University of Zurich[UZH]

MASTER PROJECT – Communication Systems Group, Prof. Dr. Burkhard Stiller

CERTIFY Artwork Tracking: Backend Services and Bootstrapping Process

Yves Meister
Uster, Switzerland
Student ID: 19-703-255

—

Robin Bättig
Dübendorf, Switzerland
Student ID: 18-736-371

Supervisor: Katharina O. E. Müller, Thomas Grübl, Daria Schumm, Prof. Dr. Burkhard Stiller
Date of Submission: June 15, 2025

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

ifi

Master Project
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, 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 All passages taken verbatim or in sense from published or unpublished writings are identified as such.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, 15.06.2025

Signature of student

Abstract

The Internet of Things (IoT) enables real-time tracking of art, yet few projects have explored mobile and secure lightweight implementations of this. Existing solutions often remain stationary or lack lightweight security capabilities. This makes transport implausible, and security expensive or impossible. This thesis attempts to address this gap by creating a cloud-based, mobile and MUD secured system for artwork tracking. To achieve this goal, a STM32L4 board with humidity and temperature sensors is used. Using cellular IoT, the system sends the collected data to an AWS-hosted server structure. This server structure integrates the MUD protocol to ensure security via enforceable network behavior. Real world test runs are used to show sensible data collection, as well as stable network connection in various use cases and locations. The results show that the approach of this thesis is both a lightweight and viable solution to protect artworks in both stationary and mobile environments.

Zusammenfassung

Das Internet of Things (IoT) erlaubt Echtzeitverfolgung von Kunst, jedoch haben nur wenige Projekte die mobile, sichere und ressourcenschonende Implementierung dessen untersucht. Bestehende Lösungen bleiben oft stationär oder haben keine, oder ressourcenlastige, Sicherheitsfunktionen. Das macht Transport implausibel und Sicherheit teuer oder unmöglich. Diese Arbeit versucht mit einem cloudbasierten, mobilen und MUD gesicherten System für Kunstwerkverfolgung diese Lücke anzugehen. Um dieses Ziel zu erreichen, wird ein STM32L4-Board mit Feuchtigkeits- und Temperatursensoren benutzt. Mithilfe von Cellular IoT sendet das System die gesammelten Daten an eine auf AWS gehostete Serverstruktur. Diese Serverstruktur integriert das MUD-Protokoll um die Sicherheit durch erzwingbares Netzwerkverhalten zu garantieren. Testläufe in der echten Welt werden verwendet, um eine sinnvolle Datenerfassung sowie eine stabile Verbindung in verschiedenen Nutzungsfällen und Standorten zu zeigen. Die Resultate zeigen, dass der Ansatz dieser Arbeit sowohl eine ressourcenschonende als auch eine praktikable Lösung ist, um Kunstwerke in stationärer und mobiler Umgebung zu schützen.

Contents

Chapter 1

Introduction

1.1 Motivation

The Internet of Things (IoT) has drastically changed industries by allowing smart devices to autonomously collect and send data. One sector where this evolution holds untapped potential is the art world, especially in monitoring and protecting artwork during transit and storage. Artworks are often delicate, vulnerable to shifts in temperature or humidity that could cause irreversible damage, such as warping, cracking, or fading. The tracking of such environmental factors is a region that IoT thrives in. Advancements in this field have been made, but are most often limited to static art within museums [13] [2] or require a central device to handle communication to the larger network [4], as discussed in Section 3. This thesis aims to develop a more adaptable and cost-effective IoT system capable of tracking both single and multiple artworks, whether stationary or on the move, with an added layer of network security.

As seen in the title of this thesis, it is a part of the broader, EU-funded CERTIFY project. That project focuses on life-cycle management for IoT devices. CERTIFY wants to create a framework for security across many IoT systems. It uses concepts like security-by-design, continuous monitoring, secure over-the-air updates, and collaborative risk assessment. With these, CERTIFY attempts to provide robust protection to IoT systems. This thesis tries to contribute to CERTIFY's mission by using its principles for artwork tracking.

1.2 Thesis Goals

The central aim of this project is to create a secure IoT monitoring solution that uses ST-boards to gather environmental data and transmit it via the Manufacturer Usage Description (MUD) protocol to a cloud-based setup hosted on Amazon Web Services (AWS). The four key objectives are as follows:

First, to build a two-server infrastructure in the AWS cloud that can handle incoming data from the ST-boards, as described in Section 4.2

Second, to integrate MUD as a security layer, giving tighter control over device behavior and helping defend against potential threats, as shown in Section 4.5.

Third, to enhance the ST-boards' capabilities so they don't just read sensor data but can autonomously send this information over a mobile network as described in Section 4.4.

And finally, to test the entire system in real-world conditions, as shown in Section 4.8.

The deliverables of this thesis include a fully functional IoT-based artwork tracking system, complete documentation of the system's design and implementation, a communication protocol leveraging MUD, and analysis based on real-world testing of the system.

1.3 Methodology

The approach taken in this thesis was mostly practical. The focus was on building a functional prototype for an IoT-based artwork tracking system. The project would need to track artworks by monitoring environmental conditions like temperature and humidity and use the mobile cellular network to transmit data. Once the goals were clear, the system architecture was designed accordingly. The original idea was to use STM32L4 boards with built-in temperature, humidity, and acceleration sensors and connect them to a cloud-based server system hosted on AWS via the cellular network. The acceleration measurements would later be scrapped, as discussed in Section 5.8.6. Security wasn't an afterthought. From the start, the system design included the MUD protocol to restrict and monitor device communication. Two AWS-based servers were to be used: one to manage communication and MUD rules, and another to store the MUD files. The server structure is further discussed in Section 4.2.

The project moved forward in phases. First, the servers were to be configured to communicate with each other and with the ST-boards. Then, the ST-boards were programmed to collect sensor data and transmit it. For testing, an Android app was to be created to gather GPS data alongside sensor data, as shown in Section 4.7. Real-world trials would be conducted in multiple stages, with and without full server integration, as shown in Section 4.8. During these tests, data from the sensors, server logs, and GPS locations were analyzed to validate performance, highlight areas for improvement and to deliver a proof of concept.

1.4 Thesis Outline

The thesis is structured into seven chapters. It begins with this introduction, detailing the motivation behind the project, the goals set, the methodology applied, and a brief layout of the document. In Chapter 2 the fundamental background information is displayed. Here key concepts such as Internet of Things, ST-boards, Manufacturer Usage Description, etc. are explained. Chapter 3 covers the related works and research in the broader area of artwork tracking. Chapter 4 dives into system design. It describes how the different

parts of our artwork tracking system are planned to work together. This chapter also entails the plans for proof of concept test runs of the system. In Chapter 5 we detail the implementation of the concepts described in the preceding chapters. A substantial part of the project was handling issues that arose, therefore those are also showcased within the chapter. Chapter 6 covers the evaluation of the system. It explains how test runs were set up, what kind of data was collected, and what results were observed. It also entails a discussion as to what the results mean. Finally, Chapter 7 gives a quick summary, concluding thoughts, and suggestions for possible future development.

Chapter 2

Fundamentals

2.1 Overview

This chapter outlines the core technologies and concepts essential to understanding the system built in this thesis. A foundational grasp of IoT, ST-Boards, MUD, and related technologies is critical for what follows.

2.2 Internet of Things

The Internet of Things (IoT) can be defined as a network of devices or ”things” embedded with sensors, software, and other technologies to connect and exchange data with other ”things” and systems over the internet. These ”things” can range from everyday household items to complex industrial machines and tools. Using this concept of IoT, devices can sense, collect, and send data, allowing smart behavior and automated processes that no longer require direct human intervention. The ever increasing number of such connected devices, combined with progress in sensor technology, cloud computing, and machine learning, has transformed IoT into a groundbreaking and foundational technology of modern day automation and digital transformation [9].

IoT systems rely on several core components to function. Firstly, sensors are the most fundamental part of most IoT systems. Their purpose is to collect data and convert it into a digital signal. In the case of this project those sensors include temperature and humidity measurements. Also IoT requires embedded systems and communication protocols. In this project, the ST-board acts as the brain of the IoT device. It collects the sensors' data and communicates it to the network. In order for communication to be possible protocols need to be in place. In this project the communication protocol is 4G LTE via cellular network. The system also requires a way of storing and computing. This is most often done using a cloud service. Lastly the system needs interfaces, that allow visualization and interaction with the IoT [9].

