System for Automatic Detection of Availability for Secretary of Department of Informatics Engineering

Capstone Project Report

André Costa
Fábio Sá
Lourenço Gonçalves
Marcos Ferreira

Bachelor of Informatics and Computing Engineering

U.Porto Tutor: João Pedro Dias
Proponent: João Cardoso

July 17, 2023

Contents

1 Introduction

This report aims to provide an overview of our project that involved an implementation of a system for automatic detection of the presence of individuals in an office setting by sonar sensors.

1.1 Background

The project was carried out within the organizational environment of the SEC DEI, the Department of Informatics Engineering of the Faculty of Engineering of University of Porto (FEUP).

The SEC DEI plays a crucial role in facilitating computer engineering education and research within the institution. However, one notable issue that arose was the lack of visibility from the exterior of the office to determine if they were able to attend students, staff and visitors questions. This limitation created a problem bringing lack of communication, productivity and valuable time wasted.

Figure 1: SEC DEI

1.2 Objectives and expected results

Recognizing the importance of optimizing workspace utilization, the project aimed to develop a solution that would enable individuals to quickly determine the availability of any attendant before entering the office area.

Figure 2: Service desk

The primary objectives of this project were to implement a connection between the sensors and a micro-controller, so we could work with the distances measured and to develop a server that would accurately update the occupancy state of each table.

A primary goal was to provide real-time visibility of table availability, which could be conveniently accessed outside the office area through a web page displayed on a television screen.

1.3 Report structure

This report is structured into four sections. Section 1 provides an introduction to the project, including the background, objectives, and expected results. Section 2 describes the methodology and main activities carried out during the project, including the methodology used, stakeholders involved, and activities developed. Section 3 presents the development of the solution, including requirements, architecture, and technologies used, as well as validation of the developed solution. Finally, Section 4 concludes the report with a summary of achieved results, lessons learned, and suggestions for future work.

2 Methodology and Development Process

This section provides an overview of the methodology used during the project, including the iterative development approach, tools utilized for communication and collaboration, stakeholder roles and responsibilities, and the activities conducted throughout the project timeline.

2.1 Methodology used

We adopted an iterative development approach, employing weekly sprints and progress meetings. This methodology enabled us to receive continuous feedback and make necessary adjustments to our code implementation as the project evolved.

In addition to these meetings, we use tools like GitHub for effective team communication, efficient code organization, task allocation, and for our tutor to assess and provide comments on our progress. GitHub facilitated seamless collaboration and version control throughout the project.

For communication within the team, we utilized Microsoft Teams for our weekly meetings. We found this tool to be propitious to productive team interactions and decision-making. Furthermore, we used Discord served as our preferred platform for code development sessions.

Additionally, we scheduled regular appointments with our clients, the staff from SEC DEI. These meetings provided valuable opportunities to gain insights into their expectations and preferences, as well as to showcase intermediate prototypes for their feedback and validation.

2.2 Stakeholders, roles, and responsibilities

This project involved multiple participants, each with distinct responsibilities and roles:

- Project Coordinator:
  - Professor and Director of Department of Computer Engineering João Cardoso: Responsible for overseeing the project, providing guidance and feedback, ensuring the project aligns with the objectives of the SEC DEI and Also the equipment necessary for the development of the project.
- Project Tutor and Technical Supervisor:
  - Visiting Assistant Professor João Pedro Dias: Acted as the technical advisor and mentor, providing expertise and guidance related to the implementation and technical aspects of the project.

- Clients:
  - SEC DEI: The clients were represented by members of the SEC DEI department. They provided requirements, insights, and feedback throughout the project.
- Project Team (4 members):
  - André Correia da Costa (up201905916)
  - Fábio Araújo de Sá (up202007658)
  - Lourenço Alexandre Correia Gonçalves (up202004816)
  - Marcos William Ferreira Pinto (up201800177)
  - All team members were actively involved in the project and all contributed to the overall progress and success of the project, collaborating on various tasks and ensuring effective communication within the team.

2.3 Activities developed

Throughout the project timeline, several activities were carried out, including project planning, code development, team gatherings, prototypes presentations, clients meetings and testing scenarios. Detailed information about these activities, including their timeline, can be found in the Gantt chart provided in the annexes at the end of the report (figure 24).

1. Project Planning and Research:
   - Conducted initial research on proximity sensor technology and workspace utilization optimization.
   - Defined project scope, objectives, and requirements.
2. Sensor and M5 communication:
   - Implemented the connection between the sensors and the micro-controller and each configuration.
3. Server Development:
   - Designed the server architecture to process sensor data and update table occupancy states.
4. Web Interface Design:
   - Designed the user interface for displaying table availability.
5. Testing and Validation:
   - Conducted extensive testing of prototypes to ensure accurate sensor measurements and server functionality.
6. Prototypes Presentations:

- Demonstrated project progress, showcased intermediate prototypes, and gathered feedback for improvement with SEC DEI and our Coordinator.

7. Client Meetings:

   - Discussed requirements, obtained feedback on deliverables, and addressed any concerns or questions.

3 Solution Development

The solution development phase focused on addressing the requirements identified for the project. This involved the implementation of both hardware and software components to achieve real-time occupancy monitoring and table state updates. Additionally, a user-friendly web interface was developed to provide easy access to the current table availability. The solution also included data storage capabilities to store relevant information about each table. Throughout the development process, considerations were made to ensure reliability, usability, scalability, and low power consumption of the system.

3.1 Requirements

Through the meetings and discussions with the SEC DEI team, we gained significant insights which lead us to define the following set of project requirements and restrictions. These specifications and constraints are critical to the effective development and high accuracy of our solution.

- Functional requirements:
  - Real-time Occupancy Monitoring: The solution shall provide real-time monitoring of table occupancy status.
  - Table State Updates: The system shall update the state of each table based on occupancy changes.
  - Web Interface: A user-friendly web interface shall display the current table availability to users.
  - Data Storage: The system shall store each table information.
- Non-functional requirements:
  - Reliability: The system shall be reliable in detecting the presence of a person.
  - Usability: The web interface shall be intuitive and easy for users to quickly check table availability.
  - Scalability: The solution shall be scalable to accommodate an increasing number of tables and users.
  - The system shall have low power consumption.
- Constraints:
  - The system assumes that a person is in attendance within a one-meter distance from the table.

3.2 Architecture and technologies

This section delves into the architectural design and the various technologies chosen to implement our solution. It provides insights into the components, frameworks, and software used, considering the project's requirements and constraints.

3.2.1 Hardware components

To achieve our goal, we utilized a combination of hardware and software components (figure 25). The hardware components included the HC-SR05 ultrasonic proximity sensor (figure 3 and the M5Stack ATOM-Lite micro-controller (figure 4a) .

