) to operate on its infrastructure [33].

LTE-M, also known as LTE Cat-M1, is a type of Low-Power Wide-Area Network (LPWAN) radio communication technology developed by 3GPP for machine-to-machine and IoT applications [31]. LTE-M was designed for cost-effective IoT applications, using low data rates, requiring long battery life, and often operated in locations that are hard to reach [32].

3.4 Cloud Computing

Cloud computing is when computing resources are delivered over the internet [35]. It allows access to computing resources, such as storage and infrastructure, which removes the need for individuals to manage physical resources, such as servers, independently, allowing them to pay only for what they use [8]. There are three types of cloud computing, public -, private -, and hybrid clouds [8]. Public Clouds are made available by third parties, offering computing, storage, and network resources over the internet [8]. Private clouds are privately hosted in their data centers, which makes them more secure [8]. Hybrid clouds combine the other two types, using the advantages of both [8].

One example of a public cloud is AWS IoT Core, which lets connected devices interact with cloud applications and other devices [5]. It offers multiple communication protocols, including MQTT and HTTPS [2]. It also provides secure communication between devices by using authentication and end-to-end encryption [2].

3.5 Message Queue Telemetry Transport

A protocol is a special set of rules and regulations in a telecommunication connection, that endpoints use to communicate to other endpoints connected to the same network [13]. One of these protocols is MQTT, a standardized publish/subscribe protocol, released by IBM in 1999 [25]. It was planned to send data accurately under long network delays and low-bandwidth network conditions [25].

As Figure 3.4 illustrates, several devices (subscribers) subscribe to a topic, for example, ”temp”, on an MQTT broker [23]. A publisher, such as an IoT device, can then publish its data, such as the measured temperature, to the broker [23]. The broker then traverses his subscription list and sends the data to the subscribers of ”temp” [23].

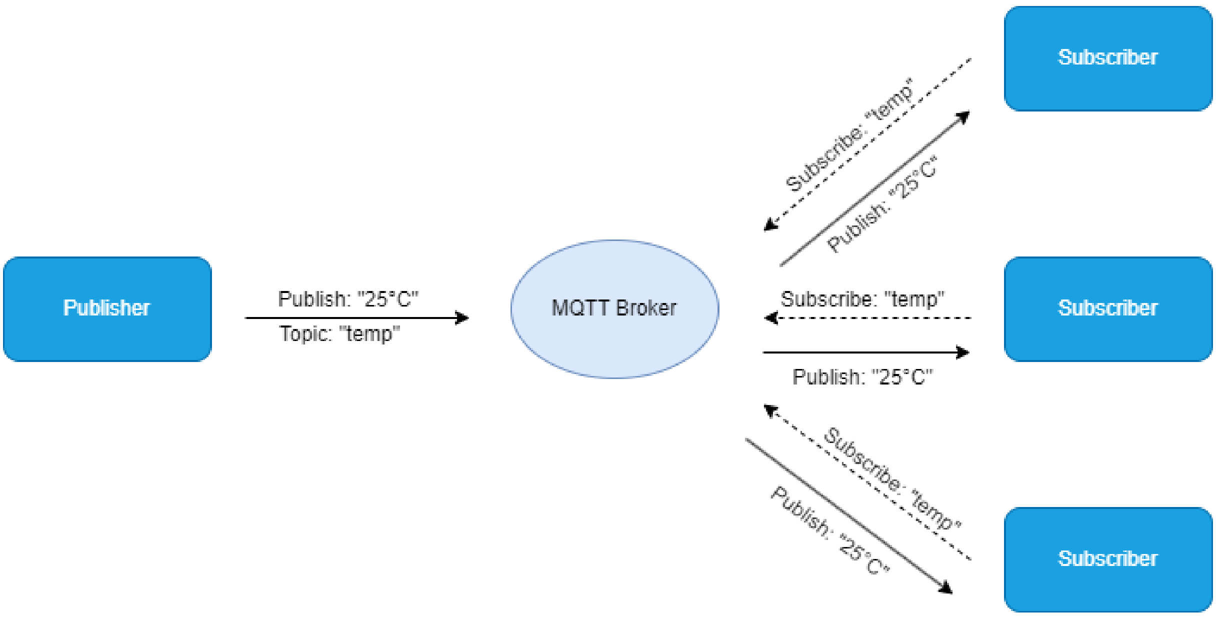

Figure 3.4: Diagram illustrating the operation of the MQTT protocol

MQTT also has its limitations, one of them being message expiration, where messages that are not gathered from the broker will stay there forever, resulting in an overload of the broker, which degrades the overall performance [25]. While message delivery is guaranteed, there is no guarantee that the order of the messages is correct, making it important to add timestamps to the messages [25].

3.6 Artwork Tracking

Artwork monitoring is one of many IoT applications. It involves IoT devices to track and monitor environmental parameters such as temperature and humidity, which are crucial for the preservation of artwork. Several studies have proposed IoT monitoring systems.

For instance, [6] discusses their work on health monitoring for artwork and large wood structures. They employed an STM32 to measure temperature, humidity, and tilt. They explored two scenarios: one static and one mobile. In the static scenario, measurement nodes transmit data via BLE to BLE Gateways, which connect to a Modem or a Switch [6]. From there, the data is sent either over the internet or the local network to a remote cloud and database or a local server [6]. In the mobile scenario, measurement nodes send data via Bluetooth Low Energy (BLE) to a smartphone or tablet acting as a gateway, which then transmits the data via cellular network to the Cloud [6].

Furthermore, [10] focuses on environmental monitoring of cultural heritage, measuring temperature, humidity, and light intensity. Their setup involves measurement nodes sending data to a central Node via BLE or ZigBee, with the central node transmitting the data via WiFi or General Packet Radio Service (GPRS) to the Cloud.

3.7 Security

Security is a big concern in the field of IoT. As IoT devices proliferate as shown in Figure 3.1, so do the potential vulnerabilities and security risks. [9] delves into the security aspects of IoT devices, providing a comprehensive overview of the challenges and solutions associated with each layer of the IoT architecture:

The IoT architecture is typically based on a 3-layer system: the Perception/Hardware Layer, the Network/Communication Layer, and the Application (Interface/Service) Layer. Each of these layers has its own set of vulnerabilities and corresponding solutions.

The perception/hardware layer is the first line of defense in an IoT system. This layer is often filled with hardcoded credentials, a significant issue that poses a risk due to the widespread use of the same password across multiple devices. This makes devices prone to unauthorized access and manipulation.

The network/communication layer is responsible for transmitting the data collected by the hardware to the application layer. Authentication emerges as a key strategy for securing communication within this layer. By ensuring that only authorized devices can connect and transmit data, the integrity and confidentiality of the data can be maintained.

The application layer is where the data is processed and presented to the end-user. This layer benefits from biometrics and multi-level authentication to keep bad actors from modifying configurations. By implementing robust authentication mechanisms, unauthorized access to the system can be prevented, thereby protecting the data and the system as a whole.

In conclusion, securing IoT devices is a complex task that requires a multi-layered approach. By understanding the vulnerabilities of each layer and implementing the appropriate security measures, the integrity and security of IoT systems can be significantly enhanced.

One security measurement, for the network/communication layer, would be a Virtual Private Network (VPN). It provides secure connections between the endpoints of the devices, by establishing secure tunnels to send and receive encrypted data [11]. One of the most common VPN security protocols for IoT devices is the Internet Protocol Security (IPSec/IPv6), which encrypts data through static IP addresses ensuring privacy and End-to-End secure communication through public Internet traffic [11].

Chapter 4

Design

This Chapter presents the scenario of this project, the goal of the system, and an overview of the actors and technical components. It also shows the methodology of the testing and gives an overview of the needed experiments.

4.1 Scenario

This project is designed for scenarios where artworks need to be transported securely from private collectors to museums. Throughout the journey, various stakeholders, including the collector, the museum, and insurers, are keen to ensure the safety of the artwork.

Before the transportation begins, the transporter sends the label, a unique identifier, to all involved parties, enabling them to track the artwork. Also before departure, predetermined thresholds for temperature and humidity are set. If these thresholds are breached during transit, stakeholders receive immediate alerts, allowing for fast intervention to reduce potential damage.

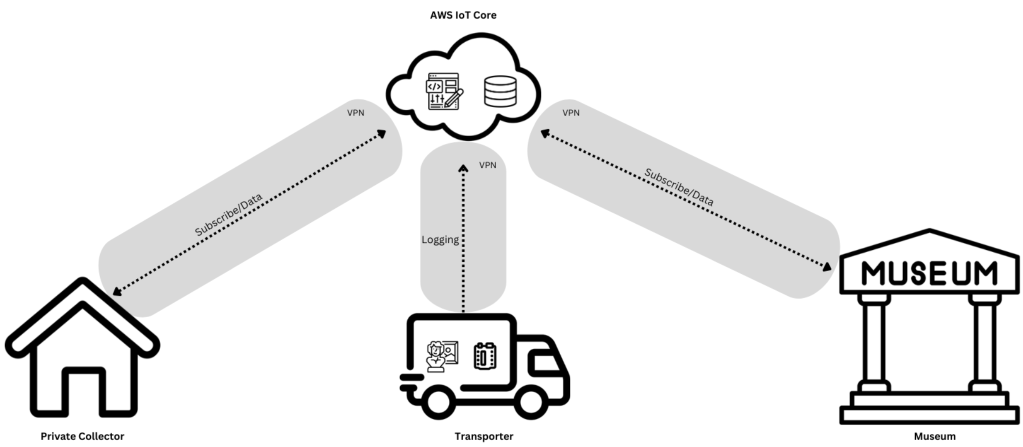

Figure 4.1: Artwork Tracking Scenario

4.2 Goal

The goal is to develop a monitoring system for the specified scenario. The system includes a monitoring device, to capture temperature, humidity, and acceleration data. Utilizing an LTE connection, the device transmits this data along with a preassigned label to the cloud. From there, the information is distributed to stakeholders, who are subscribed to the label, using the MQTT protocol. To ensure the security and confidentiality of data transfer, VPN tunnels are implemented.

4.3 Actors

This system is designed for two actors, each defined as follows:

- **Stakeholder:** The private collector, the museum, the insurer, or any other entity entrusted with the responsibility of monitoring the artwork during transportation.
- **Transporter:** A transportation company responsible for safely transporting the artwork from the private collector to the museum. They are also responsible for distributing the labels to the stakeholders and setting up the chosen thresholds.

4.4 Technical Components

The system consists of some technical components, that interact with each other and the actors to form a functional system. In Figure 4.2 the technical components and their connections are shown.