2.3 Cellular IoT

Cellular Internet of Things (CIoT) technologies require little infrastructure and can be very scalable. Unlike protocols such as Wi-Fi or Bluetooth, which need to be close to routers or gateways, CIoT enables long-range connections through regular mobile cellular networks. This way devices can operate across large distances or even in motion.

In practice, this is a big advantage for use cases such as this thesis, where consistent, secure data transmission must be possible even in movement. Whether an artwork is transported across town or between countries, cellular coverage can ensure uninterrupted tracking.

The standards behind CIoT are maintained by the 3rd Generation Partnership Project (3GPP), which defines multiple protocols like EC-GSM-IoT, LTE-M (Long Term Evolution for Machines), NB-IoT (Narrowband IoT), and NR-RedCap (Reduced Capability 5G). Each of these options caters to different requirements, from ultra-low power consumption to low-latency performance, making CIoT highly adaptable [12].

In the context of this thesis, CIoT allows ST-boards to transmit sensor data in real time, directly over the cellular network, without needing a nearby receiver. This independence from local infrastructure makes the system flexible and more suitable for real world deployment scenarios. The use of licensed, secure spectrum also enhances reliability and safeguards the transmitted data against unauthorized access.

Combined with cloud-based backend, this architecture offers a modern, mobile approach to IoT deployments, one that is suited to protecting valuable and delicate items like artworks during transportation.

2.4 ST-Boards

Built by STMicroelectronics, ST-boards are used in many IoT systems. According to STM, they are good at balancing processing power with low energy consumption. The STM series offers integrated sensor support, network communication, and real-time operating capabilities. These features are all important for this project's goals [14].

The STM32L4 boards chosen here come with temperature, humidity, and acceleration sensors, as well as SIM card slots for cellular communication functionality.

2.5 Manufacturer Usage Description

The Manufacturer Usage Description (MUD) is a framework introduced by the Internet Engineering Task Force (IETF) to enhance the security of IoT devices by defining how these devices should behave within a network. As IoT devices grow in number, so does the risk of misuse or attacks. MUD provides a mostly standardized mechanism to specify

the intended network behavior of a device, ensuring that it only communicates with authorized endpoints. This approach limits the attack surface and makes it harder for threats to make use of vulnerabilities in IoT systems [8].

MUD works via a MUD file, which is a JSON-based document created by the device manufacturer or service provider. This file describes how a device is expected to communicate, including allowed IPs, protocols, and ports. When a device, such as an ST-board, used for environmental sensing in this thesis, connects to the network, it sends a MUD URL, which may be any HTTPS address, that points to its MUD file. A server side MUD manager retrieves this file and translates the policies into rules that are enforced across a part of the network infrastructure. This ensures that the device's communication is limited to before approved interactions, preventing unauthorized access or unintended and faulty data transmission. Additionally, MUD makes network management easier as it automates creation and application of access control policies. Instead of a human having to manually configure access rules, the MUD manager, in this thesis: Bootstrapping Server, automatically handles this tasks based on the device's stored MUD file [8].

In this thesis, where an ST-board collects temperature and humidity data and sends it to a server, MUD ensures that the data is transmitted securely. The MUD file specifies that only registered boards can communicate with the server over only the designated Port. This safeguards the server from malicious actors and ensures that the board works as intended. It is important to note, that MUD protects the System from threats to the device and not from the device as a threat itself.

2.6 Cloud-Based IoT Architecture

Besides IoT devices with functional sensors, an artwork tracking system needs a good backend. In this thesis, Amazon Web Services (AWS) serves as the backbone. Their cloud infrastructure is used for sensor data storage, policy enforcement, logging and MUD file management.

For artwork tracking in general, this type of architecture ensures continuous access to environmental data from multiple mobile units, even when they are not connected to the same local network. It also provides a centralized point to enforce network policies via MUD. It also allows for future enhancements like automated anomaly detection or user alerts through notification services.

2.7 Artwork Tracking

Many artworks such as paintings, sculptures, and installations, are made and transported every day. These countless artworks need to be shipped to exhibitions, museums and the homes of art collectors.
Artworks are usually relatively fragile and susceptible to humidity, physical impact or even temperature changes. This presents a lot of challenges when transporting such delicate

goods from one place to another without harming them. Additionally, keeping track of the artworks themselves is important such that they don't get lost or stolen. These challenges make IoT-based tracking systems particularly valuable, as they offer continuous environmental monitoring, reducing the risk of damage.

Chapter 3

Related Work

3.1 Overview

The following chapter is a collection of similar projects and papers. In Section 3.2 we show what has already been achieved in the realm of artwork tracking. We also show what differences exist between existing solutions and our thesis. In Section 3.3 we show work that has already made use of the Manufacturer Usage Description standard. Here we show that MUD is being used in practice and also the differences between those projects and our own.

Related Works				
Citation	Year	Transmission	Sensors	Stationary/ Mobile
Fort et al.[4]	2022	BLE	Temperature, Humidity	Stationary
Zhang et al.[18]	2021	4G	Speed, Temperature, Humidity, GPS, Acceleration, Video	Mobile
Shah and Mishra[13]	2016	Customized Hopping Method	Temp, Humidity, Light	Stationary
Ch. B. V et al.[2]	2021	WiFi	Temp, Humidity, Light	Stationary
Trigona et al.[17]	2022	Bluetooth	Microclimate, Vibration	Stationary
Klein et al.[7]	2017	Low-power Mote Technology	Temp, Humidity, Air quality	Stationary

Table 3.1: Summary of Related Works mentioned in Chapter 3.2.

3.2 Artwork Tracking

Existing IoT art monitoring solutions are often developed for stationary objectives such as the preservation of artifacts in museums or monitoring of historic buildings. Fort et al. [4] propose one such system that uses Bluetooth Low Energy (BLE) to monitor environmental conditions and vibrations in wooden structures and artwork. Their approach integrates multiple sensing modalities into a single compact platform, making it effective in stationary settings. It must remain close to a BLE gateway to function. This limits its usefulness for mobile tracking.

Zhang et al. [18] present a system that is more aligned with the goals of this thesis. Their monitoring platform uses 4G. This eliminates the need for a local receiver and enables tracking of artworks during transport. In contrast to this thesis, their solution uses blockchain technology for logging and verifying environmental data. Blockchain can increase transparency, but it also increases complexity and processing overhead. In this project a simpler solution is used that relies on secure and predictable network behavior, thanks to MUD.

Shah and Mishra [13] and Ch.B.V et al. [2], describe IoT systems used to monitor heritage objects in buildings like temples or museums. Their projects have temperature, humidity, and light sensors to measure environmental stability. They are effective in stationary installations but aren't suited for mobile applications. This is because they depend on WiFi-based data transmission and require closely located receivers. Needing receivers makes them unsuited for tracking individual items during movement or storage outside of their network range.

Trigona et al. [17] also explore a similar domain. Their IoT system combines microclimate and vibration sensing. Their setup that is deployed within a historic monastery, demonstrates how environmental and structural data can be collected compactly. Their board is capable of logging multiple environmental parameters, but data transfer also relies on Bluetooth and even requires interaction with a mobile device. This limits automation and can increase costs. Still, their approach shows the benefit of using multiparameter sensing.

Another large-scale example is presented by Klein et al. [7]. They describe a wireless sensing platform deployed in The Cloisters, a branch of the Metropolitan Museum of Art. With over 200 nodes distributed across multiple galleries, their system analyzed temperature, humidity, and air quality trends over time. The sensor density allowed the researchers to identify localized changes and even impacts of visitors. Their system was designed exclusively for static indoor use and requires continuous power.

3.3 MUD

Security in IoT networks is an increasingly important topic, especially in very autonomous systems that make use of public networks. The Manufacturer Usage Description standard

from RFC 8520 [8], as described in Section 2.5, tries to tackle this topic. It allows device manufacturers or device administrators to define wanted network behavior.

Multiple studies demonstrate the utility and extensibility of the MUD standard. Feraudo et al. [3] expand upon the MUD framework, by introducing mechanisms for dynamic traffic filtering and rate-limiting. They use tools such as eBPF and iptables. Their implementation demonstrates how MUD can be used to limit traffic and reduce attack surfaces.

Heeb et al. [6] analyze the impact of MUD on IoT security. Their paper shows strengths of the standard, such as mitigating DDoS attacks. But it also highlights some big weaknesses, in the form of application layer vulnerabilities. In general they conclude that MUD is effective on the transport layer at reducing attack surfaces, but ineffective on the perception and the application layer. They note that MUD is especially vulnerable to spoofing attacks.