The HC-SR04 ultrasonic proximity sensor played a crucial role in detecting the distance of the closest object. By measuring the distance, we were able to determine the occupancy status of each table in real-time. The data provided by the sensor is not very accurate, but with a combination of techniques used we made it work.[1]

The transmitter emits ultrasonic sound waves, which are inaudible to humans, at a specific frequency. These sound waves travel through the air and bounce off objects in their path. The receiver of the sensor then detects the reflected waves and calculates the distance by measuring the time it took for the wave to travel from the sensor to the object and back.[1]

Figure 3: Ultrasonic sensor used in the project

The M5Stack ATOM-Lite micro-controller acts as the central processing unit of our system. It receives the distance data from the proximity sensor and performs calculations to predict the presence of a person being attended. It is also responsible for sending the data to the software components.[2]

Figure 4: M5 Atom Stack micro-controller

Figure 5 shows the diagram for the connections between micro-controller and a single sensor.

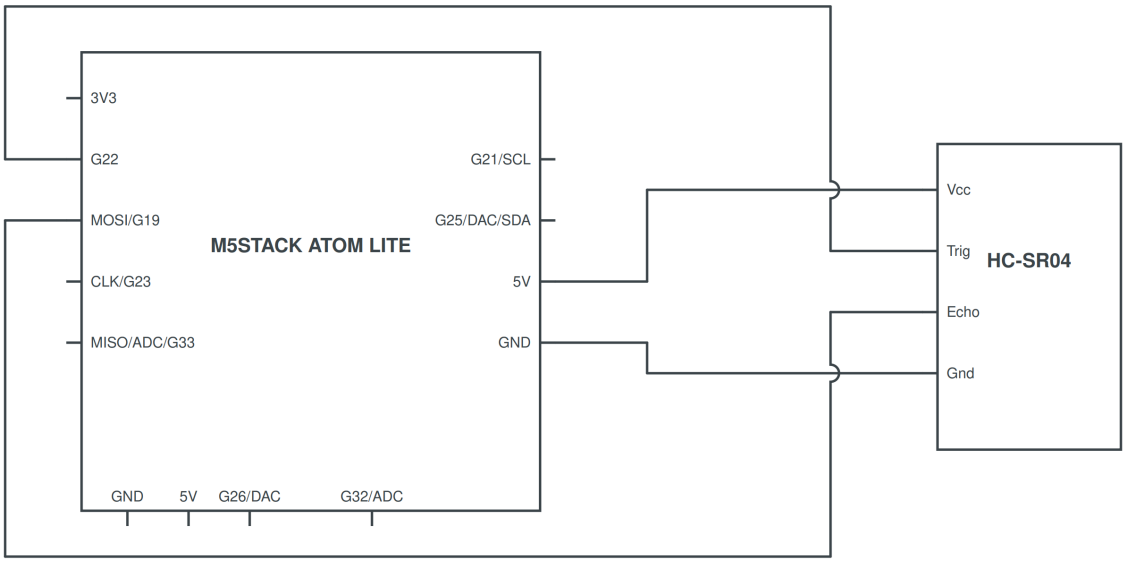

Figure 5: M5 and Sensor connections diagram

Figure 6 shows the connections between the proximity sensor and the micro-controller.

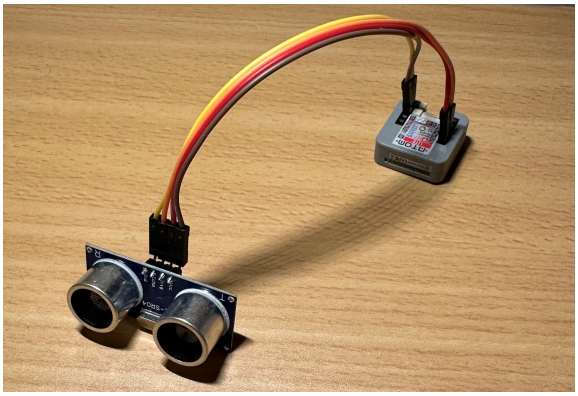

Figure 6: M5 and Sensor connections

These connections can be described as:

1. Vcc - 5V: This connection represents the power supply for the sensor, providing a 5V voltage level to ensure proper operation.
2. Trig - G22: This connection establishes the trigger line. The micro-controller sends a signal through this line to initiate the distance measurement.
3. Echo - G19: This connection represents the echo line. After sending the trigger signal, the sensor responds with an echo signal. The time taken for the echo signal to return helps calculate the distance to the object.
4. Gnd - G: This connection serves as the common ground reference, ensuring a stable and reliable electrical connection.[2]

3.2.2 Software components

In terms of software components, we employed several tools and technologies to handle data storage, communication, and visualization. The core software components used were InfluxDBv2[3], Mosquitto MQTT broker[4], Telegraf[5], and a NodeJS server[6].

- InfluxDBv2: Popular time-series database system designed to handle and store time-stamped data efficiently, making it well-suited for storing and retrieving data that changes over time.[3]
- Mosquitto MQTT Broker: Mosquitto is an open-source message broker that implements the MQTT protocol. MQTT is a lightweight publish-subscribe messaging protocol commonly used in IoT (Internet of Things) applications. The MQTT broker facilitates communication between devices by enabling them to publish messages to specific topics and subscribe to topics of interest.[4]
- Telegraf: Telegraf is an open-source data collection agent that gathers and transforms data from various sources. Telegraf can parse and manipulate data before forwarding it to other systems or storage solutions, making it a flexible tool for data integration and processing.[5]

- NodeJS Server: NodeJS is a popular JavaScript runtime built on Chrome's V8 engine. It allows developers to run JavaScript code on the server-side, enabling the creation of scalable and high-performance web applications. A NodeJS server can handle incoming requests, process data, and generate dynamic responses. It is commonly used for building APIs, web servers, and real-time applications.[6]

InfluxDBv2 serves as the database system responsible for storing the data received from the micro-controller. It also enables real-time data analysis. By leveraging the capabilities of InfluxDBv2, we are able to analyze and interpret the data sent from the micro-controller, which includes the distances measured by the sensors and the corresponding occupancy state.

The Mosquitto MQTT broker plays a crucial role in ensuring that the data collected by the micro-controller is transmitted to the desired destinations.

We use Telegraf to specify the data to be transferred and determine how it should be parsed and sent to InfluxDBv2. Telegraf serves as a versatile tool for data collection and transformation. It allows us to define the specific data points of interest and configure the desired format before forwarding it to the others software components.

Finally, a NodeJS server is responsible for displaying the state of each table on a screen. It receives the occupancy data from InfluxDBv2 and presents it in a user-friendly format web interface. The server allows users to quickly and easily determine which tables are free and which are not.