IoT Device

The monitoring device captures temperature, humidity, and acceleration data of the artwork throughout its transportation. Subsequently, it transmits the recorded data along with a corresponding label via an LTE connection to the cloud at regular intervals. Security measures are implemented through the utilization of a VPN.

Cloud

The cloud serves not only as a data storage for environmental data but also as a broker for the MQTT protocol. It receives data tagged with a specific label from the IoT device and forwards it to all subscribers of that label.

Monitoring application

To enable stakeholders to monitor the artwork effectively, a user interface is needed. This web application is specifically designed to ease the analysis of environmental data and promptly alert stakeholders when predefined thresholds are met.

VPN

VPN tunnels are implemented to secure the communication between the IoT device, the cloud, and the stakeholders. It reduces the risk of eavesdropping from third parties, by encrypting the data.

4.5 Overview

Figure 4.2 presents an overview of the system's architecture, illustrating the connections between various components and actors. It provides a visual representation of how each component interacts with others, offering insights into the system and the roles of different actors within it.

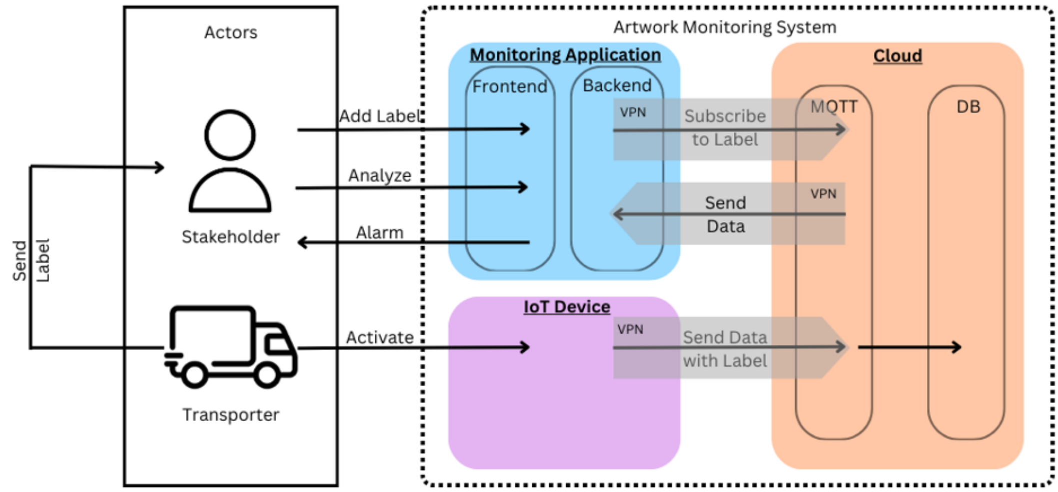

Figure 4.2: System Architecture

4.6 Testing

After implementing the system, thorough testing is essential. Therefore, three experiments are conducted. All experiments are initiated by adding the label to the monitoring application, for the stakeholder to be able to receive the data. At the end of each experiment, the transporter has to stop the data transfer by turning off the device and changing the label. This procedure ensures that stakeholders of the artwork do not have unauthorized access to data from future transports.

Experiment 1

In the first experiment, the label and thresholds are set on the IoT device, and the label is added to the monitoring application. Then the IoT device is placed in a backpack, and a walk is taken. Throughout the walk, the IoT device records the environmental data and sends them to the cloud. Meanwhile, at the starting point, a laptop assumes the role of a stakeholder, monitoring the artwork's condition by receiving real-time updates transmitted by the IoT device. As shown in Figure 4.3, the route spans 1.1 kilometers and is estimated to require approximately 15 minutes to walk. At the end of the walk, the device is turned off and the label is changed.

Figure 4.3: Route Experiment 1

Experiment 2

The second experiment involves a more dynamic setting, where the IoT device is installed within a car and gets transported, which makes it the closest to the use case. Before the transport begins, the label and thresholds are set on the IoT device, and the label is added to the monitoring application. During the ride, the IoT device continually captures temperature, humidity, and acceleration data and sends them to the cloud. Simultaneously, a laptop at the starting point works again as a stakeholder and monitors the condition of the artwork. As shown in Figure 4.4, the route spans 5.7 kilometers and is estimated to require approximately 8 to 10 minutes to drive. At the end of the transport, the device is turned off and the label is changed again.

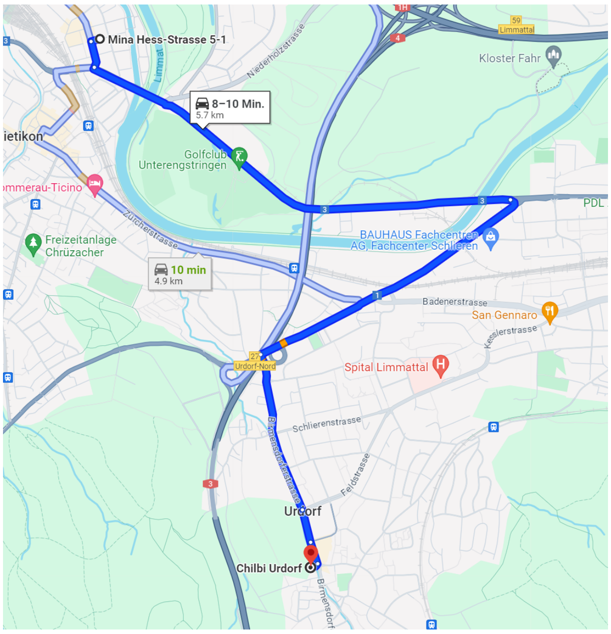

Figure 4.4: Route Experiment 2

Experiment 3

The third experiment introduces the IoT device to the potentially challenging environment of train travel, where the metallic structure of the train and other IoT devices within the train may present a risk of interference with data transmission. In this experiment, the IoT device gets transported on a train while monitoring it remotely from a laptop positioned at the starting point. As shown in Figure 4.5, the route is estimated to require approximately 36 minutes. In this experiment, the label and thresholds are also set at the beginning and the device is turned off at the end.

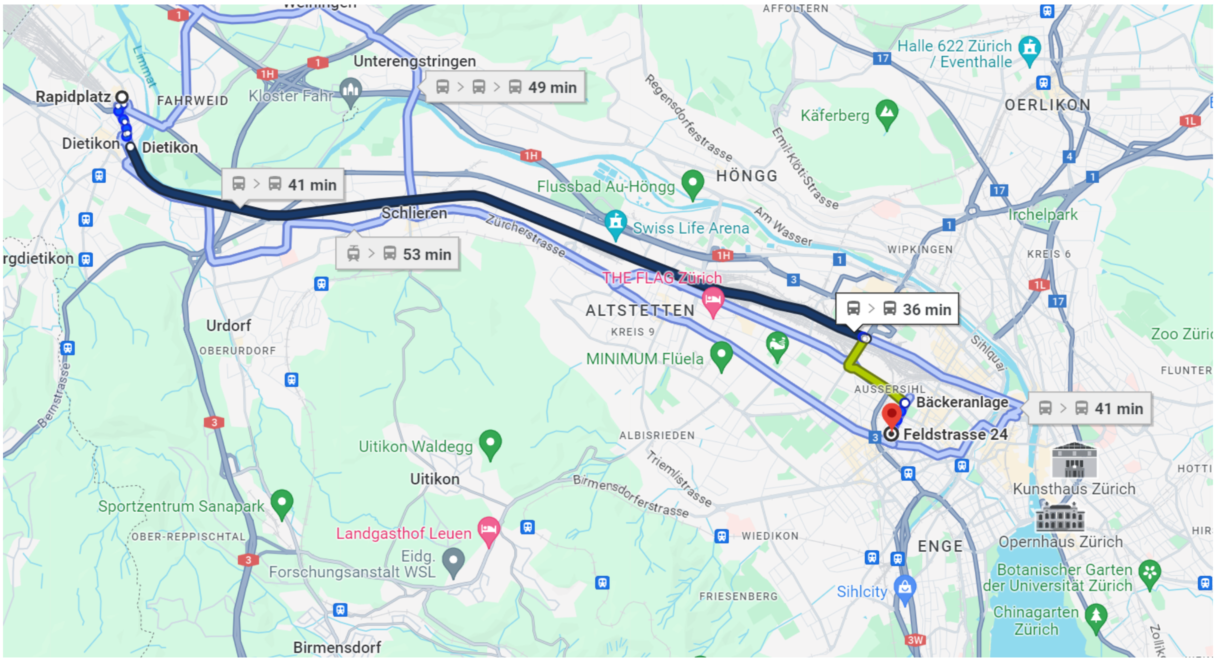

Figure 4.5: Route Experiment 3

Table 4.1 gives us an overview of the different experiments. It outlines which experiment assesses specific topics. All three experiments examine the connectivity, data measurement, and functionality in dynamic settings. Experiments 2 and 3 particularly focus on scenarios involving accelerated movement, while experiment 3 additionally investigates the resilience of the connection when the device is placed within a metal container.

	Connection	Measurement	Dynamic	Speed	Container
Experiment 1	√	√	√	X	X
Experiment 2	√	√	√	√	X
Experiment 3	√	√	√	√	√

Table 4.1: Experiments Overview Design

Chapter 5

Implementation

This chapter explores the practical implementation of the concepts discussed in earlier sections. It provides a detailed guide on configuring the system step by step, starting from sensor data acquisition to establishing a cellular connection and local monitoring. The chapter acts as a manual, demonstrating how the STMicroelectronics B-L462E-CELL1 Cellular Discovery Kit is used to measure environmental conditions like temperature, humidity, and acceleration. It also discusses the challenges faced with LTE connectivity and the subsequent shift to a local solution for data logging and analysis. The source code is available in the Bachelor-Thesis-IoT-Network Github repository[1].

5.1 Sensor Data

The first step is to get the environmental data. For that, the STMicroelectronics B-L462E-CELL1 Cellular Discovery Kit is used. As mentioned in Section 3.2, the board uses an HTS221 sensor to measure temperature and humidity and an LSM303AGR sensor to measure acceleration.

5.1.1 Setup

The implementation process is initiated by the creation of a new project on STM32Cube, with the B-L462E-CELL1 board selected from the Board Selector.