3.4 Summary

In summary, many existing systems provide monitoring capabilities in static locations, some address the mobile use case and others have built-in network security or simplicity. The work presented in this thesis aims to fill a gap by proposing a cellular IoT system secured through MUD. This allows individual and mobile artwork tracking with minimal complexity and high security. The study by Heeb et al. [6] reinforces the importance of MUD to secure IoT communication. But it also shows that depending on the use case, MUD alone may not be sufficiently secure.

Chapter 4

Design

4.1 Overview

The system is designed as a cellular IoT-based artwork tracking solution, consisting of ST-boards as nodes for data collection, a communication protocol, utilizing MUD, as discussed in 2.5, for data transmission, and a cloud-based server infrastructure on AWS for data processing and storage.

4.2 Server Setup

A two-server infrastructure was designed with AWS, to enable the artwork tracking system. The two servers are called Bootstrapping Server and MUD Server.

The Bootstrapping Server serves as the main point of contact for the IoT devices. It is responsible for handling all incoming communication from the ST-boards over the cellular network, as described in Section 4.4. This server listens for data transmission via TCP sockets. It also enforces MUD policies on the connections. It retrieves the MUD files from the second server, the MUD Server, as further discussed in Section 4.5. For security reasons, the server only allows communication with the MUD Server, registered IoT devices and if necessary other admin IP addresses. The sole exception is a registration port, where all communication must remain open to allow new boards to connect. The server uses multiple threads to allow different communication channels, including TCP with IoT devices and HTTP with the MUD Server.

The MUD Server's task is to store and send MUD files corresponding to each IoT device. Every MUD file contains information on allowed network traffic and communication behavior. The server has endpoints for uploading and downloading MUD files. It must also have signing functionality, as stated in RFC8520 [8]. This is important, to ensure authenticity and integrity of sent and received MUD files. Both of the servers are deployed as AWS EC2 instances.

4.3 Board Setup

The boards have the tasks of collecting sensor data and transmitting it to the server structure. To do this the plan for the boards is to combine and enhance template code from STMicroelectronics and the bachelor's thesis by Mdimagh [11]. The template code should have basic socket capabilities upon which the communications structure can be built. The code from the bachelor's thesis should have basic calls to the board's sensors. Both these code bases need to be majorly updated and changed to fit the needs of this thesis. It is described in Section 5.8.2 why this design choice had to be changed in the implementation phase of the project.

4.4 Server and Board Communication

The ST-boards, equipped with environmental sensors, connect to the internet using a LTE cellular module integrated into the B-L462E-CELL1 IoT Discovery Kit. Data transmission from the board to the Bootstrapping Server is performed via TCP sockets, allowing for lightweight, direct communication without the overhead of HTTP. On the server side, the Bootstrapping Server listens on a dedicated TCP port for incoming connections from ST-boards.

During the initial connection, a board transmits its associated MUD URL, as described in Section 4.5. This allows the Bootstrapping Server to fetch the board's MUD file from the MUD Server. Once the file is retrieved the Bootstrapping Server enforces the rules as iptables. This completes the board's registration process.

Each device, upon registering, can transmit messages containing environmental sensor data (temperature and humidity), signal strength and an internal timestamp. The data is sent automatically as soon as the board measures as sensor datapoint.

This communication model prioritizes simplicity. More sophisticated IoT communication protocols certainly exist. The choice of TCP was based on the requirements of this project and the capabilities of the boards.

4.5 MUD Flow Design

The MUD flow starts with the ST-boards. Each IoT device must have an inherent MUD URL, that defines the location of the device's MUD file. This URL is to be sent to the Bootstrapping Server to register. The server in turn retrieves the MUD file from the MUD Server and enforces its rules on the network traffic with the IoT device. To ensure integrity, transmissions are verified according to RFC8520[8].

The MUD Server must only allow communication directly from the Bootstrapping Server and trusted IPs of admins that are authorized to add new MUD files. It must implement

endpoints to store new MUD files and to retrieve stored MUD files. The server must also implement a signing method according to RFC8520 [8].

The Bootstrapping Server must generally only allow communication from the MUD Server, the registered IoT devices, and trusted IPs of admins. The exception is the port used for registration, as this port must remain open for all communication. The Bootstrapping Server must be able to communicate to the MUD Server via HTTP and the IoT devices via TCP. To accomplish this, the Bootstrapping Server shall, using multiple threads, listen for TCP on two ports (one for registration and one for data transmission) and HTTP on another. It must implement an HTTP endpoint to retrieve MUD files from the MUD Server. It must implement the functionality to read, interpret and enforce MUD files. It must also be able to manage multiple registered boards.

4.6 Sensor Data Collection

The sensor data collection is done via ST-boards, as explained in 5.4. This hardware portion comprises of B-L462E-CELL1 IoT discovery kits by STMicroelectronics, which encompass STM32L4 boards fitted with humidity, temperature and acceleration sensors. Our modified boards also have a SIM slot and antenna to enable cellular connection.

Data collection shall be done at regular intervals. The time of data collection must also be recorded. As the boards have no global clock, their internal time is appended to the data and is converted to a timestamp server side, as described in Section 5.8.5. The recorded data is to be converted to a reasonable format and sent to the server.

The data is transmitted to the AWS server structure via cellular sockets using X-CUBE-CELLULAR. Data is sent via TCP. Currently the system is not designed to do anything with this data other than store it. Data transmission serves more as a proof of concept.

4.7 GPS Data Collection

GPS data is not necessary for the function of the system. Although, for testing purposes GPS data is very important and in future research should be integrated, as discussed in Section 7.3. Collecting GPS data in parallel to the other data, allows us to be able to geographically trace where issues occur. The ST-boards do not have an inherent GPS sensor, therefore, an alternative approach is necessary. An Android app shall be implemented that periodically records GPS location and a timestamp. This allows us to later combine the sensor and GPS data for testing, analysis and illustration purposes.

4.8 Test Design

The functionality and reliability of all parts of the system need to be verified. To do this, two test runs were planned and executed. The test design was focused both on checking

the individual components, as well as the combined system. The tests were split into two main phases.

In the first phase the ST-boards should be tested without active servers. This was to ensure that the integrated sensors worked correctly. At the same time a GPS tracking solution was to be tested, as described in Section 4.7. During these first runs, the GPS tracking would run in parallel with pinging an echo server, to ensure cellular connectivity, and then in parallel with the sensors (see Section 6.1).

The second phase also integrated the server structure. Here, the ST-boards transmitted sensor data via cellular sockets to the Bootstrapping Server. The Bootstrapping Server would process the data and save it to a .txt file, for further analysis. Each data point is saved in JSON format. Some sample data is shown in Figure 4.1. These tests also used the final version of the GPS tracking Android app, as described in Section 5.7.

Both phases would be real-world scenarios. The boards and GPS tracking devices would be taken around the suburbs of Zürich by train and car respectively. Later the GPS and other data would be matched using timestamps and analyzed.


```
{"time": "11750", "dbm": "0", "temperature": "28.322449", "humidity": "43.508064", "rtc_timestamp": "2025-05-10 13:40:15.242"}
{"time": "14039", "dbm": "-79", "temperature": "28.322449", "humidity": "43.508064", "rtc_timestamp": "2025-05-10 13:40:18.215"}
{"time": "16995", "dbm": "-93", "temperature": "28.322449", "humidity": "43.508064", "rtc_timestamp": "2025-05-10 13:40:20.527"}
{"time": "19335", "dbm": "-77", "temperature": "28.322449", "humidity": "43.508064", "rtc_timestamp": "2025-05-10 13:40:22.839"}
{"time": "21638", "dbm": "-79", "temperature": "28.340136", "humidity": "42.825932", "rtc_timestamp": "2025-05-10 13:40:25.569"}
```


Figure 4.1: Example data for the second phase.

Chapter 5

Implementation

5.1 Overview

The system consists of a MUD Server, a Bootstrapping Server, ST-boards and a GPS App. The implementation of the system was done in accordance to the design, in Section 4, apart from the changes discussed in Section 5.8. The server infrastructure was implemented using AWS and Python flask and socket servers. The sensor data collection and transmission was done using the ST-board sensors and sockets from X-CUBE-CELLULAR. The GPS data collection was implemented in Kotlin using Android Studio. All relevant code used in this project can be found on GitHub [1].

5.2 General Server Structure

The project needed a server structure that acts as a point of contact for the boards and can store, retrieve, and enforce MUD policies. The structure should also have the capability of receiving sensor data and storing it. To do this, a Bootstrapping Server and a MUD Server were set-up using AWS. The general server architecture is depicted in Figure 5.1.