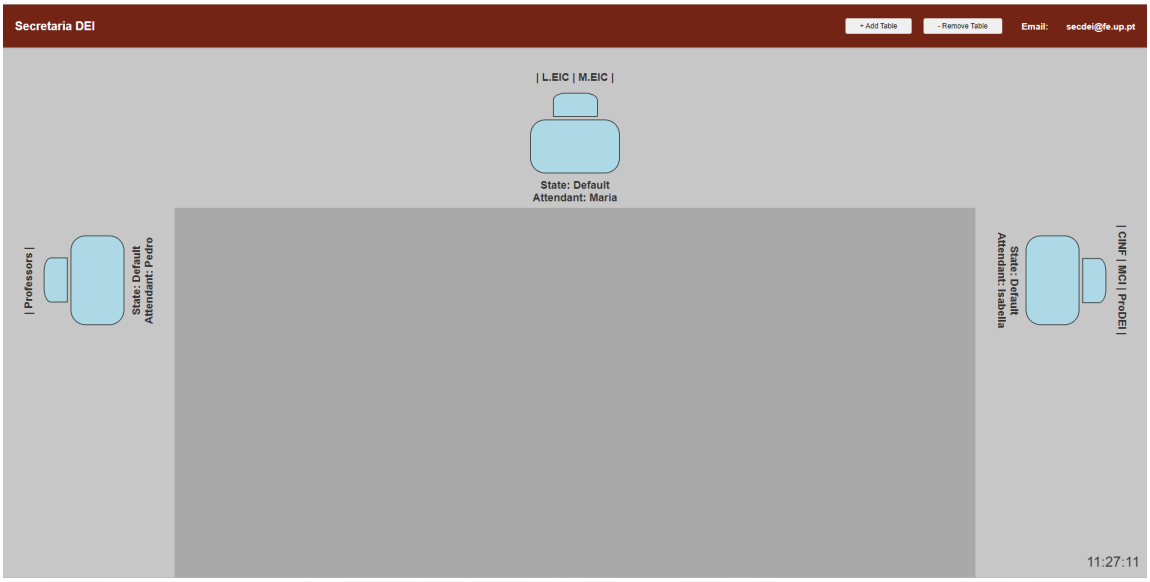

Figure 7: Web Interface

All of these software components, including InfluxDB, Mosquitto, Telegraf and the NodeJS server, were deployed on docker containers running on a Raspberry Pi 400[7]. Docker containers play a crucial role in running and managing the software components by providing a lightweight and isolated environment, ensuring that each component has its dependencies and resources encapsulated.

The Raspberry Pi 400 is a single-board computer that holds everything together. Its compact size, low power consumption, and ample computing power made it an ideal choice for our project. Its versatility and affordability made it a cost-effective and accessible option for running our system.

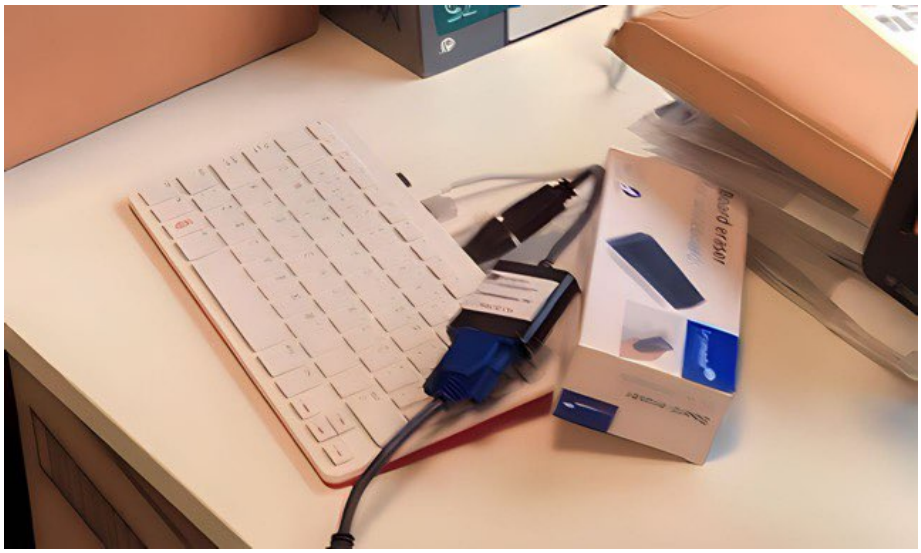

Figure 8: Raspberry Pi

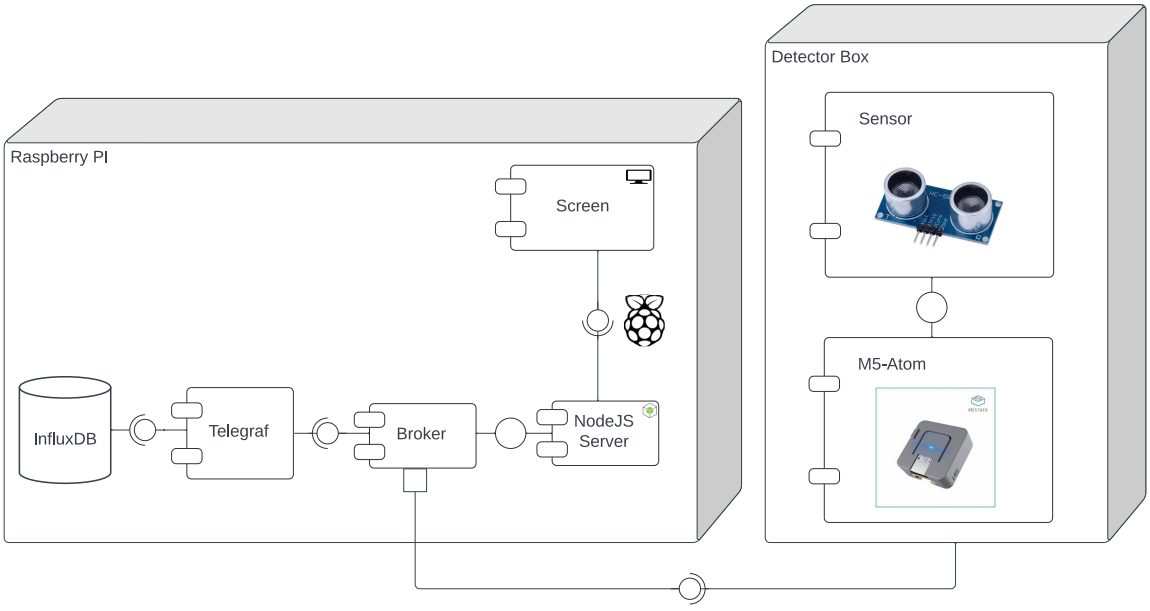

Figure 9: Project Architecture

3.3 Developed solution

3.3.1 2-sensor model

The M5 micro-controller code implementation involved establishing the connection with the sensor and defining the range of sensor measurements. Based on a similar implementation conducted in 2022 within a group of FEUP's students[8]. We determined that the valid range for measurements was between 5 cm and 250 cm, -1 if otherwise, and within that range, a person's attendance was considered between 5 cm and 80 cm. This required extensive testing to ensure accurate results. The occupancy status would only change after 5 consecutive readings for the same calculated status (0 or 1, respectively, available or unavailable).