[1] https://github.com/Islemmdimagh/Bachelor-Thesis-IoT-Network

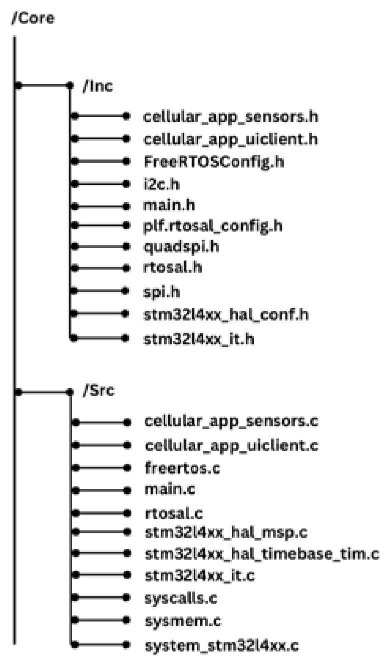

Figure 5.1: Core Directory Structure

The necessary files, specifically `cellular_app_sensors.c`, `cellular_app_uiclient.c`, and their respective header files, are sourced from the directory `en.stm32cubeexpansion-cellular-freertos-v7-1-0/STM32CubeExpansion_CELLULAR_FreeRTOS_V7.1.0/Projects/B-L462E-CELL1/Demonstrations/CellularIoT/STM32_Cellular/App`[2]. These files are subsequently added to the `Core/Src` and `Core/Inc` folders. In addition, the header files `i2c.h`, `quadspi.h`, and `spi.h` are retrieved from `en.stm32cubeexpansion-cellular-freertos-v7-1-0/STM32CubeExpansion_CELLULAR_FreeRTOS_V7.1.0/Projects/B-L462E-CELL1/Demonstrations/CellularIoT/Core/Inc` and placed in the `Core/Inc` folder. The header files `plf_rtosal_config.h` (found in `en.stm32cubeexpansion-cellular-freertos-v7-1-0/STM32CubeExpansion_CELLULAR_FreeRTOS_V7.1.0/Projects/B-L462E-CELL1/Demonstrations/CellularIoT/STM32_Cellular/Target`) and `rtosal.h` (found in `AVSystem/Anjay-freertos-client/blob/master/Middlewares/ST/STM32_Cellular/Core/Rtosal/Inc`[3]) are also incorporated into the `Core/Inc` folder. Upon completion of the modifications, the structure of the `Core` folder corresponds to the representation depicted in Figure 5.1.

[2]https://www.st.com/en/embedded-software/x-cube-cellular.html

[3]https://github.com/AVSystem/Anjay-freertos-client/blob/master/Middlewares/ST/STM32_Cellular/Core/Rtosal/Inc/rtosal.h

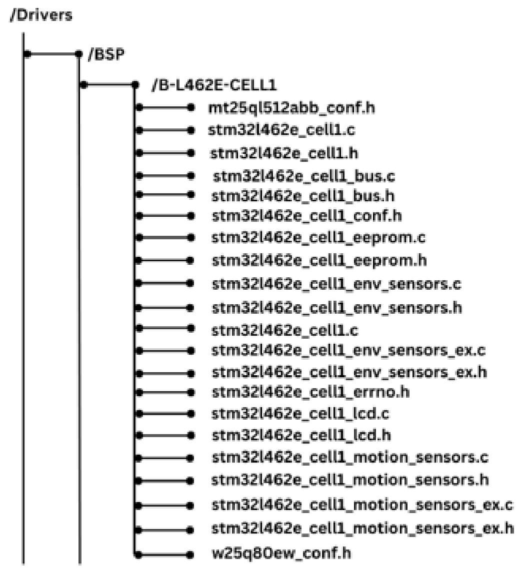

Figure 5.2: Drivers/BSP Directory Structure

Following this, the `en.stm32cubeexpansion-cellular-freertos-v7-1-0/STM32CubeExpansion_CELLULAR_FreeRTOS_V7.1.0/Drivers/BSP` folder is integrated into the project. The files `stm32l462e_cell1_qspi_in_module.c`, `stm32l462e_cell1_qspi_in_module.h`, `stm32l462e_cell1_qspi_onboard.c`, and `stm32l462e_cell1_qspi_onboard.h` are then removed from the `B-L462E-CELL1` folder. Consequently, the structure of the `Drivers/BSP` folder should resemble Figure 5.2.

The folder `en.stm32cubeexpansion-cellular-freertos-v7-1-0/STM32CubeExpansion_CELLULAR_FreeRTOS_V7.1.0/Projects/B-L462E-CELL1/Demonstrations/CellularIoT/STM32_Cellular/Target` is subsequently added to the `Core` directory, and the `en.stm32cubeexpansion-cellular-freertos-v7-1-0/Utilities` folder is included in the project.

Finally, the paths are updated to support the newly added header files.

5.1.2 Source Code

This section discusses the modifications made to the codebase to integrate temperature, humidity, and acceleration sensing functionality into the system. It also shows the implementation of data logging.

Declaration of Variables:

Three floating-point variables, `temp_value`, `hum_value`, and `acc_value`, have been introduced to store the measured temperature, humidity, and acceleration values, respectively. Additionally, character arrays `str_tmp`, `str_hum` and `str_acc` are defined to hold formatted messages displaying the values.

`uint8_t msg1[]`, `uint8_t msg2[]`, `uint8_t msg3[]`: These variables store messages that are transmitted via UART to provide status updates during the execution of the code. `msg1` signifies the beginning of measurement, `msg2` indicates the initialization of the sensors, and `msg3` confirms the successful initialization of the sensors.

`typedef struct { int temperature; int humidity; MOTION_SENSOR_Axes_t accelerometer ; } SensorData;` defines a structure named `SensorData`, encapsulating two integer fields, `temperature` and `humidity`, and a `MOTION_SENSOR_Axes_t` field, accelerometer. This structure is employed to organize and store temperature, humidity, and acceleration data coherently, potentially facilitating further processing or logging of sensor readings.

```
float temp_value = 0;  // Measured temperature value
float hum_value = 0;  // Measured humidity value
float acc_value = 0;  // Measured accelerometer value
char str_tmp[100] = ""; // Formatted message to display the temperature value
char str_hum[100] = ""; // Formatted message to display the humidity value
char str_acc[100] = ""; // Formatted message to display the accelerometer value
uint8_t msg1[] = "****** Measurement ******\n\n\r";
uint8_t msg2[] = "=====> Initialize sensors \r\n";
uint8_t msg3[] = "=====> Sensors initialized \r\n";

typedef struct {
    int temperature;
    int humidity;
    MOTION_SENSOR_Axes_t accelerometer;
} SensorData;
```

Listing 5.1: Declaration of Variables

Initialization and Configuration:

Before sensor usage, informative messages are transmitted via UART to indicate the beginning of temperature measurement. The temperature and humidity sensor, HTS221, is initialized and enabled to facilitate accurate temperature and humidity readings. Similarly, the accelerometer sensor, LSM303AGR, is initialized to begin acceleration measurement. Upon successful initialization, a confirmation message is sent via UART to acknowledge the successful setup of the sensors.

```
HAL_UART_Transmit(&huart1, msg1, sizeof(msg1), 1000);

HAL_UART_Transmit(&huart1, msg2, sizeof(msg2), 1000);

BSP_ENV_SENSOR_Init(0, ENV_TEMPERATURE);

BSP_ENV_SENSOR_Enable(0, ENV_TEMPERATURE);

BSP_ENV_SENSOR_Init(0, ENV_HUMIDITY);

```

```
BSP_ENV_SENSOR_Enable(0, ENV_HUMIDITY);

BSP_MOTION_SENSOR_Init_Acc();

HAL_UART_Transmit(&huart1, msg3, sizeof(msg3), 1000);
```

Listing 5.2: Initialization and Configuration

Data Acquisition and Formatting:

Within the main loop, temperature and humidity values are retrieved from the sensors using `BSP_ENV_SENSOR_GetValue()` function calls. Acceleration data is retrieved using the `BSP_MOTION_SENSOR_GetAxes()` function. The obtained values are then formatted into human-readable strings (`str_tmp`, `str_hum`, `str_acc`) and transmitted via UART using `snprintf()`. To ensure compatibility with UART transmission, integer representations of the temperature and humidity values are derived from the floating-point measurements.

```
BSP_ENV_SENSOR_GetValue(0, ENV_TEMPERATURE, &temp_value);
int tempInt = (int)temp_value;
int tmpInt1 = temp_value;
float tmpFrac = temp_value - tmpInt1;
int tmpInt2 = trunc(tmpFrac * 100);
snprintf(str_tmp,100," TEMPERATURE = %d.%02d degree C\n\r", tmpInt1, tmpInt2);

BSP_ENV_SENSOR_GetValue(0, ENV_HUMIDITY, &hum_value);
int humInt = (int)hum_value;
int humInt1 = hum_value;
float humFrac = hum_value - humInt1;
int humInt2 = trunc(humFrac * 100);
snprintf(str_hum,100," HUMIDITY = %d.%02d %%\n\r", humInt1, humInt2);

MOTION_SENSOR_Axes_t acceleration;
BSP_MOTION_SENSOR_GetAxes(STM32L462E_CELL1_LSM303AGR_ACC_0, MOTION_ACCELERO, &acceleration);
snprintf(str_acc,100," ACCELERATION = %d %d %d\n\r", acceleration.x, acceleration.y, acceleration.z);
```

Listing 5.3: Data Acquisition and Formatting

Transmission and Data Logging:

The formatted temperature, humidity, and acceleration strings are transmitted via UART to facilitate real-time monitoring or logging. Additionally, a custom data structure, `SensorData`, is utilized to encapsulate the integer representations of temperature, humidity, and acceleration for potential data logging or processing purposes.

```
BHAL_UART_Transmit(&huart1,( uint8_t *)str_tmp,sizeof(str_tmp),1000);
HAL_UART_Transmit(&huart1,( uint8_t *)str_hum,sizeof(str_hum),1000);
```

```
HAL_UART_Transmit(&huart1,( uint8_t *)str_acc,sizeof(str_acc)
    ,1000);

SensorData data;
data.temperature = tempInt;
data.humidity = humInt;
data.accelerometer = acceleration;
```

Listing 5.4: Transmission and Data Logging

Delay and Iteration:

A delay of 1 second (`HAL_Delay(1000)`) is incorporated between consecutive sensor readings to regulate the sampling rate and prevent data flooding. The main loop continues indefinitely, ensuring continuous monitoring of temperature, humidity, and acceleration values.

```
while (1)
{

    //Data Acquisition and Formatting

    //Transmission and Data Logging

    HAL_Delay(1000);
}
```

Listing 5.5: Delay and Iteration

5.2 Cellular Connection

The next step is to configure the cellular connection. As shown in Section 3.2 the B-L462E-CELL1 has a built-in eSIM. The configuration process is outlined as follows:

Baud rate:	115200
Data:	8bits
Parity:	none
Stop:	1bit
Flow control:	none

Figure 5.3: Tera Term Configuration

1. **Connection of the Board and Retrieval of ICCID:** To begin, the board should be connected using the USB ST-LINK port, as depicted in Figure 3.2, and Tera Term should be opened. A reset of the board should be initiated, followed by the configuration of the necessary parameters in Tera Term, such as baud rate, Data,

Parity, Stop, and Flow control, as demonstrated in Figure 5.3. Subsequently, the ICCID associated with the embedded eSIM should be retrieved.