Communication between the ST-boards and the server infrastructure is performed over TCP sockets using the cellular network connection provided by the B-L462E-CELL1 IoT Discovery Kit. This choice was made for its simplicity, low overhead, and ease of integration with the X-CUBE-CELLULAR software package. On the server side, the Bootstrapping Server and MUD Server are implemented in Python using the flask and socket libraries.

The Bootstrapping Server receives communications from the ST-board through the cellular network using TCP. This can either be a sensor data point or a request to enter the network using a MUD URL. The Bootstrapping Server then sends a request to the MUD Server to get a MUD file and a certificate URL , which will later be used for retrieving the certificate from the MUD Server. The MUD Server then retrieves the stored MUD file,

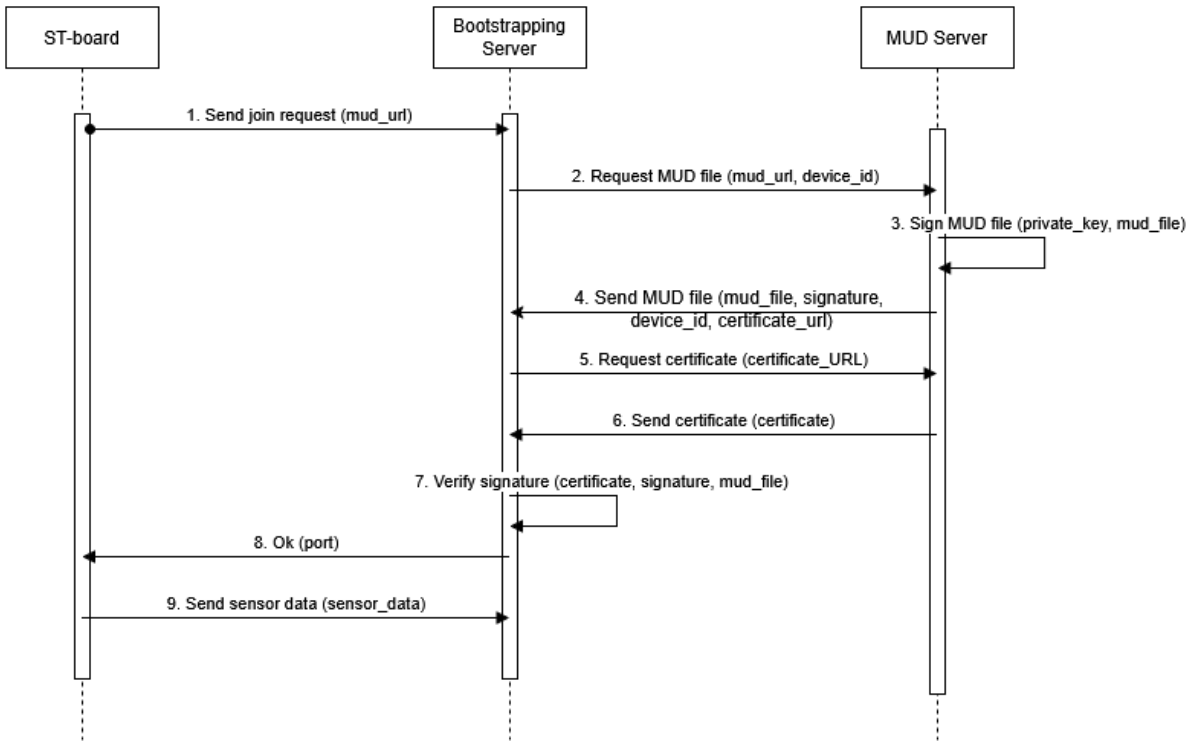


Figure 5.1: Server Architecture

signs it and sends it to the Bootstrapping Server. Using the aforementioned certificate the Bootstrapping Server verifies the signature before enforcing the IP table rules contained within the MUD file upon itself. This allows the requesting ST-board to send data to the Bootstrapping Server. The last step means sending the data receiving port to the ST-board where it can send the sensor data to.

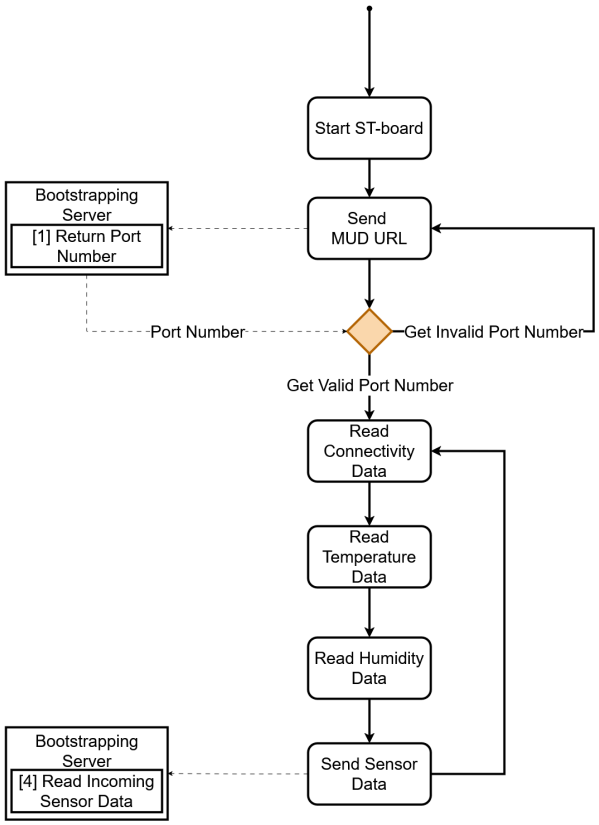


Figure 5.2: ST-board Architecture

5.3 Board

The code on the ST-board is based on template code from STMicroelectronics. It includes functionality to manually connect to a socket of a specified address. It is also able to read sensor data from the humidity sensor and the temperature sensor. Furthermore it logs data, such as signal strength.

This code base was extended by introducing automatic socket connecting because sending data over sockets via manually sending commands to the board is not practical. This allows the board to automatically send the connection request to the Bootstrapping Server as seen in the box marked ”1” in Figure 5.2. This request includes sending a MUD URL. The Bootstrapping Server will answer with either an error port number or the port number of the endpoint where the sensor data should be sent to.

It also automates the process of repeatedly sending sensory and connectivity data to the Bootstrapping Server, see the box marked ”4” in Figure 5.2. The frequency of the polling for sensory data can be changed at runtime by the user via the command `mems period <milliseconds>`. However the frequency of sending data to the Bootstrapping Server cannot be changed at runtime. We used Tera Term [16] to monitor the outputs from the board and to send commands to the board.

5.4 Sensor Data

The ST-board used in this project is equipped with sensors capable of measuring temperature and humidity. These sensors are accessed via the STM32Cube hardware abstraction layer and configured using the STM32CubeIDE environment. The sensor data is collected periodically and sent. This happens approximately every 3 seconds. The sensor readings include temperature in degrees Celsius and relative humidity in percent. The ST-board lacks a real-time clock. Therefore, each data measurement packet is appended with an internal relative timestamp. This timestamp shows how long the board has been on, or when it has last been reset. On the server side, this timestamp is combined with the server's timestamp, to approximate a time of measurement. This time is limited in accuracy due to latency, as further discussed in Section 5.8.5.

5.5 Bootstrapping Server Structure

Booting up the Bootstrapping Server means the start of three different processes. The first process is about the registration of new devices.

The Bootstrapping Server has a dedicated port to receive registration requests from IoT devices, in our case the ST-board. The Bootstrapping Server expects a MUD URL on this port , see box marked "1" in Figure 5.3. This URL is to be contacted via HTTP by the Bootstrapping Server. It points to the MUD Server responsible for the IoT device in question. The MUD Server responds with the certificate URL, the MUD file, the signature of the MUD file and the device ID, see box marked "3" in Figure 5.3.

The certificate URL is the Bootstrapping Server's target to retrieve the certificate issued by the MUD Server, see box marked "2" in Figure 5.3. The certificate is used to confirm the integrity of the MUD file. The certificate is used to decrypt the signed version of the MUD file. Then the content of the signed MUD file is compared with the content of the unsigned MUD file. After, the IP table rules in the MUD file are enforced and the board gets to send sensor data to another socket port according to the IP table rules. The device ID is saved as a device instance along the MUD URL and IP address.

The second process is about reading incoming sensor data from a registered ST-board. The server reads incoming data, see box marked "4" in Figure 5.3, transforms the timestamp data (see Section 5.4), and stores it locally. This data is used for analysis purposes and as a proof of concept. In a real-world application this data would be further manipulated and potentially handled by a different server altogether.

