

Title of the project:	"Digital twin scope lab monitoring"
Team Members:	Saida Kakimova, Asset Kapi, Miras Mukayev, Nursultan Bulekov, Sherniyaz Zhakenov
Project Advisor/Co-Advisors	Dimitrios Zormpas
Executive Summary	
Modern campuses and laboratories rely on diverse sensors to monitor environmental and equipment conditions. However, data from these sensors is often fragmented across multiple systems, making it difficult to visualize, compare, and analyze in real time. By creating a digital twin-based laboratory monitoring system that combines sensing, communication, data storage, visualization, and people counting into a single computing-based solution, this research tackles that issue. The primary objective is to design, implement, and validate a testbed that can measure and compare the performance of various sensors and laboratory devices (PC, Raspberry Pi) in actual laboratory settings. The goals include: • Integrating different types of environmental sensors with low-power microcontrollers. • Ensuring wireless data transmission via Wi-Fi (MQTT). • Developing a pipeline for real-time data storage, retrieval, and visualization. • Building a scalable digital-twin environment enabling distributed sensing. • Supporting additional intelligent monitoring functionality through machine learning-based people counting The implemented system uses microcontrollers (ESP32) with multiple sensors (temperature, humidity, sound, motion, and distance) to collect real-time data from the physical environment. The collected data is transmitted wirelessly to a centralized server using MQTT, stored in InfluxDB, and visualized using Grafana dashboards in the mobile application. Also, the system includes a people-counting machine learning that displays the number of people currently detected in the camera's field (Raspberry Pi) of view located in a designated area. The project is intended for local use by laboratory users and administrators who need access to environmental and device information through a mobile application and dashboards (Grafana). The main outcome of the project is a working laboratory monitoring platform that can allow live counting while gathering and displaying historical and real-time data from various sources and lab equipment. Sensor data collection, wireless device connectivity, backend storage, dashboard-based monitoring, and user-facing application access are all successfully integrated into the developed system.	

Nevertheless, anomaly identification, trend prediction, and Blender's 3D digital twin visualization were not all finished. Overall, the project provides a practical basis for further development in IoT monitoring and digital twin applications for a laboratory environment with a possible future upscale.

Introduction

Modern campuses and laboratories rely on multiple sensors to monitor environmental and equipment conditions. However, data from these sensors is frequently dispersed across several systems, making real-time visualization, comparison, and analysis challenging. For laboratory managers and users, who require consolidated and easy access to current and historical data about the monitored environment, this poses practical issues.

The motivation for this project is to make laboratory monitoring data more visible, accessible, and useful. When information from sensors, computers, and other devices is spread across disconnected tools, it becomes harder to observe system behavior, identify changes in the environment, and support data-driven decision-making. By integrating sensing, communication, storage, and visualization into a single platform, a unified monitoring system can assist in resolving these problems.

The significance of the project lies in both its practical and technical value. Practically, it gives managers and users of laboratories a more efficient approach to keep an eye on equipment and ambient conditions in real time. Technically, it shows how a single computing-based system may incorporate embedded devices, wireless communication, data storage, visualization tools, and machine-learning-based people counting. As a result, the project functions as a foundation for future work in IoT monitoring and digital twin applications in addition to providing a helpful monitoring platform for a single laboratory environment.

The proposed solution is a digital-twin-based laboratory monitoring system that combines environmental sensors, microcontrollers, communication protocols, data storage, and visualization tools into a single platform. In the implemented system, ESP32-based devices collect real-time data from multiple sensors, while additional information from laboratory computers is also integrated into the monitoring pipeline. The data is transmitted through MQTT to a centralized server, stored in InfluxDB, and visualized through Grafana dashboards that are accessible through the mobile application. In addition, the system includes a people-counting component that provides live information about the number of people currently detected in the camera's field of view. This approach allows the project to support both real-time observation and historical monitoring within one computing-based environment.

The rest of this report is structured as follows. The report begins by outlining the limits that affected the project, including technical and practical issues that affected design and execution. The background and related work part then examines current computing-based monitoring techniques and relevant literature. The complete solution, including system features, architecture, tools, and workflows, is then presented in the project approach section. The development process over the course of the two semesters, including

design choices, difficulties faced, and teamwork contributions, is then described in the project execution section. After that, the project approach evaluation part presents how the solution was reviewed using user feedback from the survey. The report's key conclusions, limits, and potential avenues for further development are included in the conclusion.