2. **Creation of Truphone Account:** The Truphone website[4] is accessed, and a personal account is created.

3. **Activation of eSIM:** Upon the creation of the account, the eSIM is activated by inputting the previously obtained ICCID. The free plan is selected, and the activation process is finalized.

4. **Management of eSIM via Truphone Dashboard:** The Truphone dashboard[5] is utilized to monitor and manage the eSIM functionality. This includes tracking the number of SIM cards associated with the account and monitoring data usage.

5. **Verification of Network Connection:** After the activation of the eSIM, network connectivity is verified by resetting the board and confirming successful connection establishment via Tera Term.

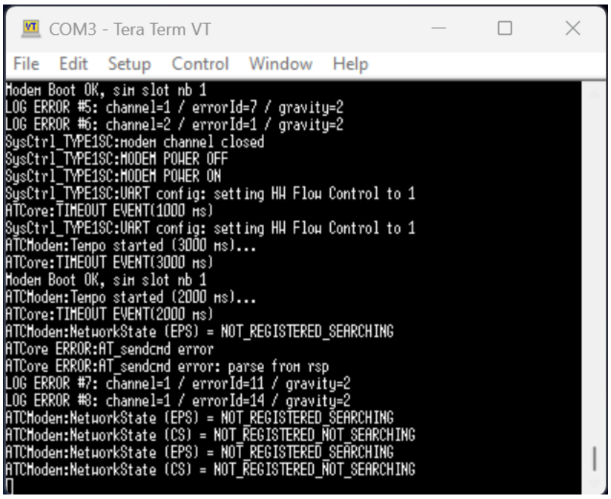

Figure 5.4: Tera Term Log NetworkState

Unfortunately, step 5 could not be accomplished. Despite all preceding steps being completed, the board failed to establish a network connection, as illustrated in Figure 5.4. The issue was reported to STMicroelectronics, who identified a problem from their end. Consequently, the eSIM functionality became unusable.

The alternative solution involved utilizing the SIM card slot of the B-L462E-CELL1 and employing an external SIM card. The "digitec iot Flat SIM" was selected, with the "digitec Flat 0.4" plan offering a 30-day usage period.

The initial step involved registering the SIM card on the digitec website[6]. Subsequently, the X-Cube-Cellular software version 5.2.0[7] was downloaded. The binary file (`en.x-cube-`

[4] iot.truphone.com/website
[5] iot.truphone.com/dashboard
[6] iot.digitec.ch
[7] https://www.st.com/en/embedded-software/x-cube-cellular.html

`cellular/STM32CubeExpansion_CELLULAR_V5.2.0/Projects/B_L462E/Demonstrations/Cellular/Binaries/l462_t1se_socket_v520.bin`) was then uploaded onto the board. Upon resetting the board, the X-CUBE-CELLULAR main menu became visible.

```
============================
    X-CUBE-CELLULAR
    Version: V5.2.0
============================
Select the application to run:

1: Setup configuration Menu
2: FAQ display
3: Modem power on (without application)
4: Display Voucher code

Or type any key to start
1

--------------------------------
Date: Mon 01/01/2000 - 00:10:53
--------------------------------
   Setup configuration Menu
--------------------------------
Select the action to process:

0: Quit
1: Date/Time setting (RTC)
2: Configuration: Cellular Service
3: Configuration: Ping
8: Status of above configurations
9: Erase all FEEPROM configurations (restore to DEFAULT)

2

------------------------------------
 Cellular Service configuration Menu
------------------------------------
 c : update configuration by console and store it in FEEPROM
 e : erase the configuration stored in FEEPROM (restore to DEFAULT)
 l : list current configuration
 h : help
 q : quit

--------------------------------
 Cellular Service from UART
--------------------------------

Version (7):
Enter Sim Slot List (0: socket / 1: embedded sim) (possible values (0 1 or 01) (0):
Sim slot 0 (MODEM SOCKET) config:
Enter APN [enter "" to delete the current value]  ():

Enter CID (1-9) (1):
1
Enter username [enter "" to delete the current value]  ():

Enter password [enter "" to delete the current value]  ():

Enter cellular target state (0: modem off / 1: sim only / 2: full cellular data) (2):
2
Enter attachment timeout (ms)  (180000):
180000
Enter Network register mode (0: AUTO / 1: MANUAL / 2: DEREGISTER / 3: MANUAL THEN AUTO) (0):
0
Enter Operator name format (0: LONG / 1: SHORT / 2: NUMERIC / 9: NOT PRESENT) (9):
9
Enter Operator name (00101):
00101
Enter Access techno present (0: NO / 1: YES) (0):
0
Enter Access techno (0: GSM / 7: CatM1 / 9: NB1) (7):
9
Enter Low power inactivity timeout (ms)  (1000):
1000
Enter NFMC activation (0: inactive / 1: active)  (0):
0
New config is written in feeprom (104 bytes)
```

Figure 5.5: External SIM Configuration

Subsequently, the board was configured to utilize the external SIM card, as depicted in Figure 5.5. However, utilizing an external SIM also proved unsuccessful. Following further communication with STMicroelectronics, it was concluded that establishing an LTE connection using the B-L462E-CELL1 is presently not feasible in Switzerland.

5.3 Local Monitoring

Due to the issues encountered with the LTE connection, a local solution is necessary. The local approach involves measuring environmental data, logging it, and subsequently analyzing it locally. The implementation of the measurement process has been detailed in Chapter 5.1. For local logging, Tera Term serves as the chosen tool.

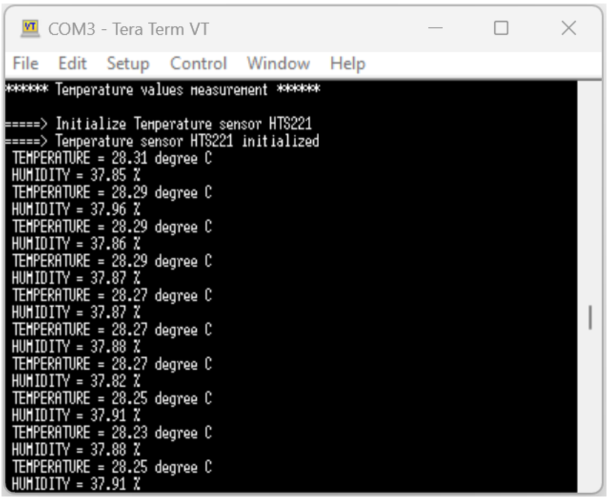

Figure 5.6: Environmental Data Tera Term

Upon establishing a new connection, environmental data is logged on Tera Term by selecting ”Serial” and choosing the Port ”COM3: STMicroelectronics STLink Virtual COM Port (COM3),” as depicted in Figure 5.6. To save the log file locally, navigate to ”File”, then ”Log...”, choose the desired saving location, choose the ”Timestamp” option, and proceed to save the file.

The subsequent task involves the creation of a tool to monitor and analyze the logged data. A Python script has been developed for this purpose, capable of parsing the logging data and generating temperature, humidity, and acceleration plots, which update regularly.

The script is structured as follows:

1. Importing Libraries:

```
import matplotlib.pyplot as plt
import datetime
import time

```

Listing 5.6: Import Libraries

This part imports the necessary libraries: `matplotlib.pyplot` for plotting graphs, `datetime` for handling date and time data, and `time` for adding time delays.

2. Main Loop:

```
while(True):

```

Listing 5.7: Main Loop

This creates an infinite loop. The code within this loop will continue to execute indefinitely until manually interrupted or a break condition is met.

3. Reading Tera Term Log File:

```
with open("teraterm.log", "r") as log_data:
lines = log_data.readlines()

```

Listing 5.8: Reading Tera Term Log File

This section opens the Tera Term log file named 'teraterm.log' in read mode and reads all lines into a list called `lines`.

4. Parsing Log Data:

```
timestamps, temperatures, humidities = [], [], []
accelerations_x, accelerations_y, accelerations_z = [], [], []
last_temperature, last_humidity, last_x, last_y, last_z = None, None, None, None, None
for line in lines:
    if '] ' not in line:  # Skip lines without the expected format
        continue
    timestamp_str, data_str = line.split('] ')

```

Listing 5.9: Parsing Log Data

This loop iterates through each line in the log file, parsing the timestamp and data values. It skips lines that don't match the expected format. It splits each line into a timestamp string and a data string based on the ']' delimiter.

5. Extracting Data:

```
timestamp = datetime.datetime.strptime(timestamp_str[1:], '%Y-%m-%d %H:%M:%S.%f')

```

Listing 5.10: Extracting Data

This converts the timestamp string into a `datetime` object, stripping the leading '[' character and formatting it as 'YYYY-MM-DD HH:MM:SS.ssssss'.

6. Handling Data Type:

```
if data_type == 'TEMPERATURE':
    last_temperature = value
elif data_type == 'HUMIDITY':
        last_humidity = value
elif data_type == 'ACCELERATION':
    # Extract acceleration components
    data_type, equal, x_value, y_value, z_value = data_parts[:5]
    last_x = x_value
    last_y = y_value
    last_z = z_value

```

Listing 5.11: Handling Data Type

This section identifies whether the data corresponds to temperature, humidity, or acceleration and stores the value accordingly.

7. Plotting Data:

```
    if last_temperature is not None and last_humidity is not None and last_x is not None and last_y is not None and last_z is not None:
            timestamps.append(timestamp)
            temperatures.append(last_temperature)
            humidities.append(last_humidity)
            accelerations_x.append(last_x)
            accelerations_y.append(last_y)
            accelerations_z.append(last_z)
            last_temperature, last_humidity, last_x, last_y, last_z = None, None, None, None, None

```

Listing 5.12: Plotting Data

Once the data is available, it is appended to the respective list.

8. Plotting:

```
    plt.figure(figsize=(12, 8))

    # Plot for temperature
    plt.subplot(5, 1, 1)
    plt.plot(timestamps, temperatures, label='Temperature (C)', color='red')
    ...
    # Plot for humidity
    plt.subplot(5, 1, 2)
    plt.plot(timestamps, humidities, label='Humidity (%)', color='blue')
    ...
    # Plot for acceleration x
    plt.subplot(5, 1, 3)
    plt.plot(timestamps, accelerations_x, label='Acceleration X', color='green')
    ...
    # Plot for acceleration y
    plt.subplot(5, 1, 4)
    plt.plot(timestamps, accelerations_y, label='Acceleration Y', color='orange')
    ...
    # Plot for acceleration z
    plt.subplot(5, 1, 5)
    plt.plot(timestamps, accelerations_z, label='Acceleration Z', color='purple')

```

Listing 5.13: Plotting

Listing 5.13 initializes a new figure for the plots with a specific size. Creates five subplots within the figure, one for temperature, one for humidity, and one for each acceleration axis, and plots the corresponding data.