The third process is about updating MUD files from registered boards. Every given time period the Bootstrapping Server requests the MUD file from the MUD Server for each currently registered STM board. Just like in the initial registration the MUD files get validated using the corresponding certificate. Then they are enforced. If no MUD file or certificate was found on the MUD Server or the validation failed all IP table entries associated with the current board are removed and the board is unregistered. This stops

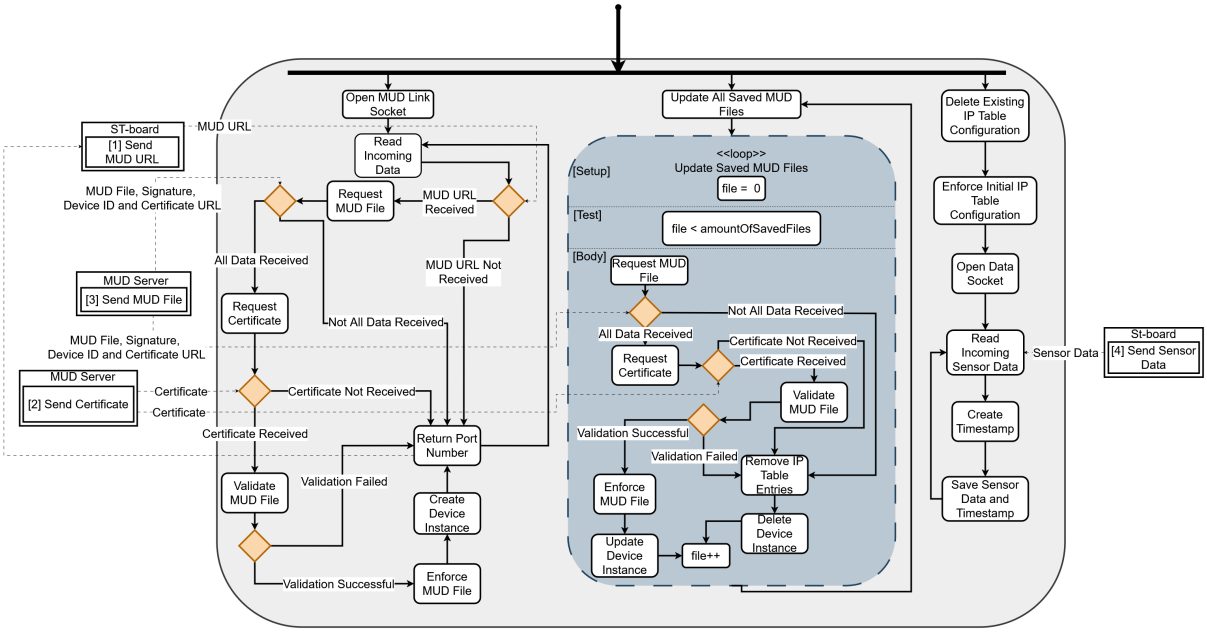


Figure 5.3: Bootstrapping Server Architecture

all sensor data traffic coming from this device and it has to re-register to send new sensor data.

5.6 MUD Server Structure

The MUD Server uses RSA, a public key cryptography system, to maintain integrity of MUD files. So a RSA key pair is generated on boot-up. Then the MUD Server checks for existing MUD files in its directory and creates device instances for device ID and the MUD file.
Now three processes start simultaneously.

The first listens to incoming MUD files. This is a HTTP endpoint for device manufacturers and alike to upload MUD files and the device ID to the MUD Server from outside, see box marked ”5” in Figure 5.4. Just like in the boot-up procedure device instances are created with device ID and MUD file for each received MUD file.

The second process is about certificate requests. The second process listens on another HTTP port for requests from Bootstrapping Servers requesting the certificate. If the certificate is successfully retrieved it is sent back to the requesting Bootstrapping Server, see box marked ”2” in Figure 5.4.

The third process listens on a third HTTP port for MUD file requests. It receives a device ID from a Bootstrapping Server. If such a device is found the MUD file is retrieved and signed. If no errors occurred the signature, the MUD file, the device ID and the certificate URL are sent back to the Bootstrapping Server.

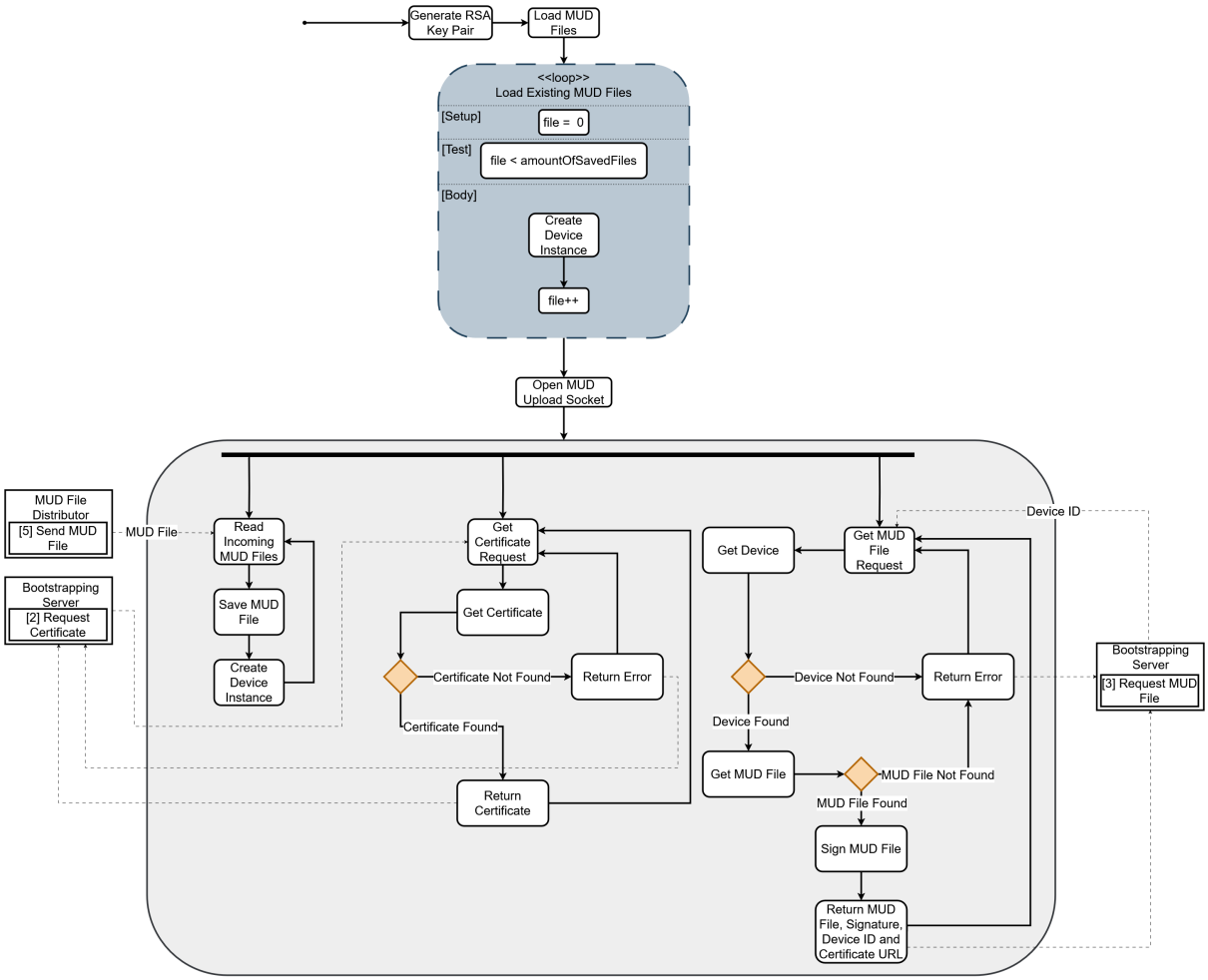


Figure 5.4: MUD Server Architecture

5.7 GPS App

The GPS data collection is done via Android app on a smartphone. The first implementation idea was to create a simple Python script that would run on QPython, an Android-based mobile Python IDE. We noticed soon that Android Python libraries cannot fine-tune the usage of the phone's GPS sensors. They are restricted to updates every couple of seconds or minutes, as the phone attempts to save power and simply overrides the Python code. The only remaining solution was to write a complete standalone Android app in Kotlin. The app can force the phone to ignore battery saving and query the GPS sensor as often as needed. The app collects its GPS data every three to five seconds and appends a timestamp. The data is then saved line by line in a .txt file on the Android phone. Later this data can be used to map GPS locations to ST-board sensor data. The issues that arose are discussed in Section 5.8.4.