Identification of constraints

- **Economic:** Although we had a limited budget, the primary concern was not a price, but the variety of sensors available and their number inside the lab. Unlike other projects, most of the equipment used was already present, like cameras, a variety of suitable microcontrollers, sensors and cameras. For this reason, apart from the power consumption and future extensions, our setup used the standard and non expensive base to organize a working system.
- **Environmental:** The main working space was designated for the laboratory area as an initial objective. Although the lab space contains variable factors such as noise level, presence of people inside the room, number of outside targets for counting using ML, there were physical limitations on how sensors and devices can be fitted in such a constrained area, opening issues for upscaling and functional expansion. The low quality resolution on the camera heavily limited the performance of the ML model and people detection reached a limit in accuracy, so adjustment of camera or target space for it may need some adjustment.
- **Social:** The main users of the system are the lab users and inside workers. Therefore, there is a limitation on how occupied space can be without causing inconvenience both physical and functional (occupying PCs, using an excess amount of needed microcontrollers). To ease up this concern, removal of Graphana and replacement by remote server may be considered. Lab users' feedback and personal opinion regarding the app and the system, opens up some opportunities and drawbacks that affect the project development course
- **Ethical:** The people-counting functionality sparked a discussion about privacy issues. The designed model counts the people present on camera but does not use facial recognition technology, and any other personally identifiable data is kept private. All bounding boxes annotated during training had one class - "person", while the inference process results in counting of people without providing their identification.
- **Health and Safety:** Continuous monitoring of various parameters including temperature, humidity, sound levels, and people counting helps track real-time conditions in the lab space. Thresholds set up for these metrics allow triggering alerts via Grafana that could be seen as an early warning system that ensures health and safety of the personnel working in the room.
- **Manufacturability and Scalability:** Simple hardware based on ESP32 microcontrollers programmed with MicroPython and basic sensors such as DHT11, ultrasonic sensors, and microphones allows easy replication of the hardware side. Open-source software (Mosquitto, Telegraf, InfluxDB, and Grafana) also simplifies implementation of this project. Scalability is a

potential issue for this project since adding another layer in the form of Grafana causes delays. Furthermore, scalability of the whole solution was not tested yet because no more than three ESP32 nodes, eight PCs, and two Raspberry Pi boards were used at a time.

- **Sustainability:** The reason why the ESP32 was selected over the Raspberry Pi for distributed sensing was due to its low power usage and built-in Wi-Fi module. Scripts that collect non-blocking data using Crontab scheduling prevent the unnecessary computation of data on PCs. In addition, efficient MQTT data transfer periods at QoS=1 levels further reduce computing power needed to process data.
- **Security and Institutional Policy:** The system runs entirely inside the university's LAN network, communicating on TCP port 1883. Running locally inside the network was a conscious decision in order to align with the policies set by the institution. However, this also means that the mobile application is unable to work unless the user connects to the local area network.

Evaluation and Trade-offs

These requirements created specific trade-offs during the implementation phase. For example, the decision to use a local-only network enhances security and reduces reliance on external internet stability, but it limits the ability to access data from outside the campus or use of external API's in order to get better and environment agnostic compute. Furthermore, the decision to use existing cameras reduces costs but limits the flexibility of the computer vision model to detect and learn on better images. To mitigate the risk of data loss, the system uses InfluxDB to ensure high-velocity data points are recorded accurately even during peak usage.

Person detection model

The person detection model was evaluated on the randomized validation subset of data after every training iteration based on **mAP@0.5:0.95 score**, with the best-performing checkpoint recorded at epoch 92 out of 100 total epochs, indicating the model was close to the fine-tuning limit of its given inputs mostly the image quality.

The final model achieved a precision of **0.758** and a recall of **0.735**, resulting in an **mAP@0.5 of 0.806** and a composite **mAP@0.5:0.95 of 0.514**. The mAP@0.5 score reflects strong detection confidence at a looser IoU threshold given the quality of the source, while the mAP@0.5:0.95 score of 0.514 indicates moderate performance under stricter confidence requirements.

A primary limitation is the quality of the camera feed used for image collection. The low image quality introduced noise and reduced clarity of person features, impacting the model's ability to generalize. We tried to mitigate this by switching to a larger YOLO11 variant, but it yielded little to no improvements in the metrics, resulting in under 5 to 10 percent improvements, while also costing exponentially more to compute, further proving that the bottleneck lies in the image quality rather than the model capacity. Under these constraints, the current model size represents the closest to the best computational choice.

In terms of resource intensity, the model is lightweight and suited for deployment in resource-constrained environments like Raspberry Pi. The model weighs **5,345 KB**, with a total of **2,590,035 parameters**. The estimated runtime memory usage is **9.9 MB** (float32), and the model consumes only **10.2 MB of VRAM** during inference, making it viable for edge deployment.

Background and Related Work

Prior Research and Existing Solutions

Modern research highlights the significance of adopting lightweight communication protocols within IoT ecosystems. According to Kodali and Valdas (2018), MQTT (Message Queuing Telemetry Transport) can be successfully applied for monitoring systems because of its lightweighness and use of publish/subscribe principle. Likewise, Zare and Iqbal (2020) noted that such microcontrollers as ESP32 could serve as cost-efficient substitutes for more traditional SCADA systems for data gathering. It seems like the combination of flexible microcontrollers and conventional protocols can provide an appropriate basis for real-time monitoring.