9. Customizing Plots:

```
    plt.xlabel('Time')
    plt.ylabel('Temperature (C)')
    plt.title('Temperature over Time')
    plt.legend()
    plt.grid(True)
    ...
    plt.xlabel('Time')
    plt.ylabel('Humidity (%)')
    plt.title('Humidity over Time')
    plt.legend()
    plt.grid(True)
    ...
    plt.xlabel('Time')
    plt.ylabel('Acceleration')
    plt.title('Acceleration over Time X-Axis')
    plt.yticks([-100, -50, 0, 50, 100])
    plt.legend()
    plt.grid(True)
    ...
    plt.xlabel('Time')
    plt.ylabel('Acceleration')
    plt.title('Acceleration over Time Y-Axis')
    plt.yticks([-100, -50, 0, 50, 100])
    plt.legend()
    plt.grid(True)
    ...
    plt.xlabel('Time')
    plt.ylabel('Acceleration')
    plt.title('Acceleration over Time Z-Axis')
    plt.yticks([-100, -50, 0, 50, 100])
    plt.legend()
    plt.grid(True)

```

Listing 5.14: Customizing Plots

This section adds labels, titles, legends, and gridlines to the plots for better interpretation.

10. Displaying Plots:

```
    plt.tight_layout()  # Adjust layout to prevent
   overlapping labels
    plt.show()

```

Listing 5.15: Displaying Plots

The plots are displayed. `plt.tight_layout()` adjusts the layout to prevent overlapping labels.

11. Delay:

```
    time.sleep(5)

```

Listing 5.16: Delay

After displaying the plots, the code pauses execution for 5 seconds before looping back to the beginning.

Chapter 6

Evaluation

This chapter goes through the experiments conducted to simulate various artwork transportation scenarios, building upon the experiments in Section 4.6. Due to connectivity challenges outlined in Chapter 5.2, modifications were made to facilitate these experiments using local implementations.

6.1 Experiment 0

Experiment 0 is the basis for these experiments, it shows the measured data of the board staying 28 hours on a table. While temperature and humidity might change, the acceleration has to be almost non-existent for the board to be accurate.

The experiment was conducted following these steps:

1. Connect the board (USB ST-LINK) to the laptop using a USB micro cable.
2. Launch Tera Term, select 'Serial' and choose 'COM3: STMicroelectronics STLink Virtual COM Port (COM3)' as the 'Port'.
3. Adjust the baud rate to 115200 under 'Setup' > 'Serial Port...' in Tera Term.
4. Log data with timestamps using 'File' > 'Log...' in Tera Term.
5. Execute the 'Temperature' project on the board.
6. Run TeraTermPlot.py, specifying the log file name and location chosen in step 4.
7. Place the setup on a table and leave it like that for 28 hours.
8. Stop data plotting after 28 hours and proceed to analyze the collected data.

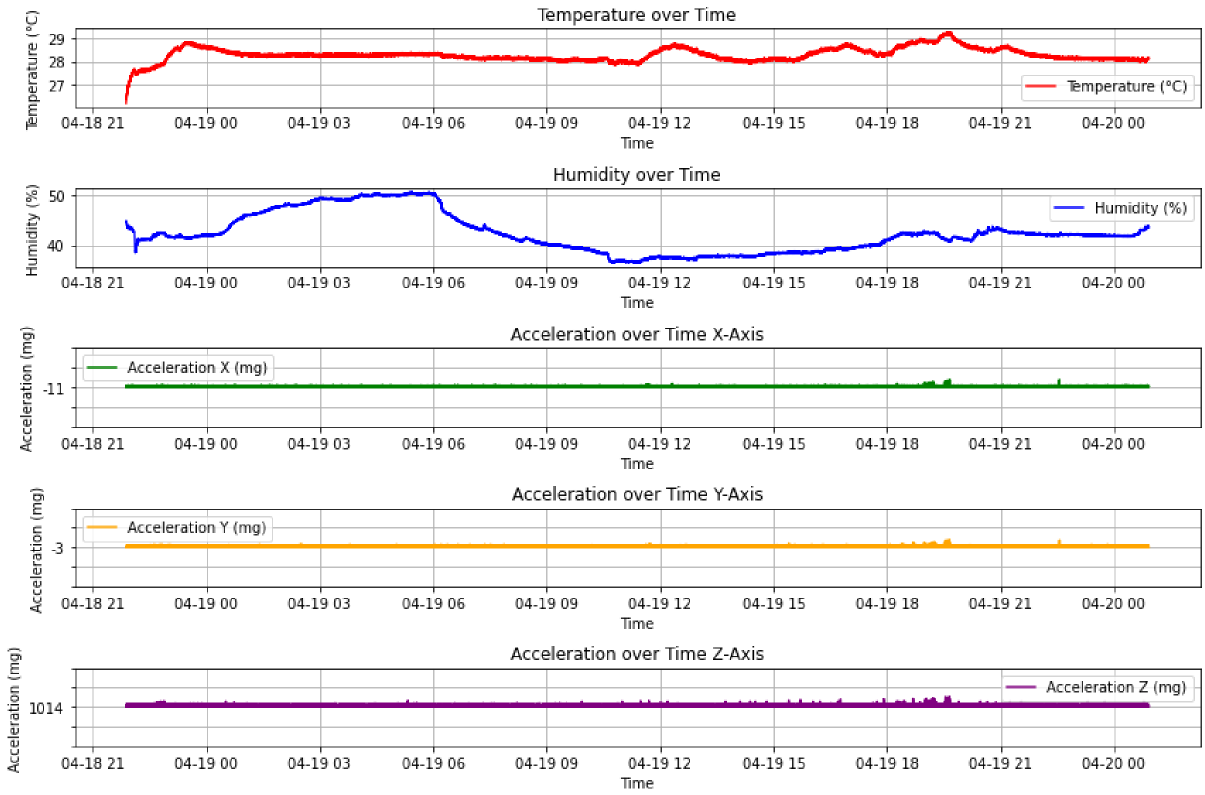

Figure 6.1: Environmental Data Experiment 0

As shown in Figure 6.1, both temperature and humidity changed during the 28 hours. One reason for the humidity change between 04-19 00 and 04-19 06 was the presence of a sleeping person in the room during that period. Throughout the 28 hours, the acceleration across all three axes remained constant. The X- and Y-axes recorded values of -11 mg and -3 mg, respectively, indicating minimal acceleration close to zero on these axes. The Z-axis, however, showed an acceleration of 1014 mg, with some slight movement caused by the usage of the table, over the 28 hours, nearly aligning with the expected value of around 1000 mg when the sensor is oriented parallel to the Earth's surface. This alignment corresponds to Earth's gravity, which is approximately 1000 mg. This experiment also demonstrates that the system is capable of functioning effectively during extended transportation durations, which is frequently encountered in real-world scenarios.

6.2 Experiment 1

Experiment 1 tested local communication between the ST-Board and a laptop, measuring the system's temperature, humidity, and acceleration. It also tested the system's reliability in a dynamic scenario, such as going on a walk.

Figure 6.2: Backpack scenario

The experiment was conducted following these steps:

1. Connect the board (USB ST-LINK) to the laptop using a USB micro cable.
2. Launch Tera Term, select 'Serial' and choose 'COM3: STMicroelectronics STLink Virtual COM Port (COM3)' as the 'Port'.
3. Adjust the baud rate to 115200 under 'Setup' > 'Serial Port...' in Tera Term.
4. Log data with timestamps using 'File' > 'Log...' in Tera Term.
5. Execute the 'Temperature' project on the board.
6. Run TeraTermPlot.py, specifying the log file name and location chosen in step 4.
7. Place the setup in a backpack as shown in Figure 6.2 and walk the designated route (see Figure 4.3).
8. Stop data plotting at the end of the walk and proceed to analyze the collected data.

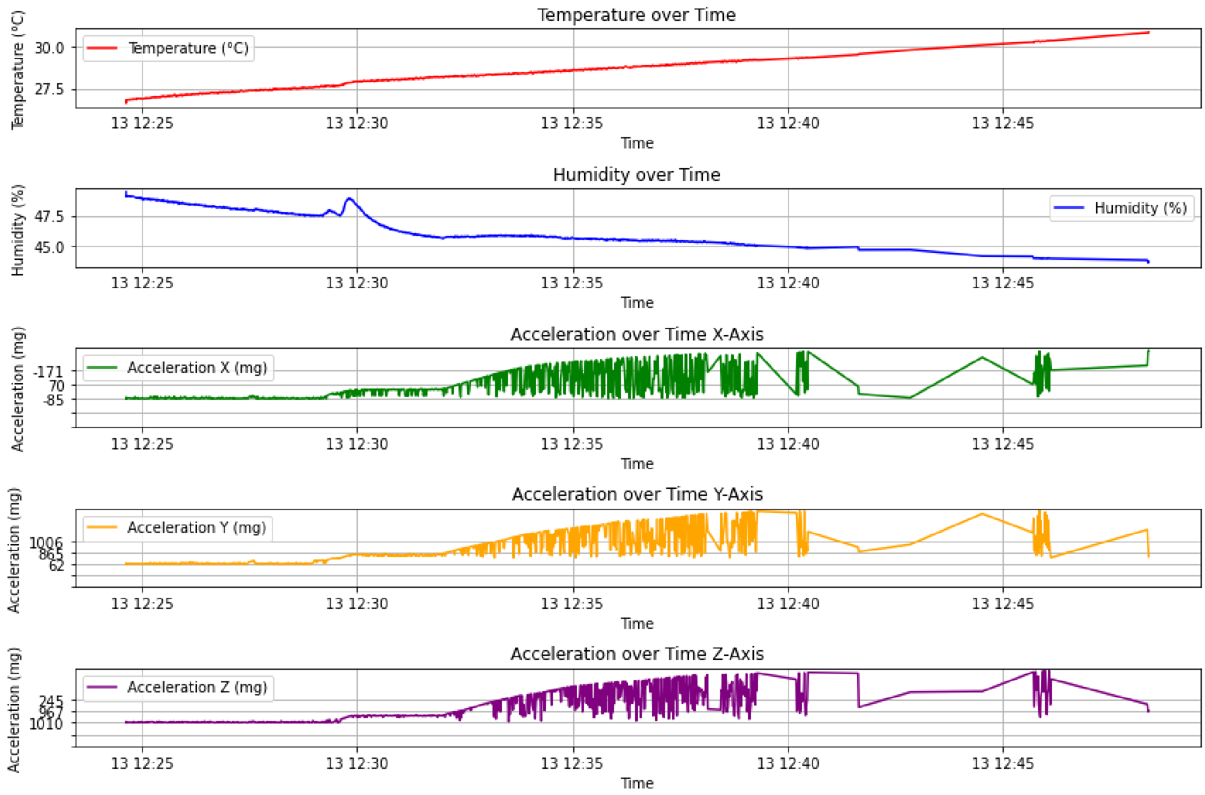

Figure 6.3: Environmental Data Experiment 1

As shown in Figure 6.3 the temperature rose during the walk. The reason for that was the direct sunlight on the backpack, which also led to the humidity drop throughout the walk.