5.8 Issues

A large part of the implementation phase of this thesis was identifying and solving issues. In some cases, the issues were so great that fixing them would exceed the scope of the project. Both types of problems are discussed in the following Section.

5.8.1 AWS

During the implementation of the AWS servers we noticed that AWS servers, on a free account, do not have static IP addresses that remain the same after rebooting. This is an issue, as the MUD Server shouldn't allow global access, but permanently whitelisting the Bootstrapping Servers IP is not possible. This means that the MUD Server will only have the previous IP address of the Bootstrapping Server whitelisted. Since for the purposes of this project no funds for a static IP address were allocated, this means that for every booting process of the system, either the new Bootstrapping Server IP must be whitelisted or global access must be allowed. As this project is not deployed for public or commercial use the latter option was preferred.

5.8.2 Version Incompatibilities

At the start of this project we had access to two large code bases, one from a bachelor's thesis by Mdimagh [11] and the other from STM themselves. These two code bases had to be merged. This would have produced a singular code base with socket pinging capabilities and sensor data measurement capabilities.

However, the two code bases were not compatible due to version issues. Both of them are built on top of STM libraries which went through lots of changes between when the sensor data code from the bachelor's thesis was written and the code for the sockets was written.

This meant that it was infeasible to take the functionality from one code base and insert it into the other one. Header files, functions and their parameters, and variables were either renamed, relocated or just non-existent.

We decided to drop the code from the bachelor's thesis and instead ask STM directly for a code template including basic functionality for sensors and sockets.

5.8.3 Missing Header Files

The code base we got from STM was supposed to include some basic functionality for operating the sensors on the board as well as a socket implementation just like in the previous code base. This would solve our previous problem of having two different code bases for each functionality that couldn't be merged easily. However, the first iteration of this new code was faulty. The code would not compile due to missing header files. We tried to compensate for that by painstakingly going through the errors and trying to find the fitting header files in the previous iteration of the code. Some files were findable but many were not. In addition to this there we changes in the way they controlled the sensors in their libraries. This meant we needed to ask for a working code base anew, including the missing header files. Following this, we had a code base that would compile and that had basic socket and sensory functionality. This base code was then possible to be tweaked and improved upon for the needs of this thesis.

5.8.4 GPS App

One issue during the implementation phase of the project occurred while testing the implementation state approximately halfway through. At that stage the plan was to take the board on a test journey using public transportation in Zürich. The board would first log data from an echo script. Basically, it would ping a server over the mobile network and log the signal strength and bit error rate. On the way back the board would log temperature, humidity and acceleration data to check, that all systems work in a semi-realistic environment. Additionally a way would need to be found to log the boards GPS coordinates during the test, to map the data points to a geographical location. Since this project's ST-board does not include a GPS module, the idea was to run a simple Python script on an Android phone, using the QPython mobile IDE. It quickly turned out, that these mobile IDEs do not have the capability to override Android's GPS tracking standards. This lead to the code only being allowed to log GPS data approximately every minute, which was insufficient. The quickest workaround was to create a functioning Android app using Android Studio, that would have the capabilities of logging GPS data in any way the project required. This was a relatively large unforeseen time sink.

5.8.5 Timestamp Compatibility

Another challenge arose during the implementation and testing phase. The ST-boards have no real-time clock. As a result, each ST-board internally generates a relative times-

tamp, representing the elapsed time since the device was powered on or last reset. This relative timestamp is included in the data payload sent to the Bootstrapping Server along with sensor readings. While sufficient for ordering data points locally on the board, these relative timestamps cannot be directly used for comparing or correlating data between multiple devices or against absolute time references, such as GPS logs. As the board measures sensor data in approximately three second intervals and the delay to the server is variable, these time differences must be addressed.

To fix this, a server side method reads incoming data and applies a server internal real-time timestamp. This introduces uncertainties, as network delay and the three second periods can make these times inaccurate. Luckily, inaccuracies of a few seconds have little impact on the results of our tests, or much longer real-world uses.

During testing, this alignment of timestamps was critical for the combination of sensor data and GPS logs to map geographic locations to the performance of the system.

5.8.6 Accelerometer

Another problem that arose, was the accelerometer turning off. When first testing the board infrastructure with sockets, sending real data via the mobile network to the AWS servers, the accelerometer stopped recording sensible values. It would still give an acceleration triple of (0,0,0) to the board, which would in turn send this triple to our server infrastructure. Short amounts of testing would confirm that this data could not be correct. The problem turned out to be that the acceleration sensor was disabled. We tried to enable it, trying to understand the underlying structure, going through many layers of STM boiler plate code, comparing to the other sensors. This didn't bear any fruits after a big time investment. We decided not to sink any more time into this as the accelerometer data wouldn't be too important anyways given our GPS data measurements.

Chapter 6

Evaluation

6.1 Experimental Setup

To evaluate the functionality of the system test runs were set up. The first test run was done without connection to the server structure. The idea was to check if the boards were functional and could read sensible data. Also, a first version of a GPS tracker was used, still running on a mobile Python IDE, which did not yet produce sufficiently good results, see Section 5.8.4. Additionally, we pinged a server to check if the board had mobile connectivity. The test run consisted of the board being brought onto a train in the area of Zürich, along with a phone running the GPS code. The board's collected data and the phone's collected GPS data would then be matched and analysed.

In the second test run, the board was used with full functionality. The sensor data was appended with a signal strength measurement in dBm. The board would measure and send data every 2-3 seconds. Simultaneously a phone with the Android GPS tracking app was brought. This could measure and log accurate GPS data every 4-6 seconds. This time the experiment was conducted by bringing all devices in a car in the area around Wangen in the Canton Zürich, as seen in Figure 6.1.

Figure 6.1: Route of second test run. Lighter red means stronger connection.

6.2 Results

6.2.1 First Test Run

The first test run produced mixed results. On the one hand, the board was able to reliably record environmental data, and the signal strength of the pinging was reasonable. On the other hand the GPS tracking code only produced novel data every few seconds to minutes and was heavily inaccurate, as further explained in Section 5.7. Due to the faulty GPS data no collected GPS data is shown in this chapter, as it would not add anything of note. Figure 6.2 shows the temperature and humidity data collected during test run one. The temperature had a high of 23.37°C and a low of 19.55°C, with a mean of 22.6°C. The humidity ranged between 53.97% and 68.13%, with a mean of 58.87%.

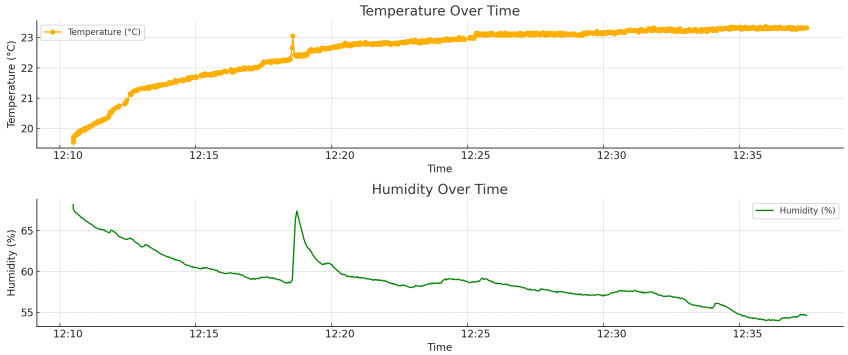


Figure 6.2: Temperature (orange) and humidity (green) data of the first test run.

The signal strength of the system is measured in dBm. This is a negative metric, meaning that values closer to 0 are better and large negative values bad. As we had multiple extreme outliers of exactly 0dBm during the two test runs, and good reason to believe -51dBm was the best possible signal strength for our system, we treated all values of 0dBm as signal losses, where the system would return a default value. Figure 6.3 shows the measured signal strength during the first test run excluding a singular signal loss of 0dBm. In this run we had a high of -51dBm and a low of -85dBm, with a mean of 56.54dBm.