Computational Solutions

Data management is still one of the main issues in laboratory monitoring. According to Salituro (2023), the utilization of time-series databases (InfluxDB) is preferable for managing sensor data due to their suitability for working with high-frequency timestamp. Visualization is also considered an important part of the solution. According to McCollam (2022), Grafana became the industry leader for creating real-time dashboards.

Methodology Comparison and Justification

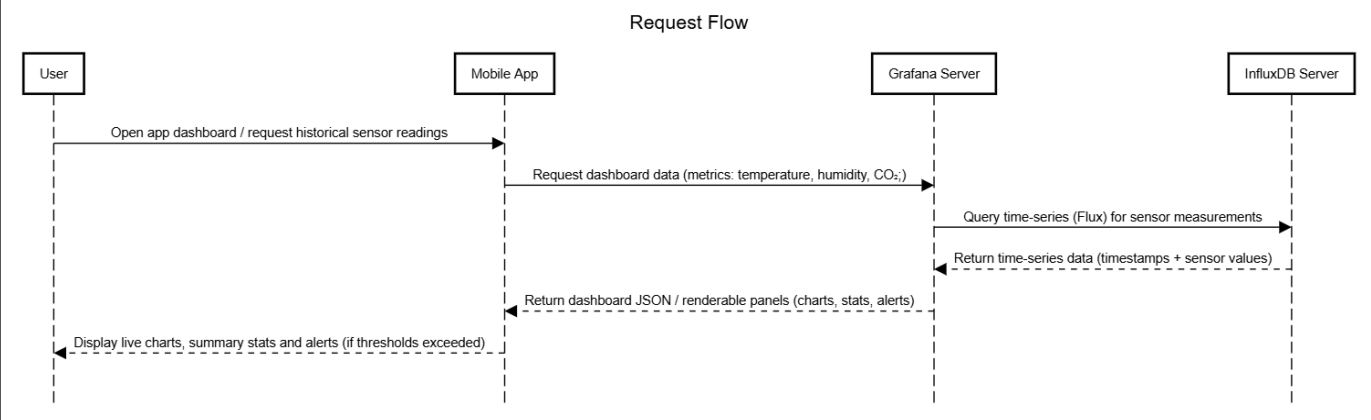
Among other solutions that could have been used are HTTP REST APIs and different microcontrollers such as the Raspberry Pi. The ESP32 was picked over other microcontrollers due to their greater computing capabilities compared to those of the ESP8266 and the use of MQTT instead of HTTP was due to its superiority in situations involving high latency or low bandwidth, a situation common in the congested environment within the campus. MQTT was used since the ESP32 has built-in Wi-Fi capability and consumes less energy than other alternatives.

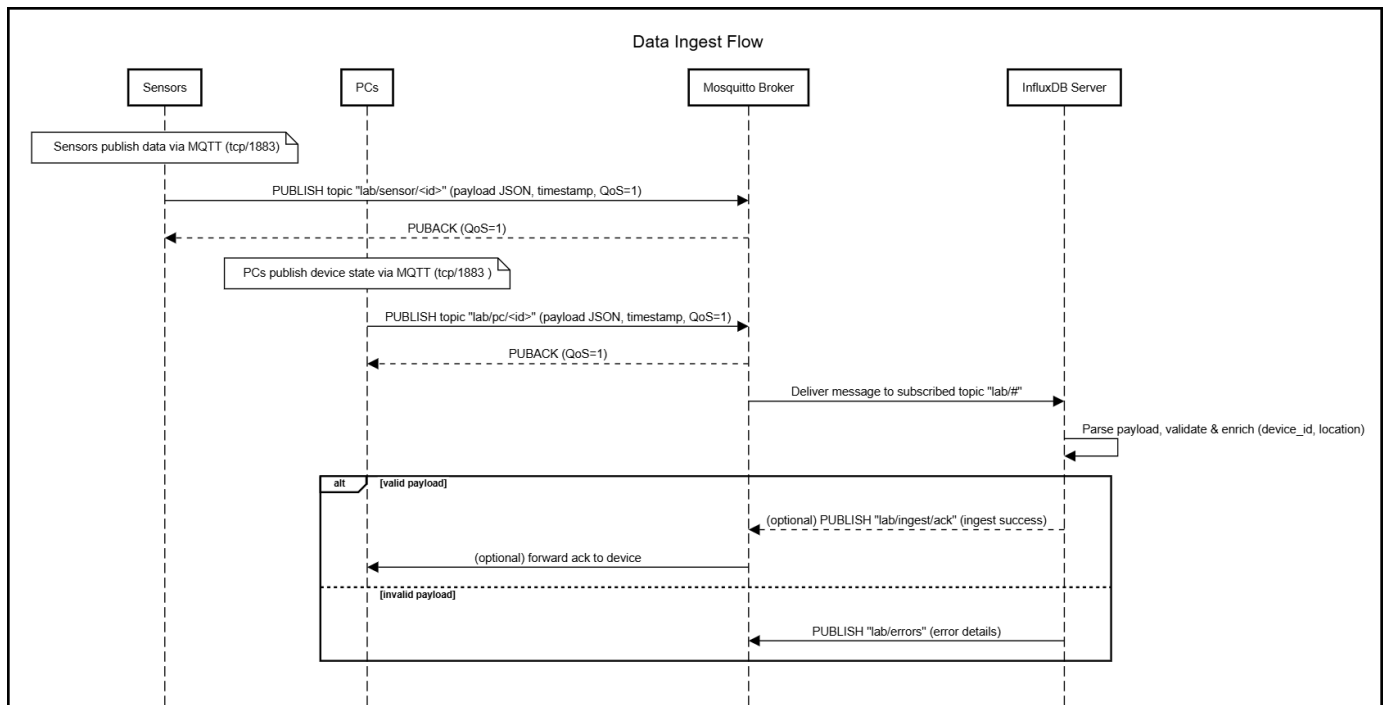
Use of these tools as a combination of ESP32, MQTT, InfluxDB and Grafana is a well-known approach within the field as it provides sufficient performance capacity for up to 500 simultaneous connected devices.