6.3 Experiment 2

Experiment 2 expanded on local communication and data measurement by testing the system in a faster scenario. To test that, the system was put into a car, giving insights into its performance under faster conditions.

The experiment procedure included:

1. Connecting the board (USB ST-LINK) to the laptop using a USB micro cable.
2. Initiating Tera Term, selecting 'Serial' and choosing 'COM3: STMicroelectronics STLink Virtual COM Port (COM3)' as the 'Port'.
3. Setting the baud rate to 115200 under 'Setup' > 'Serial Port...' in Tera Term.
4. Logging data with timestamps using 'File' > 'Log...' in Tera Term.
5. Running the 'Temperature' project on the board.

6. Executing TeraTermPlot.py with the specified log file name and location from step 4.
7. Placing the setup within the car and driving the designated route (see Figure 4.4).
8. Stopping data plotting at the end of the drive and proceeding to analyze the collected data.

6.3.1 Experiment 2a

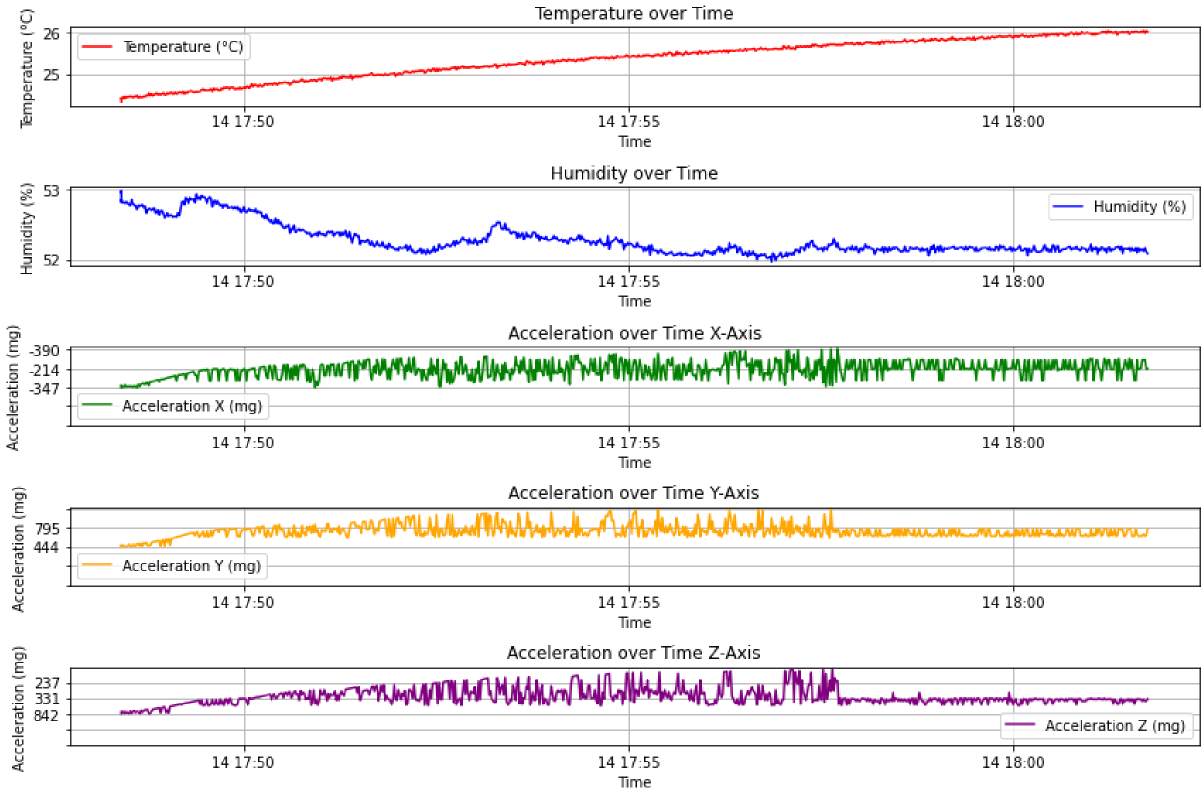

Figure 6.4: Environmental Data Experiment 2a

In Figure 6.4 the temperature showed a slight increase during the drive, rising from 24.5 degrees to 26 degrees. The humidity remained relatively stable, fluctuating between 53% and 52%. The acceleration data showed consistent patterns throughout the ride, without any longer pauses or significant fluctuations.

6.3.2 Experiment 2b

A second experiment was conducted under similar conditions. The only difference was an open window for 6 minutes during the ride. This allowed for the investigation of how external factors can influence the data.

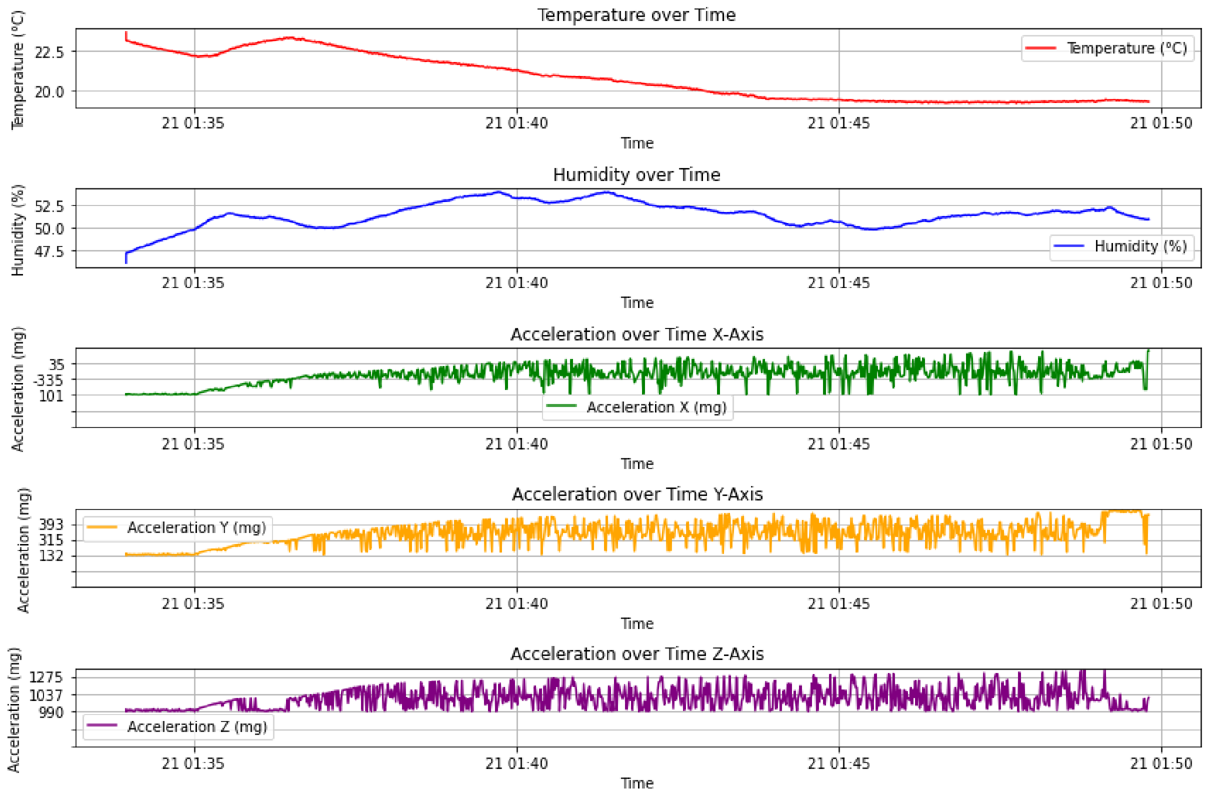

Figure 6.5: Environmental Data Experiment 2b

Figure 6.5 shows a temperature drop between 21 01:37 and 21 01:43 from 23.4 degrees to 19.8 degrees, due to the open window during that period. The humidity had more fluctuation than in Experiment 2a going from 47.5% up to 53%. The acceleration looks similar to the first drive with a little bit more fluctuation and a longer stop at around 21 01:36 due to a construction site.

6.4 Experiment 3

Experiment 3 combined elements from experiments 1 and 2 and introduced additional challenges, such as using public transportation like buses and trains. It also tests the reliability of the system in a metal container.

The experiment followed these steps:

1. Connecting the board (USB ST-LINK) to the laptop using a USB micro cable.
2. Opening Tera Term, selecting 'Serial' and choosing 'COM3: STMicroelectronics STLink Virtual COM Port (COM3)' as the 'Port'.
3. Setting the baud rate to 115200 under 'Setup' > 'Serial Port...' in Tera Term.
4. Logging data with timestamps using 'File' > 'Log...' in Tera Term.

5. Running the 'Temperature' project on the board.
6. Executing TeraTermPlot.py with the specified log file name and location from step 4.
7. Placing the setup in a backpack and following these steps:
   (a) Walking to the train station.
   (b) Taking the train.
   (c) Changing to a bus.
   (d) Leaving the bus and completing the designated route (see Figure 4.5) by walking.
8. Stopping data plotting at the end of the route and proceeding to analyze the collected data.