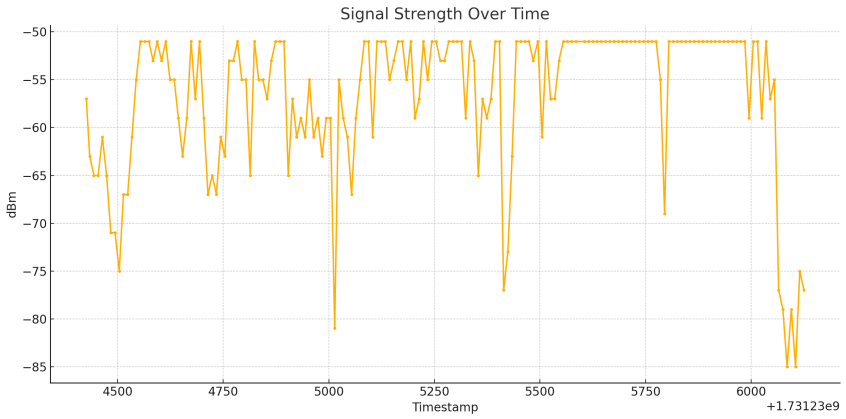
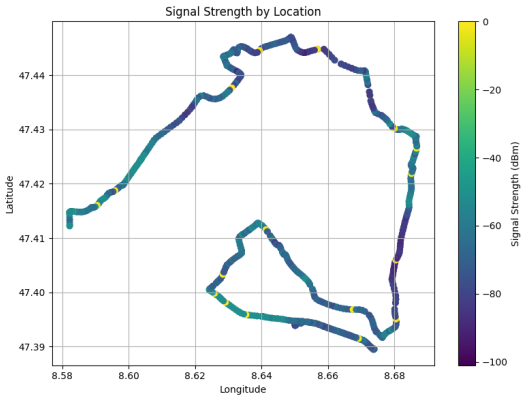


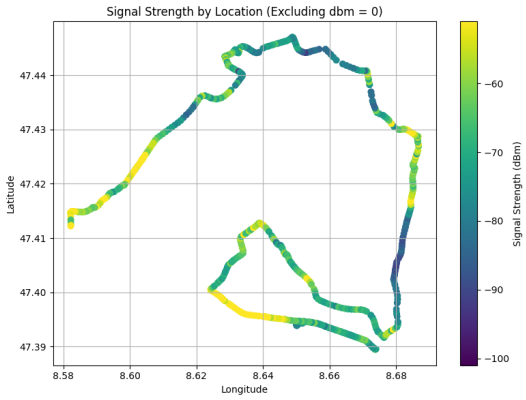
Figure 6.3: Plot showing the signal strength in dBm over time during the first test run.

6.2.2 Second Test Run

The second test run produced much cleaner results. Especially due to the GPS app. Figure 6.4 shows the signal strengths recorded during the second test run, with respect to the GPS location. Figure 6.4a shows all values recorded. These range from a high of 0dBm to a low of -101dBm, with a mean of -65.1dBm. Values closer to 0 represent better connection, while values of 0 represent a lack of connection. The 31 occurences of 0dBm, make up 3.27% of the data. Figure 6.4b shows the same values, excluding signal strength values of 0dBm. This way, the data had a high of -51dBm and a low of -101dBm, with a mean of -67.3dBm. In Figure 6.5 the same data can be seen plotted over time instead of location.



(a) Signal strength in dBm, over GPS location.



(b) Signal strength in dBm, with dBm values of 0 filtered out, over GPS location.

Figure 6.4: Plots showing signal strength in dBm at different GPS locations during the second test run.

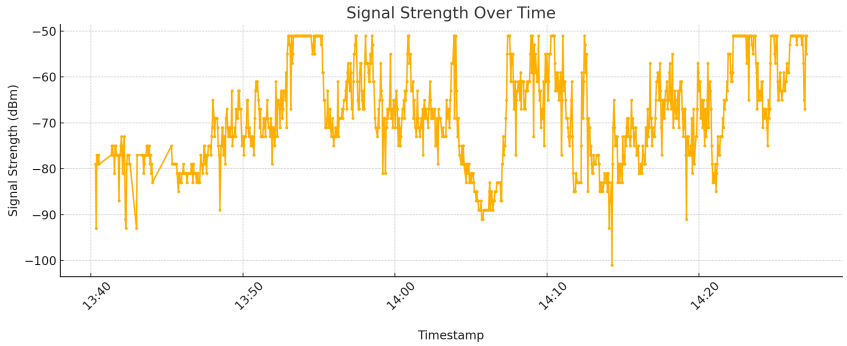
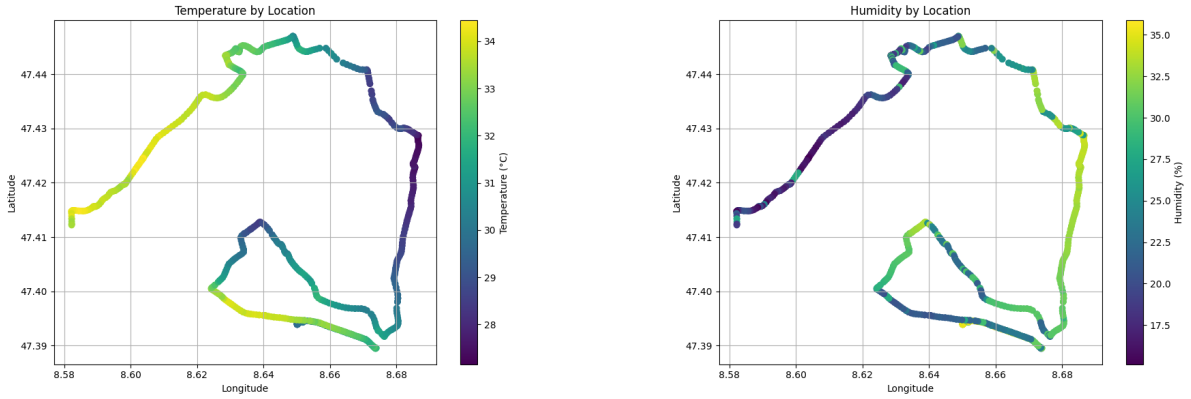


Figure 6.5: Plot showing the signal strength in dBm over time during the second test run, 0dBm filtered out.

Figure 6.6 shows the temperature and humidity data collected, dependent on GPS location. In Figure 6.6a the temperature data is displayed. During the second test run the temperature achieved a maximum of 34.44°C and a minimum of 27.17°C, with a mean of 31.28°C. In Figure 6.6b the humidity data is shown. It varied between a maximum of 35.87% and a minimum of 15.13%, with a mean of 26.21%.

Figure 6.7 shows the same humidity and temperature data on a combined plot. This time not in relation to geographical location, but time.



(a) Temperature. (b) Humidity.

Figure 6.6: Plots showing the temperature and humidity at different GPS locations during the second test run.

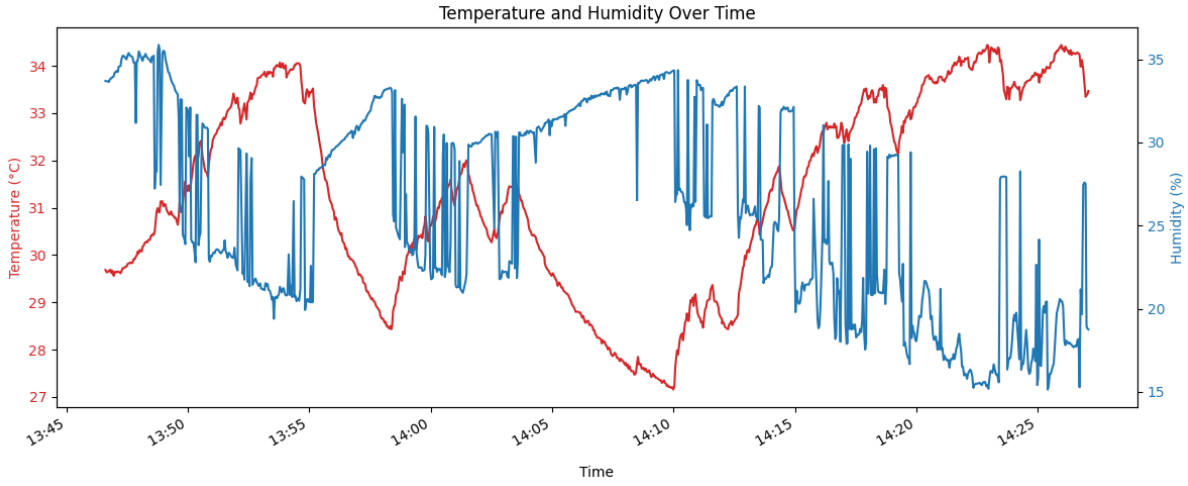


Figure 6.7: Plot showing the temperature (red) and humidity (blue) measurements over time during the second test run.

6.3 Discussion

6.3.1 Sensor Data

The collection of sensor data during this project was done to show the functionality of the system and not to analyze the data itself. A small analysis still follows to explain the varying values in the two test runs. This is to show that the data is not arbitrary.