Project Approach

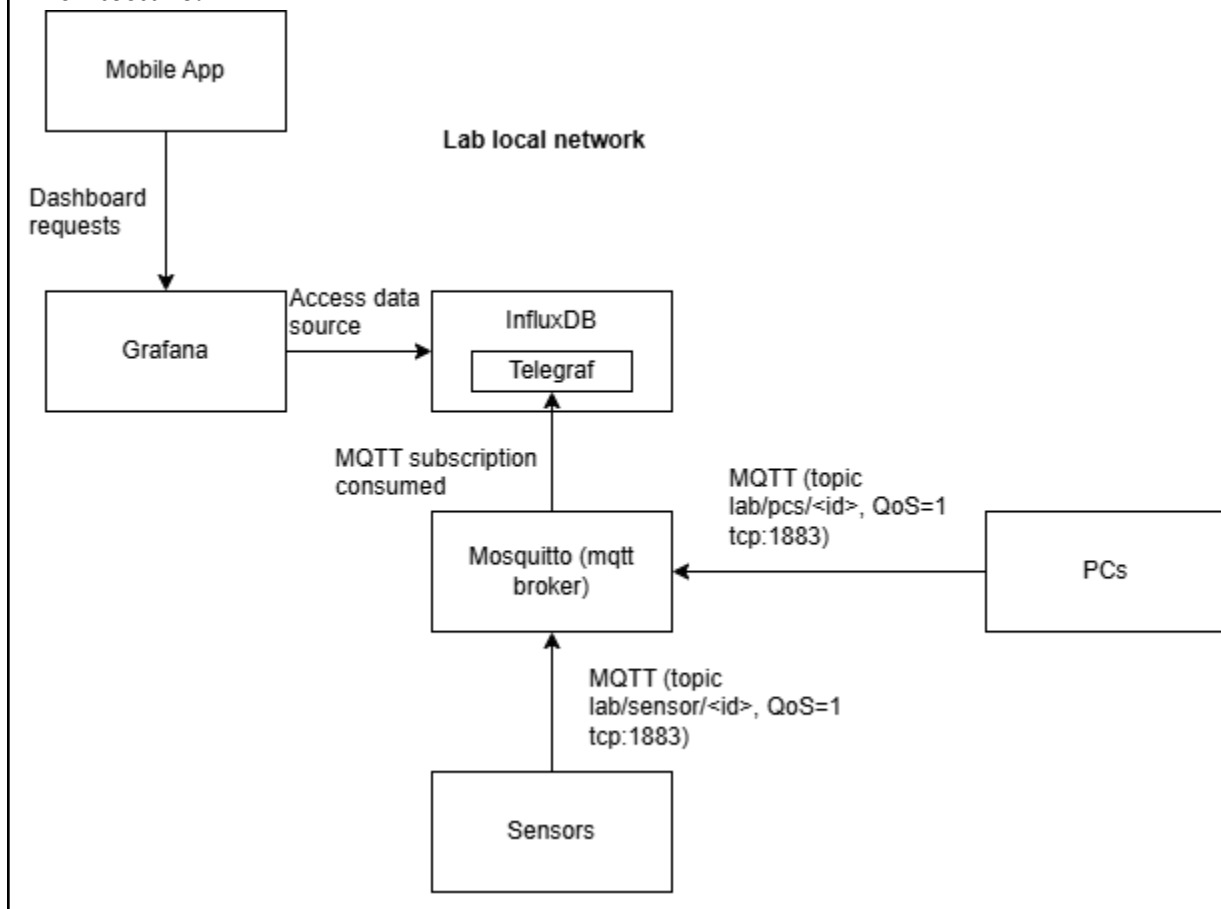
- **Use case:** Look up environmental data (dashboard view)
- **Stakeholders** are laboratory users - people working in and accessing the lab. The goal is to be able to monitor environmental conditions in the lab and PCs statistics via mobile application. Primary. High priority.
- **Preconditions:** User is connected to the local network via WiFi
- **Primary actor:** Laboratory user
- **Trigger:** User opens the mobile app
- **Success:** User opens the mobile app. Grafana queries InfluxDB for the metrics. The mobile app displays readings from the sensors and PCs. The user sees historical values from the laboratory devices connected to the system.
- **Exceptions:** Device is offline/ not connected to the local network -> show cached data. No data available -> show no recent readings. Sensors return errors -> show available data with warnings.