Additionally, as part of the solution development, we incorporated a feature to handle the situation when the attendant is away from the table. To indicate this status, we added a physical button that the attendant could press to signal their absence. This button was external to the sensor system and allowed for manual control of the occupancy status.

Due to the imprecise nature of the sensor measurements already mentioned, we conducted several test cases that led to the development of two prototypes: the 2-sensor model and the 3-sensor model. For the two distinct models we needed to send to the software components the information in different formats. The data sent was the distance measured be each sensor, the calculated state of occupancy and the MAC address of the M5 micro-controller.

The 2-sensor model consisted of one sensor positioned in the middle of the table, pointing towards the chair direction but with a higher elevation to avoid interference from the chair (see Fig. 10a), and another sensor mounted on the desk's wall (see Fig. 10b), following the same criteria. Each sensor was connected to a M5 micro-controller.

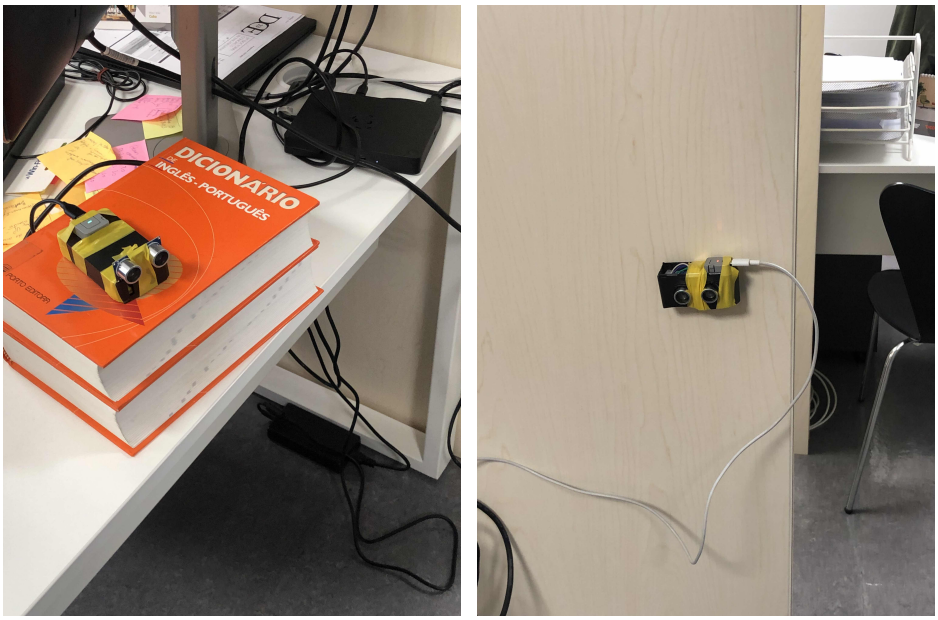

Figure 10: 1st sensor (left) and 2nd sensor (right)

3.3.2 3-sensor model

The 3-sensor model involved a box enclosure containing three sensors positioned to cover the zone where the attendee would typically be located during attendance. One sensor was directed towards the chairs zone, while the other two sensors had different radius applied. We have both pictures and model designs to demonstrate this configuration. In the setup, each proximity sensor was connected to its own micro-controller for power supply, while a main micro-controller housed the code responsible for managing the connections (see Fig. 11).

Figure 11: Configuration using three sensors per micro-controller

Through testing, we found that the 3-sensor model provided significantly more precise results compared to the 2-sensor model. Here is the schematic for it:

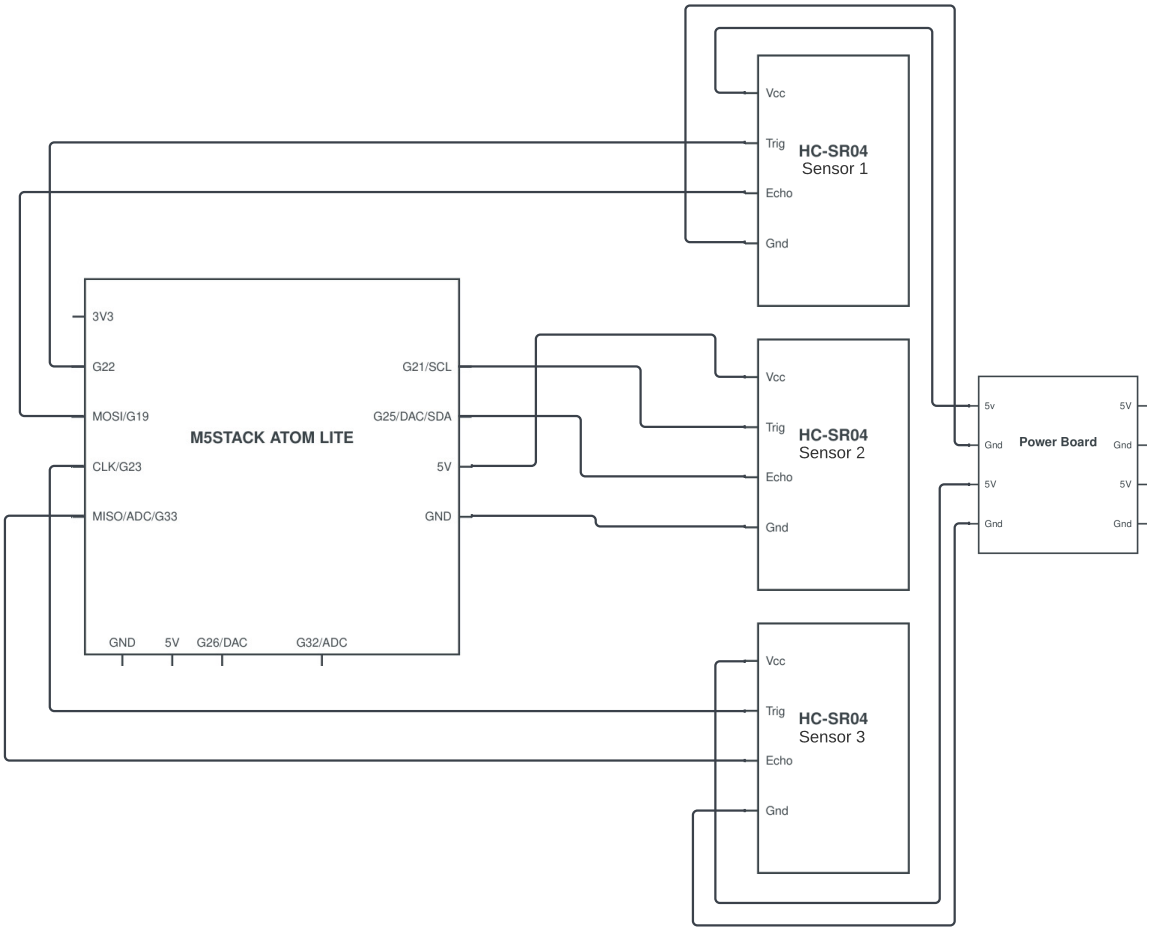

Figure 12: Schematics for a 3 sensors model

To connect the 3 sensors to the same micro-controller we use the same 5v and ground pins (using an adapter or an external power board). Furthermore the connection of the data pins are as follows:

Sensor	**Trig**	**Echo**
First Sensor	22	19
Second Sensor	23	33
Third Sensor	21	25

Table 1: Connections of the sensors to the micro-controller

The following figure shows the layout of the sensors:

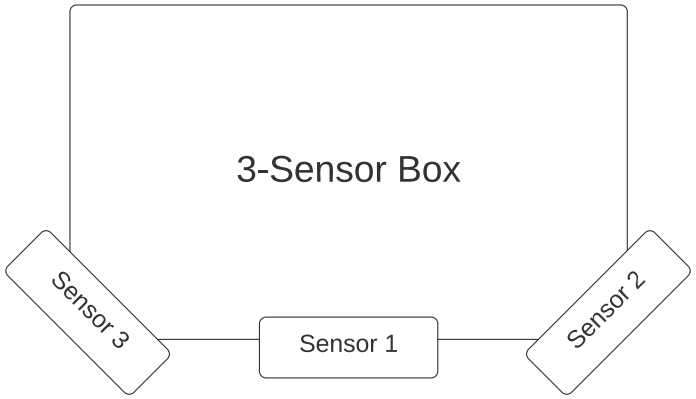

Figure 13: Organization of the sensors in the box

3.3.3 Data storage

To evaluate our prototypes, we needed to collect data related to the distances measured by each sensor, as well as the state of the system. For that end, we use a Mosquitto MQTT broker to communicate between the micro-controller and the InfluxDB database.

To facilitate the ingestion of data from MQTT topics into InfluxDB, we utilize Telegraf. By configuring Telegraf, we were able to parse the MQTT data received from the micro-controllers and write it to InfluxDB for storage. The MQTT topics were parsed using the input plugin `inputs.mqtt_consumer.topic_parsing` with the following configuration:

```
[[inputs.mqtt_consumer.topic_parsing]]
topic = "sadasdei/+/+"
tags = "_/_/sensor_id"
measurement = "_/state/_"
```

3.3.4 Web interface

The NodeJS Server runs locally on the Raspberry Pi. It was implemented using Express and facilitated the management of table information. This included storing the MAC address of the M5 micro-controller, table ID, attendant name, and assigned courses. The server offered simple methods such as adding a table, removing a table, and retrieving the table set for display. It was connected to MQTT and would update the table information upon receiving a message.

The user web interface was designed to be adaptable as per the requirements. Tables could be added, as depicted in Figure 14. Each table on the interface displayed the assigned education degrees, attendant information, and occupancy status. The web interface was divided into two sections: one for administrators, featuring buttons to add and delete tables, and another for display purposes only.

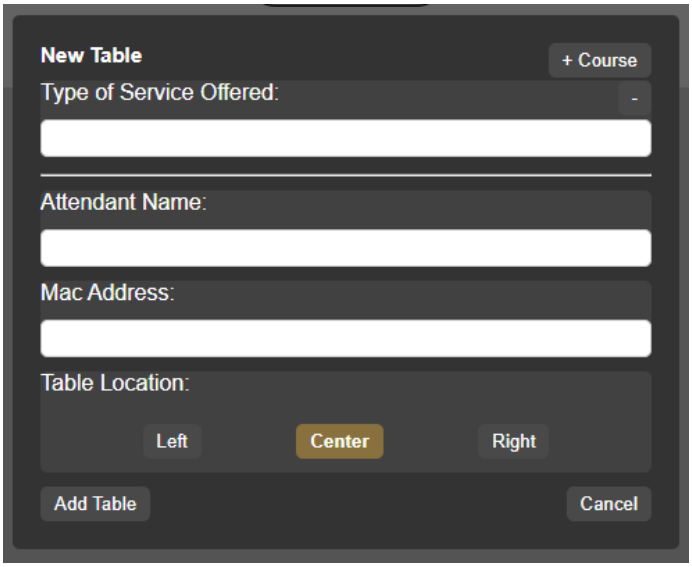

Figure 14: Form for adding a table in workspace

3.3.5 Sequence diagram

With all the components together, the data always flows in one direction. The following sequence diagram shows the messages exchanged between the components after 4 readings from the sensor indicate a presence:

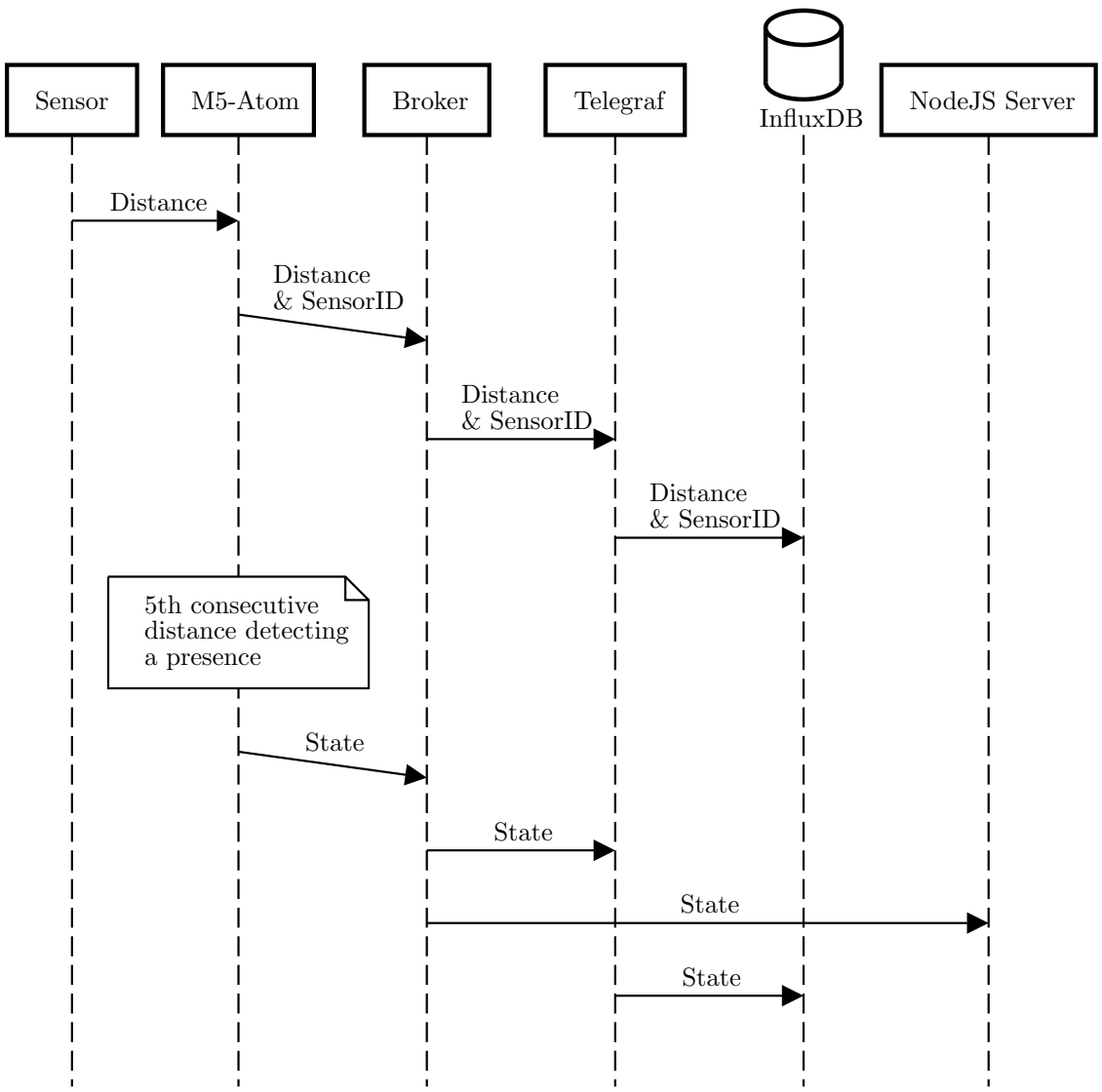

Figure 15: Sequence diagram for the flow of the system

3.4 Validation

To validate the developed solution and assess its compliance with the identified requirements and constraints, we conducted several actions:

1. Test Cases: We performed various test cases using real attendances within the SEC DEI environment. This allowed us to analyze different scenarios and evaluate the accuracy and

performance of the solution. We also did some attendance simulations, testing both normal and more extreme positions of the attendee.

2. Data Analysis: We utilized InfluxDBv2 to store the collected data during the testing period. By analyzing the data, we were able to assess the effectiveness of the solution in accurately detecting attendances and updating the table states accordingly.

3. Algorithm Development: To further evaluate the performance of the solution, we implemented algorithms in Python, leveraging libraries such as Pandas for data manipulation. These algorithms allowed us to analyze the data and generate visual representations, such as graphs, to assess the accuracy and consistency of the solution.

For the 2-sensor model, we encountered challenges and inconsistencies in the results. The time difference between real-time and the Raspberry Pi's clock affected the assessment process. The accuracy of the sensor placed on the desk's wall was very low, particularly for standing attendances. Consequently, the model exhibited poor accuracy in detecting attendances and updating the table states accordingly. Due to these limitations and after fixing the bug with the Raspberry Pi's clock time, we proceeded to the 3-sensor model, which yielded more promising results.

The 3-sensor model demonstrated improved accuracy and consistency in detecting attendances, resulting in an accuracy rate of 88.9%. This discrepancy can be attributed to a constraint inherent in the model, namely, the limitation posed by the distance between the sensor and the attendee. To quantify the accuracy of each sensor, we measured the percentage of times the sensor's distance fell below 80 cm when the table was occupied or unavailable.

Sensor	**Accuracy**
1	54.00%
2	4.00%
3	52.00%

Table 2: Accuracy for each Sensor

The sensor layout is detailed in Fig. 13.

These results demonstrate that the sensor 2 has a relatively lower relevance compared to the other sensors in the 3-sensor model. While its impact on the results may not be as pronounced as the other sensors, its still significant and should not be overlooked. The following charts showcase the results obtained from the 3-sensor model, providing visual evidence of the solution's improved accuracy and usability.