The sensor data from test run one, as seen in Figure 6.2, seems sensible. The board was run for roughly 25 minutes in a stable and air conditioned environment inside a nearly empty public train in Zürich. The temperature is expected to slightly rise over time, as the board was always kept in relative proximity to three heat sources. A laptop and the two people conducting the test run. Generally as temperature rises, relative humidity sinks, as the air can hold more water. This can be observed in the lower half of Figure 6.2. The short spike in temperature and slightly longer spike in humidity can be attributed to one of the people conducting the test run, breathing onto the board to ensure the sensors react and do not measure arbitrary data.

The sensor data of the second test run is a little harder to decode. The second run was done in a poorly air conditioned car, explaining the higher temperatures in general. Also the test run was done on a very sunny day in May in the region of Zürich. As one can see in Figure 6.7, the data was collected in the afternoon at around 14:45 to 15:30 (rtc timestamps are one hour behind the swiss timezone). This means the sun is south-west. Looking at Figure 6.6a it is clear that temperatures tend to rise when traveling south or west, as the sunlight hits the board through the windshield, and tend to sink in the opposite direction. In Figure 6.7 it is visible that temperature and humidity roughly go in opposite direction as they are expected to. On the other hand the humidity data is also very volatile during test run two, especially when temperatures rise. Our believe is that this is not a fault in the sensors, but much more due to human perspiration at these temperatures.

6.3.2 Connectivity

It is vitally important that our system can provide reasonable signal strength and have few disconnects. According to multiple commercial sources: Teltonika [15], MCA [10] and Haven Technologies [5], for LTE a signal strength of more than -65dBm is excellent while less than -85dBm is poor.

Given this benchmark, Figure 6.3 shows that our signal was always acceptable during test run one, most of the time even excellent. With a mean signal strength of -56.54dBm and only a single loss of connection, the system seems to work very well on train lines.

During the second test run the results were still very good, with a mean signal strength of -67.3dBm and a loss of connection in 3.27% of data sent. This test showed that the system works marvelously on a train line with good mobile communication infrastructure, but has more issues when traveling through less populated areas, especially by car. This

can be seen in Figure 6.1, where the path is light red (solid connection) in populated urban areas and dark red (poor connection) in more rural regions. What can also be observed in Figure 6.4a is the scattering of connection losses. In Figure 6.4a the yellow dots are points where the signal was lost for a moment. These points seem to correlate heavily with the entry and exit points of urban areas. They are also found often within urban areas. Our interpretation is that these signal losses occur when the signal is handed between cell towers. This obviously happens more frequently within urban regions and upon entering or leaving them.

Chapter 7

Final Considerations

7.1 Summary

The aim of this thesis was to develop an IoT-based artwork tracking system that has the ability to track artworks in transit and stationary. This was successfully accomplished with the use of a STM32L4 board with humidity and temperature sensors, a GPS Android app, and a two-server AWS backend infrastructure that employs the MUD protocol.

The main goals were as follows: First, deploying an AWS Bootstrapping Server and an AWS MUD Server. Second, integrating the MUD protocol to enable secure communication between servers and IoT devices. Third, ensuring the ST-board could collect and transmit sensor data via cellular networks. Fourth, test the entire system in real-world scenarios, to ensure functionality.

7.2 Conclusions

This project enabled insights into IoT devices, cloud server infrastructure, embedded programming, as well as network security through MUD. The project posed many hurdles to overcome, such as version incompatibilities, static IPs, missing files, missing GPS sensors, faulty environment sensors and relative timestamps, all explored in Section 5.8. Despite the encountered issues, all main aims of the project could be accomplished, even though the scope changed in certain areas (e.g. development of a GPS Android app). The system proved to be functional in both test runs, as described in Section 6.3, and provided sensible sensor data with good connectivity.

7.3 Future Work

Several enhancements or changes could be made to the system in the future. For research purposes integrating a GPS module into the IoT device would be sensible. This could

also have positive implications for a commercial use, as with GPS data the location of an artwork can also be transmitted, massively improving the information usefulness and security.

A UI dashboard could be added to the server infrastructure to massively enhance usability of the system. This is not so much an issue for academic uses, but must definitely be explored for commercial uses.

Security can be improved drastically. MUD is a protocol that protects the IoT device and the server against threats, but not against the IoT device as a threat. There is no spoofing protection for example. In a real world use case there is always a tradeoff between security and light weight infrastructure, but at the least, one would have to expand upon the MUD capabilities of this thesis.

Scalability is also something that has not been tested in this thesis. There is no proof this system is scalable. At least scalability testing would need to be conducted in the future and maybe the system would need to be expanded upon to enhance scalability.

As seen in some other papers, referenced in Section 3.2, more sensors could be added. For example light, shock or air quality. These additional sensors could improve the monitoring capabilities of this system drastically.

Anomaly detection is also something that the current system does not do. In a commercial use, such a system would need a way to automatically detect anomalies both in sensor data and communication, to alert the user.

For public distribution of such a system, the HTTP protocol currently in use would need to be substituted with HTTPS to enhance security.

This thesis forms a foundation for a lightweight, secure and mobile artwork tracking system that allows the use of modern IoT technology in the art sector. But as stated in this Section, it is to be viewed as a base and needs to be expanded upon for more secure and scalable use, in real world scenarios.

Bibliography

[1] Robin Bättig and Yves Meister. *artworkTrackingMAP*. https://github.com/robaet/artworkTrackingMAP.git. 2025.

[2] Prasanth Ch.B.V. et al. "IoT Based Environment Monitoring System To Protect Heritage Artefacts". In: *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*. 2021, pp. 145–150.

[3] Angelo Feraudo et al. "Mitigating iot botnet ddos attacks through mud and ebpf based traffic filtering". In: *Proceedings of the 25th International Conference on Distributed Computing and Networking*. 2024, pp. 164–173.

[4] Ada Fort et al. "Performance Evaluation of an IoT Sensor Node for Health Monitoring of Artwork and Ancient Wooden Structures". In: *Sensors* 22.24 (2022).

[5] *Haven Technologies*. `https://thewirelesshaven.com/cellular-signal-guide/`.

[6] Zeno Heeb et al. "The Impact of Manufacturer Usage Description (MUD) on IoT Security". In: *2022 1st International Conference on 6G Networking (6GNet)*. 2022, pp. 1–4.

[7] Levente J. Klein et al. "Wireless Sensor Platform for Cultural Heritage Monitoring and Modeling System". In: *Sensors* 17.9 (2017), p. 1998.

[8] Eliot Lear, Ralph Droms, and Dan Romascanu. *Manufacturer Usage Description Specification*. RFC 8520. Mar. 2019. URL: `https://www.rfc-editor.org/info/rfc8520`.

[9] Somayya Madakam, R. Ramaswamy, and Siddharth Tripathi. "Internet of Things (IoT): A Literature Review". In: *Journal of Computer and Communications* 3.5 (2015), pp. 164–173.

[10] *MCA*. `https://lte.callmc.com/understanding-lte-signal-strength-values/`.

[11] Mohamed Islem Mdimagh. "Secure IoT Network Prototyping for Artwork Tracking". bachelor's thesis. University of Zürich, Communication Systems Group, 2024.

[12] Teshager Hailemariam Moges et al. "Cellular Internet of Things: Use cases, technologies, and future work". In: *Internet of Things* 24 (2023), p. 100910.

[13] 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)*. 2016, pp. 361–366.

[14] *STMicroelectronics*. URL: `https://www.st.com/en/microcontrollers-microprocessors/stm32-ultra-low-power-mcus.html`.

[15] *Teltonika*. https://wiki.teltonika-networks.com/view/Mobile_Signal_Strength_Recommendations.
[16] *Tera Term*. https://teratermproject.github.io/index-en.html.
[17] Carlo Trigona et al. "IoT-Based Microclimate and Vibration Monitoring of a Painted Canvas on a Wooden Support in the Monastero of Santa Caterina (Palermo, Italy)". In: *Sensors* 22.14 (2022), p. 5097.
[18] Jingwen Zhang et al. "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

AWS	Amazon Web Services
BLE	Bluetooth Low Energy
CIoT	Cellular Internet of Things
DDoS	Distributed Denial of Service
DoS	Denial of Service
DNS	Domain Name System
EC2	Elastic Compute Cloud
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated Development Environment
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
LTE	Long Term Evolution
MUD	Manufacturer Usage Description
QoS	Quality of Service
RFC	Request for Comments
RTOS	Real-Time Operating System
STM	STMicroelectronics
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
3GPP	Third Generation Partnership Project

List of Figures

List of Tables