Sequence diagrams:





Architecture:



The project relies on several third-party components to handle data collection, storage, visualization, script execution and display:

- Mosquitto (MQTT broker) for communication between devices and the server. Sensor data from ESP32 and PCs is published through MQTT and then processed and stored.
- InfluxDB to store environmental and PC data.
- Grafana to create dashboards and visualize sensor data. It uses InfluxDB as a data source.
- React Native to display Grafana dashboards on mobile phones.
- Crontab to schedule script execution and enable autostart of servers on PCs.

These integration allowed the team to focus on system design and user experience rather than building all components from scratch.

The team collaborated by dividing responsibilities and maintaining communication. Initially, all members participated in setting up the system, including configuring PCs, sensors, InfluxDB, and Grafana.

After the setup phase, two people were developing the mobile app using React Native and the rest three were developing the machine learning algorithm. Two people took leadership roles by organizing tasks and scheduling meetings.

On crucial milestones, the team met in the lab to discuss progress, identify problems, and find solutions. Survey results were taken into account when making an update for the mobile app, shown in improvements such as better layout organization and the addition of the ML tab.

This structured division of work and continuous collaboration allowed the team to efficiently develop and improve the computing-based solution.

Project Execution

Over the last two semesters, we worked on building a system that collects environmental data from sensors and displays it through a mobile application.

At the beginning, we installed the Mosquitto client on 8 PCs and 2 Raspberry Pis, and configured a server (one of the PCs) with Mosquitto broker, InfluxDB, and Grafana. PCs and sensors communicate with the server using MQTT. The data is then processed by Telegraf (InfluxDB plugin), stored in InfluxDB, and visualized in Grafana.

We used three ESP32 microcontrollers and sensors, including an ultrasonic sensor, a microphone, and a DHT11 (temperature and humidity) sensor. To send messages via MQTT, we wrote a script for PCs and scheduled it with Crontab to run every 5 seconds. We also wrote one general script for microcontrollers, where we used threads to enable parallel execution.

To extend the system's environmental monitoring capabilities, a person detection model was developed and integrated as a monitoring component. The model's purpose is to count the number of people, as well as label them, present in Atrium.

For this component, we selected YOLO11 (You Only Look Once, version 11) as the object detection architecture, due to its high inference speed and detection accuracy, making it suitable for real-time application. The model was fine-tuned on a custom dataset of annotated images collected specifically for this project. All annotations contained a single class, person, and bounding boxes were labeled manually prior to training.

The training pipeline started with labeling around a thousand pictures taken prior and labeling them with bounding boxes. The dataset was split into training and validation subsets using an 80/20 ratio, and standard data augmentation techniques were applied during training to improve generalization. The model was evaluated using mean Average Precision (mAP), which served as the primary indicator for selecting the best-performing checkpoint across training epochs.

The resulting model returns bounding box predictions for all detected persons within a frame, enabling the system to derive occupancy counts.

During development, we encountered several challenges. For example, initially we wanted to use Grafana's official mobile app, but it did not display dashboards and also did not work locally. Instead, we decided to build our own mobile app. At the start, we considered using a Progressive Web App (PWA), but since we were not familiar with it we wrongly assumed that it would be fast and easy, so we switched to React Native, which we had prior experience in.

Another issue we faced was app deployment. While we develop the mobile app on a cross platform framework which supports ios operating system, releasing it was not feasible due to Apple's paywall and hardware requirements (needing a Mac and iPhone). As a result, we focused mainly on Android.

Overall, the project was successful and we managed to achieve real-time data visualization. However, we faced difficulties with platform limitations and initial design choices, which required us to change our approaches.

Overall, teamwork played an important role in our project. We regularly gathered in the laboratory to work together, which allowed us to quickly identify and solve issues. It helped us solve problems faster and stay within the specified timelines.