6.4.1 Experiment 3 Fail

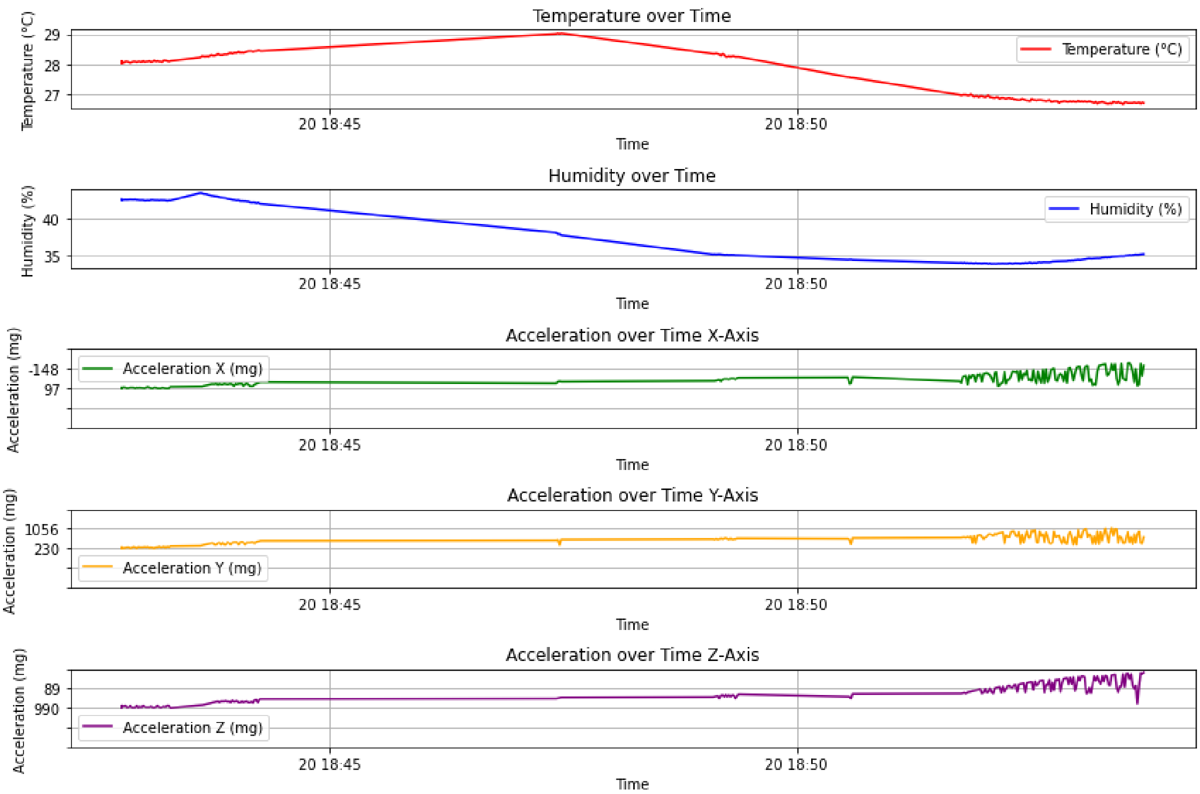

Figure 6.6: Failed Environmental Data Experiment 3

The initial attempt failed after 10 minutes. The board disconnected during the transport, due to movement of the cable inside the bag.

6.4.2 Experiment 3 Second Attempt

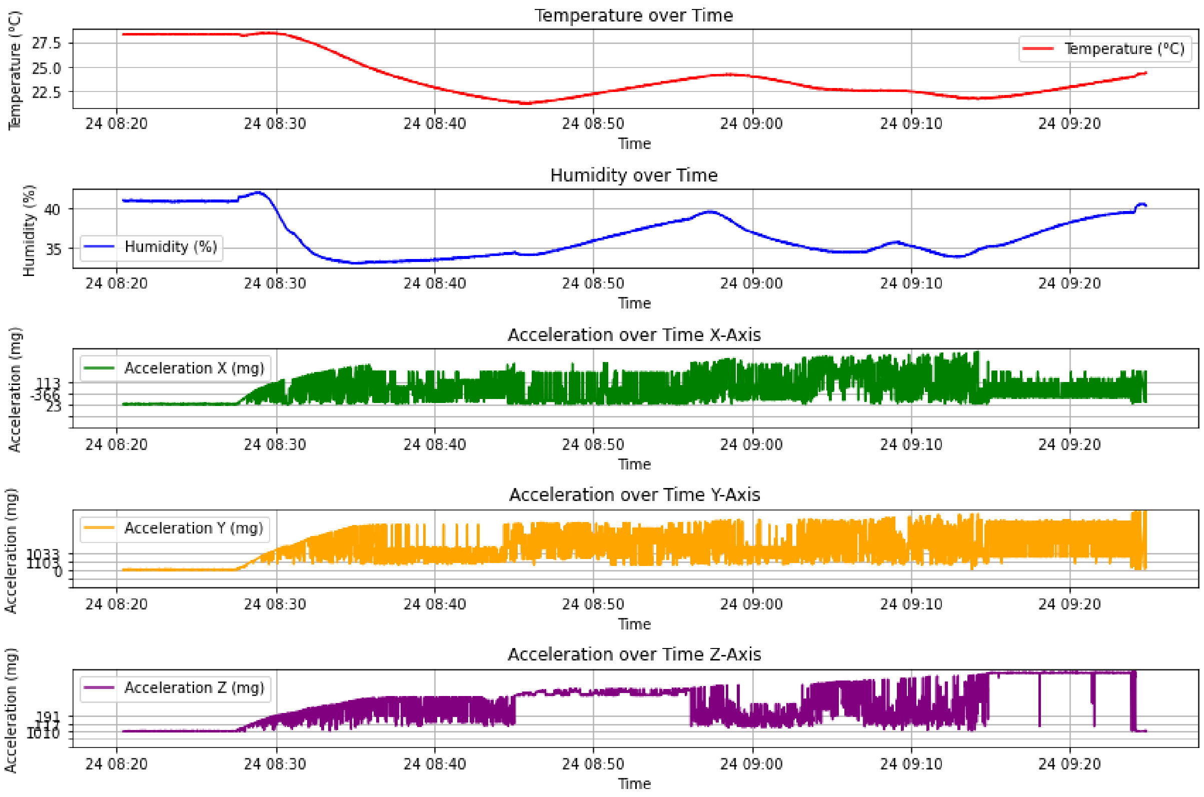

Figure 6.7: Environmental Data Experiment 3

On the second attempt data collection worked. Various transportation methods can be seen using the data. Initially, from 24 08:28 to 24 08:45, the board was transported on foot, exhibiting acceleration data similar to Experiment 1. At the same time, there was a decrease in temperature and humidity. Subsequently, from 24 08:45 to 24 08:56, the board was transported by train, showcasing reduced z-axis fluctuations, and a rise in temperature and humidity during the ride. Following this, from 24 08:57 to 24 09:03, another walking phase was conducted before switching to bus transport until 24 09:07. The final leg of the journey was again on foot.

6.5 Discussion

Even Though it was not possible to implement the planned system as a whole, due to connectivity challenges, adaptations were made to test data measurement and local logging in different scenarios. The lack of documentation for the board made the implementation of data sensing more challenging. However, despite this hurdle, the system exhibited consistent transmission during all experiments, showcasing its reliability in local communication. The only point of failure identified was the cable disconnecting, which can be addressed by using a bigger backpack. The system proved capable of capturing environmental data accurately, even in faster-paced situations like car and train transportation. This shows that the prototype can reliably function in the real-world art transport environment. It also proved to be useful for a longer transport as shown in experiment 0, where the system worked for 28 hours without any problem.

Table 6.1 gives an overview of the different experiments that were conducted. It shows that all 4 Experiments tested the connection between the board and laptop, and they all tested if the data was measured correctly. Experiments 1, 2, and 3 also tested if the system works in dynamic situations, whereas Experiments 2 and 3 also tested if there is a difference if the system moves faster. Moreover, Experiment 3 explored the impact of being in a metal container, like a train, on the accuracy of the data. Experiment 0 also tested if the system works for a longer period.

	Connection	Measurement	Dynamic	Speed	Container	Longer Duration
Experiment 0	√	√	X	X	X	√
Experiment 1	√	√	√	X	X	X
Experiment 2	√	√	√	√	X	X
Experiment 3	√	√	√	√	√	X

Table 6.1: Experiments Overview Evaluation

Chapter 7

Summary and Conclusions

The purpose of this chapter is to wrap up the thesis by providing a summary, drawing conclusions, and discussing potential future work. These discussions are based on the insights gained during the development process and the identified limitations of the system.

7.1 Summary

This thesis introduces a system designed for monitoring artwork during transportation. The concept involves the integration of an IoT device onto the artwork, tasked with capturing and storing critical environmental data like temperature, humidity, and acceleration to ensure the safety of the artwork. The board assesses this recorded data against predefined thresholds and notifies stakeholders if these thresholds are reached. The measured data is stored in the cloud and distributed to various stakeholders via the MQTT protocol. To demonstrate the feasibility of the proposed system, a prototype was developed as part of this thesis.

The first step involved gaining the theoretical knowledge for this project through a literature review. The thesis provided the theoretical background on IoT, microcontrollers, wireless communication technologies, cloud computing, MQTT, and their application in artwork tracking, along with addressing IoT security concerns.

The system design includes a description of an artwork monitoring scenario, an overview of the involved actors, a list of the technical components, an overview of the system architecture, and a testing methodology to evaluate the system. The system contains a monitoring application for the stakeholders to monitor the artwork during transport, and the IoT device which measures the environmental data. This data is then transmitted to the cloud, which not only distributes it to stakeholders but also stores it in a database for future reference and analysis.

The implementation of the prototype followed the outlined system design, with the B-L462E-CELL1 board serving as the IoT device. The first step was to enable the HTS221 sensor to measure temperature and humidity and the LSM303AGR sensor to measure acceleration. After that, logging of the measured data was implemented. The next step was to establish a cellular network connection, which unfortunately was not possible. Due to the network issues, the designed system was no longer viable, necessitating the implementation of a local solution. Instead of using a monitoring application, the artwork has to be monitored locally. For this purpose, a tool to monitor and analyze the data has been implemented in Python.

The system was evaluated in four simulated artwork monitoring scenarios. The first scenario tested the longevity of the system, by making it run for 28 hours. The second scenario focused on testing communication between the board and laptop, along with measurements in a dynamic environment. In the third scenario, speed was introduced during a car ride, representing a more realistic scenario. The fourth scenario was a hybrid test, involving transportation by walking, bus, and train to assess transitions between different transport modes.

7.2 Conclusion

A comprehensive exploration of the standards and practices within the artwork industry has been conducted, giving insight into the diverse applications of IoT devices for artwork monitoring across various settings and environments. The prototype for monitoring artwork during transportation was successfully designed. However, due to network connection issues, the planned remote implementation was not feasible, leading to a local implementation instead. The lack of documentation for the board made the implementation challenging. The local setup involves plotting the measured data to a laptop connected to the board, allowing for continuous local monitoring of the artwork. The data can be saved locally for future analysis. The effectiveness of the local prototype was validated through three simulations of artwork transportation, demonstrating its reliability in real-world scenarios.

7.3 Future Work

The goal of a remote monitoring system using the cloud was not reached, due to network connection issues, therefore these issues should be resolved and the initial design should be implemented in the future. Once that is implemented there are some improvements which can be done. To minimize the cost of data transmission, save the data locally and transmit it only when necessary, such as when a threshold is reached or the memory of the board is approaching its capacity. Another improvement could be to connect an air conditioner or a heater to the board, to regulate the temperature automatically once the predefined thresholds are met.