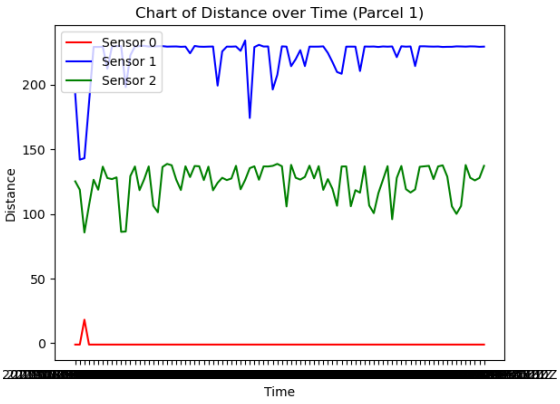

Figure 16: Distances 1st Parcel

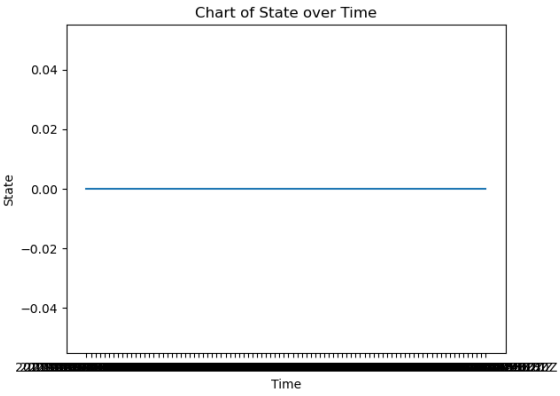

Figure 17: State 1st Parcel

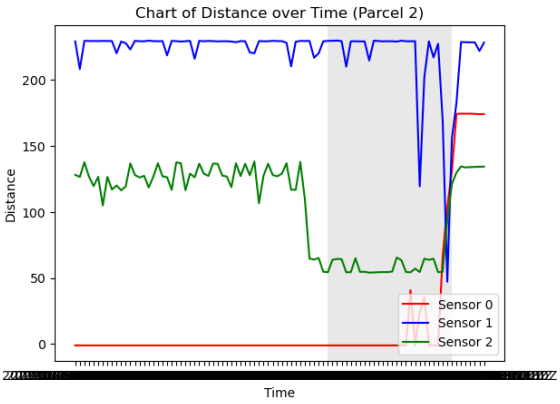

Figure 18: Distances 2nd Parcel

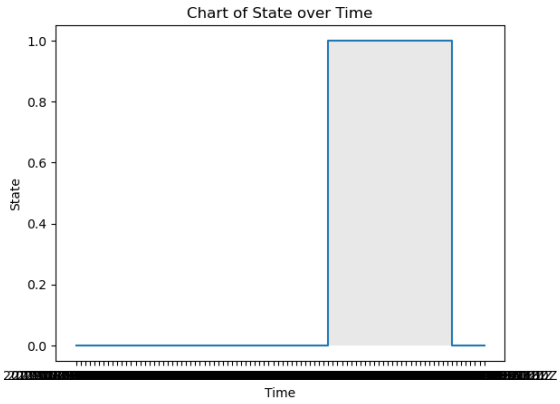

Figure 19: State 2nd Parcel

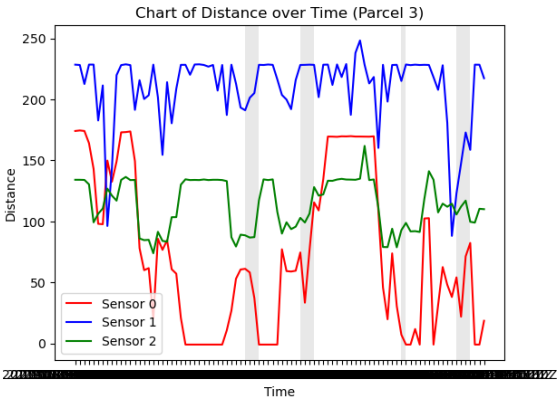

Figure 20: Distances 3rd Parcel

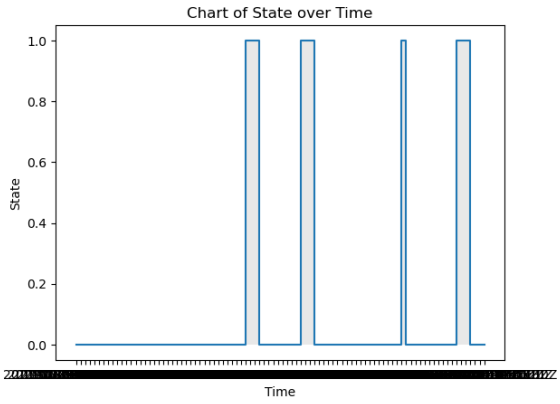

Figure 21: State 3rd Parcel

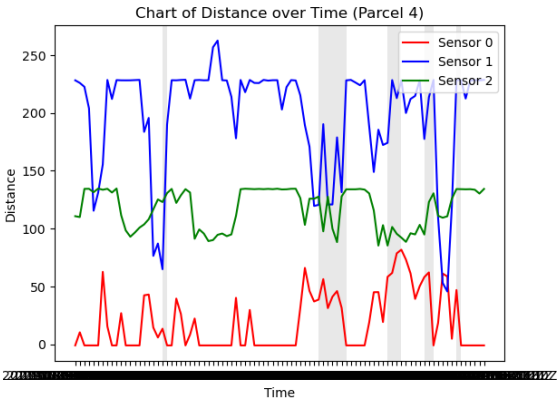

Figure 22: Distances 4th Parcel

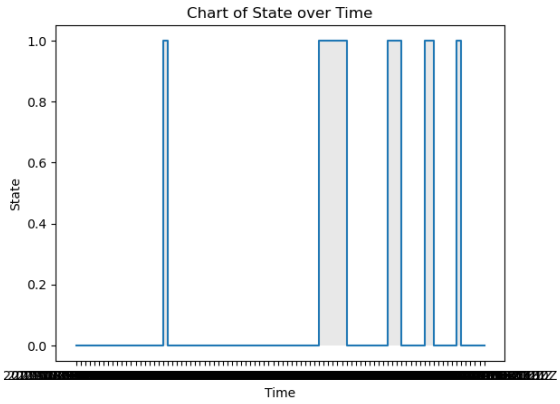

Figure 23: State 4th Parcel

Overall, the validation process involved conducting extensive tests, analyzing data, and comparing the results against the identified requirements and constraints. The positive outcomes from the 3-sensor model demonstrated the effectiveness and reliability of the solution in accurately detecting attendances and providing real-time updates on table occupancy status.

4 Conclusions

In conclusion, the results of this project have demonstrated the successful development and implementation of a table occupancy detection solution. Through the utilization of the 3-sensor model and the integration of various hardware and software components, we have achieved reliable and accurate detection of table occupancy.

4.1 Achieved results

The validation process, which included diverse test cases and analysis of the charts provided by InfluxDBv2, has confirmed the effectiveness of the solution. It is important to notice that the results obtained from the 2-sensor model were inconclusive due to the significant challenges encountered during the testing phase. The data collected from the 2-sensor model did not provide reliable measurements, making it difficult to assess its accuracy and comparing it to the 3-sensor model. This highlights the importance of sensor configuration and placement in achieving accurate and reliable results. Despite the limitations with the 2-sensor model, the implementation of the 3-sensor model showcased promising results and great accuracy in detecting attendances.

The implemented solution has provided real-time information about table availability, allowing for optimized workspace utilization. This contributes to a more efficient and productive work environment. The system's ability to update the state of each table based on occupancy has addressed the challenge of visibility and enabled individuals to quickly determine table availability before entering the office area.

Overall, the results of this project align with the objectives set out at the beginning. While the deployment phase could not be fully realized due to time constraints and the lack of available products, the conducted tests and evaluations have demonstrated the successful achievement of the project's main objective.

4.2 Distribution of Work

Throughout the project, we successfully navigated different sections, each playing a crucial role in the development of our product. The work was evenly distributed among team members, ensuring a collaborative and balanced effort.

In the sections of Project Planning and Research, Client Meetings, Prototypes Presentations, and Testing and Validation, everyone actively participated, contributing their insights and expertise. These sections required collaborative discussions, coordination, and thorough evaluation, and each team member played an integral role.

On the other hand, sections such as Sensor and M5 communication, Server Development, and Web Interface Design involved more distributed work. For these sections, we divided the tasks into pairs, allowing for efficient collaboration and feedback loops. Each pair worked on their designated areas, ensuring progress and accountability.

By working together and leveraging each team member's strengths and expertise, we were able to tackle different aspects of the project simultaneously. This collaborative approach fostered skill development, mutual support, and a well-rounded solution.

Overall, the combination of group effort ensured a balanced and effective approach to the project. It fostered collaboration, enabled efficient progress, and facilitated continuous feedback, resulting in the successful development and completion of our product.