At the start, all team members contributed to the initial setup, including configuring PCs, sensors, InfluxDB, and Grafana. This was a conscious decision to ensure that everyone understood the system architecture. After that, responsibilities were divided based on our specializations. Sherniyaz and Nursultan worked on developing the mobile application, while Saida, Asset, and Miras focused on the machine learning model.

For troubleshooting, we relied on group discussions where each member could propose a solution. If anyone encountered an issue, we worked together to find a solution.

Sherniyaz and Miras took leadership roles. They organized the workflow, divided the tasks among the team, and scheduled meetings. This helped to keep the project structured and ensured that everyone stayed on track.

Evaluation

We evaluated the project by releasing an android version of the app on GitHub page and asking lab members to test it, after that we asked them to complete a survey. After we collected the survey data, we noted areas of improvement and updated the mobile app. Since there are not many lab members, we decided to switch focus from quantitative evaluation to qualitative, and conducted an interview to evaluate the mobile app after the update.

The evaluation process validates the mobile app by directly involving end users (lab members) who are the target audience of the application. Their feedback is the most reliable way to evaluate how well the system performs.

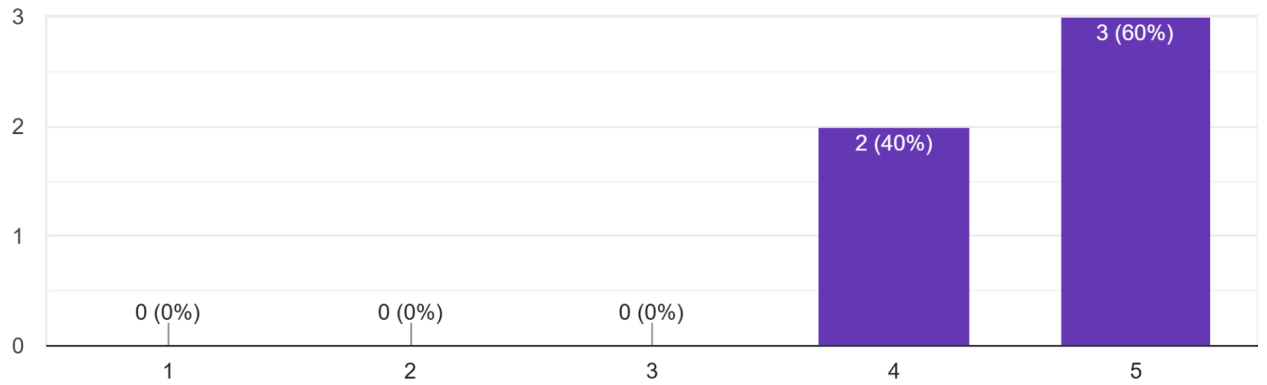
From the survey results:

- Users rated the app positively in navigation, responsiveness, data readability, and layout.
- Overall satisfaction is high.
- The layout and non-interactive WebView design were generally accepted, though some users suggested adding limited interaction (zooming).

Survey results:

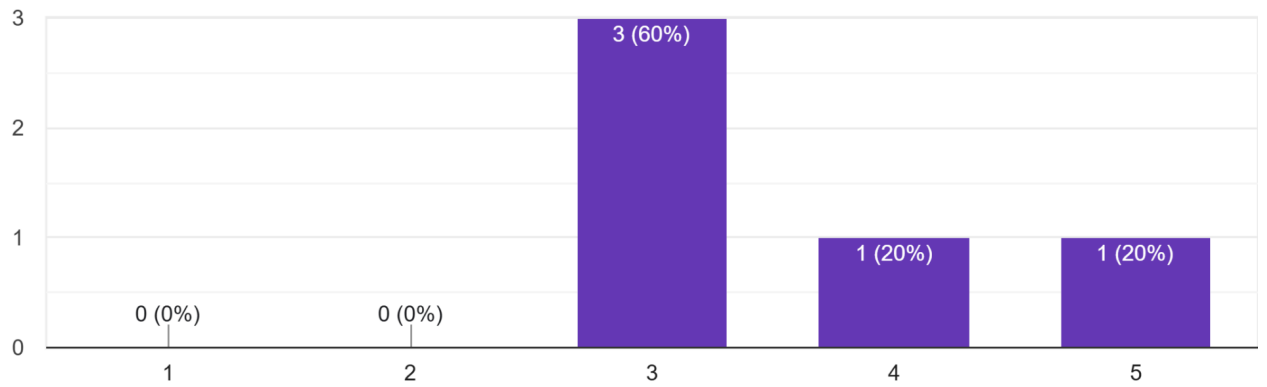
How easy is the app to navigate?