Due to the mentioned issues a local prototype to monitor artwork during its transportation was implemented. One improvement would be to automatically save the plots during the monitoring, instead of having to save them manually at the end of the transportation. Another one would be to add an alarm, once a threshold is reached. The alarm could be a print to the console or an audible alert to warn the user to act accordingly. Lastly, a script that automates all the necessary steps for the system to work should be implemented, simplifying user interaction to just running the script.

Bibliography

[1] Ghada Alsuhly and Ahmed Khattab. “An IoT monitoring and control platform for museum content conservation”. In: *2018 International Conference on Computer and Applications (ICCA)*. IEEE. 2018, pp. 196–201.

[2] Amazon Web Services. *AWS IoT Core*. Accessed: March 04, 2024. URL: https://aws.amazon.com/de/iot-core/.

[3] Prasanth Ch BV, G Sreekar, Gvs Jatin, and Jalpa Shah. “Iot based environment monitoring system to protect heritage artefacts”. In: *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*. IEEE. 2021, pp. 145–150.

[4] F John Dian and Reza Vahidnia. “LTE IoT technology enhancements and case studies”. In: *IEEE Consumer Electronics Magazine* 9.6 (2020), pp. 49–56.

[5] EMQ Technologies Inc. *Understanding AWS IoT Core: Features, Use Cases, Quick Tutorial*. Accessed: March 04, 2024. 2023. URL: https://www.iotforall.com/understanding-aws-iot-core-features-use-cases-quick-tutorial.

[6] Ada Fort, Elia Landi, Marco Mugnaini, Lorenzo Parri, and Valerio Vignoli. “Performance Evaluation of an IoT Sensor Node for Health Monitoring of Artwork and Ancient Wooden Structures”. In: *Sensors* 22.24 (2022), p. 9794.

[7] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Khalid Al-Ali, Xiaojiang Du, Ihsan Ali, and Mohsen Guizani. “A survey of machine and deep learning methods for internet of things (IoT) security”. In: *IEEE communications surveys & tutorials* 22.3 (2020), pp. 1646–1685.

[8] Google Cloud. *What is Cloud Computing?* Accessed: March 04, 2024. URL: https://cloud.google.com/learn/what-is-cloud-computing?hl=en.

[9] Wan Haslina Hassan et al. “Current research on Internet of Things (IoT) security: A survey”. In: *Computer networks* 148 (2019), pp. 283–294.

[10] Adrián Hinostroza and Jimmy Tarrillo. “Wireless Sensor Network for Environmental Monitoring of Cultural Heritage.” In: *SENSORNETS*. 2022, pp. 171–175.

[11] Mazen Juma, Azza Abdel Monem, and Khaled Shaalan. “Hybrid End-to-End VPN Security Approach for Smart IoT Objects”. In: *Journal of Network and Computer Applications* 158 (2020), p. 102598. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2020.102598.

[12] A Bin Karim, AZ Hassan, MM Akanda, and Avijit Mallik. “Monitoring food storage humidity and temperature data using IoT”. In: *MOJ Food Process Technol* 6.4 (2018), pp. 400–404.

[13] Surapon Kraijak and Panwit Tuwanut. “A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and

future trends". In: *2015 IEEE 16th International Conference on Communication Technology (ICCT)*. IEEE. 2015, pp. 26–31.

[14] Tyler Ross Lambert. "An introduction to microcontrollers and embedded systems". In: *Auburn University. July* 344 (2017).

[15] Jesús Lizarraga, Roberto Uribeetxeberria, Urko Zurutuza, and Miguel Fernández. "Security in embedded systems". In: *Computer Science Department, Mondragon University, Spain* (2006).

[16] Rwan Mahmoud, Tasneem Yousuf, Fadi Aloul, and Imran Zualkernan. "Internet of things (IoT) security: Current status, challenges and prospective measures". In: *2015 10th international conference for internet technology and secured transactions (ICITST)*. IEEE. 2015, pp. 336–341.

[17] Nataliia Neshenko, Elias Bou-Harb, Jorge Crichigno, Georges Kaddoum, and Nasir Ghani. "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations". In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2702–2733.

[18] Carmine Noviello. *Mastering STM32: A Step-by-step Guide to the Most Complete ARM Cortex-M Platform, Using a Free and Powerful Development Environment Based on Eclipse and GCC*. Leanpub, 2016.

[19] R. Ab Rahman, U. R. Hashim, and S. Ahmad. "IoT based temperature and humidity monitoring framework". In: *Bulletin of Electrical Engineering and Informatics* 9.1 (2020), pp. 229–237. DOI: 10.11591/eei.v9i1.1557.

[20] V Ramesh, M Sankaramahalingam, MS Divya Bharathy, and R4 Aksha. "Remote temperature monitoring and control using IoT". In: *2017 International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE. 2017, pp. 1059–1063.

[21] Rapeepat Ratasuk, Nitin Mangalvedhe, and Amitava Ghosh. "Overview of LTE enhancements for cellular IoT". In: *2015 IEEE 26th annual international symposium on personal, indoor, and mobile radio communications (PIMRC)*. IEEE. 2015, pp. 2293–2297.

[22] Karen Rose, Scott Eldridge, and Lyman Chapin. "The internet of things: An overview". In: *The internet society (ISOC)* 80.15 (2015), pp. 1–53.

[23] OPC Router. *Was ist MQTT? Erklärung mit industriellem Fokus*. Accessed: February 28, 2024. URL: https://www.opc-router.de/was-ist-mqtt/.

[24] Jalpa Shah and Biswajit Mishra. "Customized IoT enabled wireless sensing and monitoring platform for preservation of artwork in heritage buildings". In: *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE. 2016, pp. 361–366.

[25] Dipa Soni and Ashwin Makwana. "A survey on mqtt: a protocol of internet of things (iot)". In: *International conference on telecommunication, power analysis and computing techniques (ICTPACT-2017)*. Vol. 20. 2017, pp. 173–177.

[26] Statista Search Department. *Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030 (in billions)*. URL: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[27] STMicroelectronics. *Capacitive digital sensor for relative humidity and temperature*. Version 5.1. 2023.

[28] STMicroelectronics. *Ultracompact high-performance eCompass module: ultralow-power 3-axis accelerometer and 3-axis magnetometer*. Version 11.0. 2022.

[29] STMicroelectronics. *UM2743Discovery kit for LTE Cat M/NBIoT with STM32L4 Series*. Version 1.0. 2021.

[30] Ahmad F Subahi and Kheir Eddine Bouazza. "An intelligent IoT-based system design for controlling and monitoring greenhouse temperature". In: *IEEE Access* 8 (2020), pp. 125488–125500.

[31] Telekom. *Einfach erklärt: LTE-M | Deutsche Telekom*. Accessed: February 29, 2024.

[32] Telenor. *LTE-M vs NB-IoT - a guide and in depth comparison*. Accessed: February 29, 2024.

[33] Telit. *What Is LTE, and How Is It Used in IoT?* Accessed: February 29, 2024.

[34] Thalesgroup. *What is 4G LTE? Long Term Evolution Explained*. Accessed: February 29, 2024.

[35] The Investopedia Team. *What is Cloud Computing? Pros and Cons of Different Types of Services*. Accessed: March 04, 2024. 2023. URL: https://www.investopedia.com/terms/c/cloud-computing.asp.

[36] Adam Thierer and Andrea Castillo. "Projecting the growth and economic impact of the internet of things". In: *George Mason University, Mercatus Center, June* 15 (2015).

[37] Tutorialspoint. *LTE Overview*. Accessed: February 29, 2024.

[38] L. S. Vailshery. "Number of IoT connected devices worldwide 2019-2023, with forecasts to 2030". In: *Statista* (2023).

[39] Jingwen Zhang, Meiting Guo, Biyan Li, and Ruimin Lu. "A transport monitoring system for cultural relics protection based on blockchain and internet of things". In: *Journal of Cultural Heritage* 50 (2021), pp. 106–114.

Abbreviations

IoT	Internet of Things
3GPP	3rd Generation Partnership Project
UMTS	Universal Mobile Telecommunication System
GSM	Global System for Mobile Communications
VoIP	Voice over IP
TDD	Time Division Duplex
FDD	Frequency Division Duplex
LPWA	Low Power Wide Area
LPWAN	Low-Power Wide-Area Network
LTE	Long Term Evolution
IT	Information Technology
AWS	Amazon Web Services
VPN	Virtual Private Network
IPSec/IPv6	Internet Protocol Security
BLE	Bluetooth Low Energy
GPRS	General Packet Radio Service

List of Figures

3.1 Number of IoT connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030, source: [26] 6

3.2 B-L462E-CELL1 Discovery kit (top view), source: [29] 7

3.3 B-L462E-CELL1 Discovery kit (bottom view), source: [29] 8

3.4 Diagram illustrating the operation of the MQTT protocol 9

4.1 Artwork Tracking Scenario 14

4.2 System Architecture 16

4.3 Route Experiment 1 17

4.4 Route Experiment 2 17

4.5 Route Experiment 3 18

5.1 Core Directory Structure 20

5.2 Drivers/BSP Directory Structure 21

5.3 Tera Term Configuration 24

5.4 Tera Term Log NetworkState 25

5.5 External SIM Configuration 26

5.6 Environmental Data Tera Term 27

6.1 Environmental Data Experiment 0 34

6.2 Backpack scenario 35

6.3 Environmental Data Experiment 1 36

6.4 Environmental Data Experiment 2a 37

6.5 Environmental Data Experiment 2b 38
6.6 Failed Environmental Data Experiment 3 39
6.7 Environmental Data Experiment 3 40

List of Tables

2.1 Paper Overview 4

4.1 Experiments Overview Design 18

6.1 Experiments Overview Evaluation 41

Appendix A

Installation Guidelines

Steps to Use the Monitoring System

1. Download Zip File
    - Download the project zip file from [https://github.com/Islemmdimagh/Bachelor-Thesis-IoT-Network].
2. Connect the Board (USB ST-LINK)
    - Connect the board to your device using a USB micro cable.
3. Launch Tera Term
    - Launch Tera Term, select 'Serial' and choose 'COM3: STMicroelectronics STLink Virtual COM Port (COM3)' as the 'Port'.
    - In Tera Term, under 'Setup' > 'Serial Port...', adjust the baud rate to 115200.
    - In Tera Term, go to 'File' > 'Log...' to start logging data with timestamps.
4. Run the Temperature Project
    - Run the 'Temperature' project on the connected board.
5. Run TeraTermPlot.py
    - Specify the log file name and location chosen in step 3.
    - After logging data, run TeraTermPlot.py.