4.3 Lessons learned

Throughout the course of this project, we accomplished various objectives that taught us valuable lessons that will be useful in future projects and can be used to a variety of scenarios

1. Planning and Requirement Analysis: We recognized the need of detailed planning and requirement analysis. Taking the time to analyze the layout of SECDEI allowed us to identify

many possible problems early on.

2. Testing and Validation: Thorough the tests done on SECDEI, with both simulated and real service scenarios, our notion of the need for testing was reinforced.

3. Brokers and Database Management: Our expertise with technologies such as Mosquitto MQTT and InfluxDBv2 gave us with significant insights into message brokers and database management. We received hands-on experience configuring and deploying MQTT brokers and how to use InfluxDBv2 to store and retrieve data with Telegraf.

4. Working with micro-controllers and hardware components: This was a crucial part of our project. We dived into micro-controller programming, strengthening our skills in developing code for embedded systems. This included learning pin configurations, communication protocols, and troubleshooting hardware-related difficulties. Eventually, we were able to connect three sensors to a single micro-controller.

Reflecting on these lessons has provided us with significant insights that will help in future projects. Our experiences with brokers, InfluxDB, micro-controller development, and hardware integration have widened our technical skills.

4.4 Future work

Ideas for improvements and future work:

- Adjustable Sensor Boxes: Introduce 3D printed sensor boxes that allow for changing the angle of the sensors. These boxes can be mounted on walls or placed on desks, providing flexibility in sensor placement.

- External Display: Enable the SEC DEI external display. This would allow users to view the status or information from a distance, improving visibility and accessibility.

- Manual Mode: Implement a mode without sensors, where SEC DEI staff can manually change their status by user interface. This mode would provide an alternative for situations where sensor-based tracking is not feasible or desired.

- Predicting Service Time: Develop a feature that can estimate the duration of each interaction or service provided. This prediction could help in managing resources, scheduling appointments, and improving overall efficiency.

- Queue Management System: Introduce a system that assigns and manages digital queues or tickets for individuals waiting for service. This system would streamline the process, reduce waiting times, and enhance users satisfaction.

By incorporating these ideas into future iterations of the project, it is possible to enhance functionality, user experience, and overall effectiveness. These improvements can contribute to creating a more versatile, efficient, and user-friendly system.

5 Annexes

Figure 24: Gantt chart for work distribution

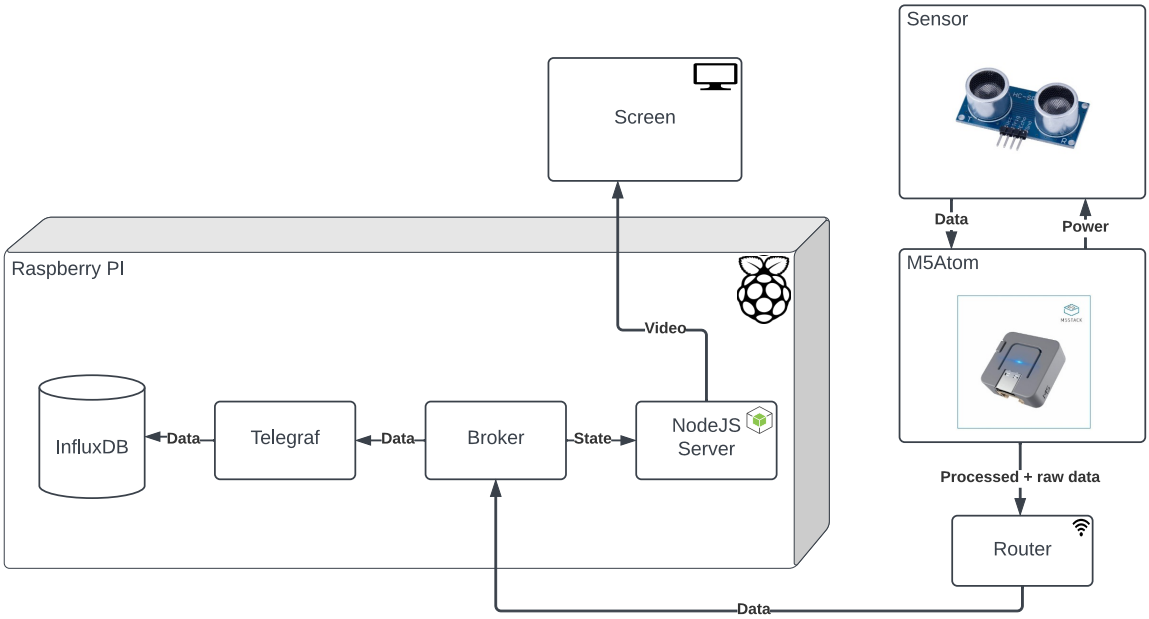

Figure 25: More detailed architecture diagram

References

[1] Alessandro Massayuki Nakatani, Anderson Valenga Guimarães, and Vicente Machado Neto. MediÇÃo com sensor ultrassÔnico hc-sr04. In *International Congress on Mechanical Metrology*, volume 3, Gramado, 2014. https://www.devopsschool.com/blog/wp-content/uploads/2022/09/influxdb_2017.pdf.

[2] M5Stack. Atom lite esp32 iot development kit — m5stack-store. https://shop.m5stack.com/products/atom-lite-esp32-development-kit.

[3] Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Dr. Esteban Zimanyi. Time series databases and influxdb, 2017. https://www.devopsschool.com/blog/wp-content/uploads/2022/09/influxdb_2017.pdf.

[4] Ana C. Franco da Silva, Uwe Breitenbücher, Kálmán Képes, Oliver Kopp, and Frank Leymann. Opentosca for iot: Automating the deployment of iot applications based on the mosquitto message broker. In *Proceedings of the 6th International Conference on the Internet of Things*, IoT'16, page 181–182, New York, NY, USA, 2016. Association for Computing Machinery. https://doi.org/10.1145/2991561.2998464.

[5] Nicolas Chan. A resource utilization analytics platform using grafana and telegraf for the savio supercluster. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*, PEARC '19, New York, NY, USA, 2019. Association for Computing Machinery. https://doi.org/10.1145/3332186.3333053.

[6] Lakshmi Prasanna Chitra and Ravikanth Satapathy. Performance comparison and evaluation of node js and traditional web server (iis). In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, pages 1–4. IEEE, 2017.

[7] Jolle W Jolles. Broad-scale applications of the raspberry pi: A review and guide for biologists. *Methods in Ecology and Evolution*, 12(9):1562–1579, 2021. https://besjournals.onlinelibrary.wiley.com/doi/pdfdirect/10.1111/2041-210X.13652.

[8] Henrique Pinho, Diogo Vale, Pedro Santos, and Tiago Gonçalves. Dei-interface. https://github.com/daniel-s-goncalves/DEI-Interface.