5 responses



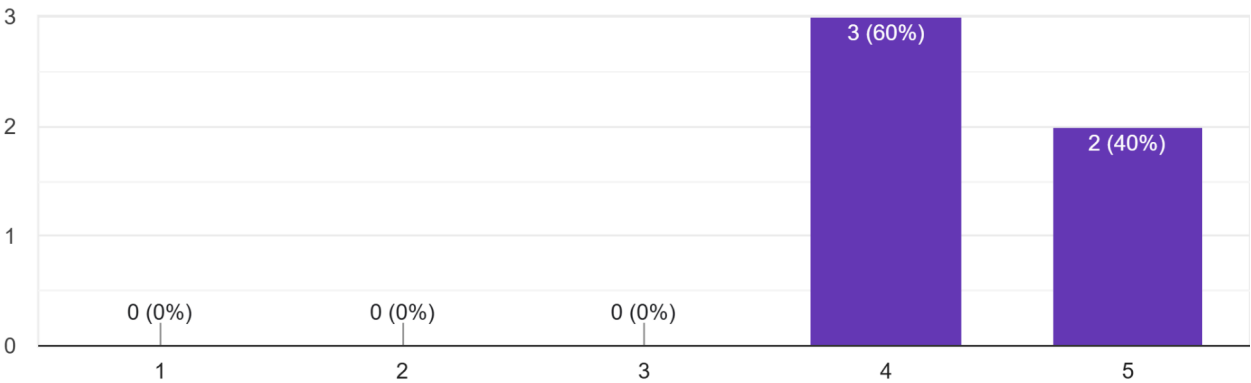
The app is quick and responsive.

5 responses



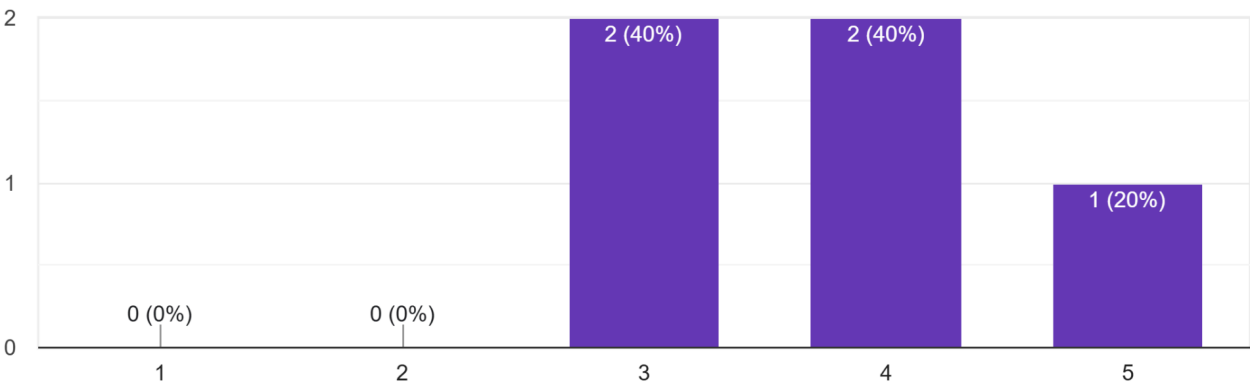
The data in the panels is easy to read.

5 responses



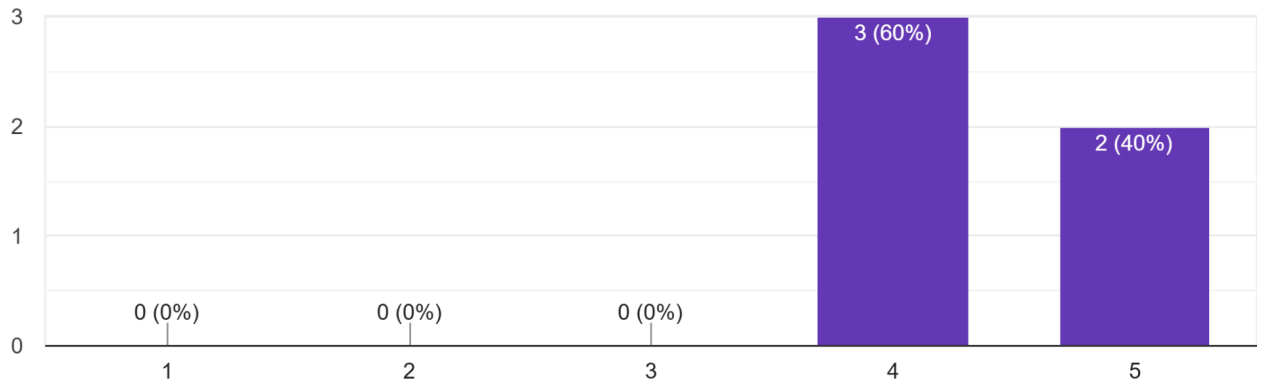
The layout of the app is visually clear and organized.

5 responses



Overall, I am satisfied with the app.

5 responses



Q: What improvements would you suggest? (Optional)

- Map page no interactables, maybe a window on statistics of a specific selected target?
- reduce the number of dashboards, make them organized. improve the design, ui.
- Possibly more detailed and visually representative 2D map. Maybe subsections for better categorization of devices to navigate easily between them
- Make app for the IOS users

From the interview:

- User noted that the app became more compact after the update.
- One of the identified issues is device statuses disappear after minimizing the app.
- Overall, the app slightly improved.

The interview:

Q: Compared to the previous version, what were the most significant differences you noticed in this updated app?

A: All of the devices on/off statuses were consolidated on top of the page, now everything takes less space.

Q: Do you think the app has improved after the changes?

A: Slightly Improved. One of the problems I encountered is that when you minimize and then go back to the app, the said statuses on the top page disappear.

Q: Did anything feel confusing or unclear while using the app like layout, labels, or flow?

A: more or less ok

Q: How would you rate integration of the ML model into the app? What could be improved?

A: ML updates were not happening, the photo is stuck, whereas the graph was working ok.

(The photo serving server was not configured to autostart yet, so that's why it didn't update)

Q: Are the card/panel sizes generally appropriate for the data they display?

A: Yes, sizes are appropriate for phones, but look worse on the tablet.

Q: What is the single most useful improvement made in this version?

A: I suppose camera feed

Q: What is still missing or needs the most immediate improvement?

A: Maybe some filtering options on the main screen.

Q: If you could change only one thing in the entire app right now, what would it be?

A: The map of the devices is still inactive

Q: Do you have any additional comments or suggestions not covered above?

A: No

The evaluation shows that the mobile app reached a satisfying level, but there are still unresolved issues that will need to be addressed in the future updates. However, some of the limitations and drawbacks of the app are a direct result of the architecture, or rather reliance on 3rd party software, notably Grafana.

Conclusion and possible future work

To summarize, in this project we successfully developed a well-balanced and functional base/prototype for the laboratory monitoring system that operates consistently within the confinements and restrictions of the lab environment. Most of our functional and nonfunctional requirements in regards to the system performance, responsiveness and consistency that were set initially are met except for some extensions that were planned as an additional feature of the project (3D map). The project development allowed us to gain and apply knowledge of the:

- MQTT - main communication protocol
- Graphana + InfluxDB - visualizing tool for the dashboards + main database using Telegraf
- CronTab - tool for scheduling and autostarting the project scripts
- ML model - YOLO11 for detecting and counting of the object
- Edge Impulse - main tool used for annotation of the images
- React Native - for mobile app. development

Working with said tools allowed us to learn practical applications and gain versatile skills, giving a deeper understanding of the logic behind developing distributed systems. The system effectively supports the collection, delivery, and visualization of both real-time and historical data from laboratory devices such as PCs, sensors, and microcontrollers, additionally integrating a ML based people-counting feature. As a result, the main objective of improving accessibility and providing a simple visualization of laboratory data through a mobile application and dashboards was successfully achieved. Therefore, the

project may be considered as a solid foundation for a more digital smart monitoring systems for laboratory or more complex settings.

One of the core limitations of the system is that it is working entirely locally with restrictions to the laboratory network. PCs and Sensors have to share the same WiFi network, despite that the lab has multiple networks for different purposes. Correspondingly, the mobile application is also limited to such WiFi constraints to display dashboards and correct data gathered in the lab, so outside usage for users is heavily limited, making it impractical for a distant monitoring. Another technical problem considers the performance and scalability of the product. In addition to this, while Graphana simplifies the development and visualization, it acts as an intermediary level between database and lab devices, affecting the latency and performance of the system. Given the mentioned delays of the mobile app in the questionnaire, it would be better to take measures to optimize the communication between the database and devices.

Future work should include finishing initial additional extensions of the projects, such as a creation of a 3D Blender model of the system, as well as an anomaly detection feature implementation. Furthermore, the performance and scalability limitation problem should be addressed by removing the middle layer (Graphana) and implementing a remote server, which opens the possibility to improve latency of the system, as well as allowing for remote access to the lab dashboards and mobile application. Mobile application design improvements and more interactive features should be implemented to improve lab users' experience. These additional features would contribute to the transformation of the system toward a more scalable and intelligent monitoring form.

The project establishes a strong and practical base for future research and development in IoT-based laboratory monitoring and digital twin applications.

References

- [1] Banner Engineering, "Miniature self-contained photoelectric sensors in a universal housing - Datasheet," Rev. H, Document 128911, Oct. 17, 2016. [Online]. Available: <https://info.bannerengineering.com/cs/groups/public/documents/literature/128911.pdf>
- [2] I. Hindi, M. Alyaman, A. AboZenah, A. Zaid, and M. Shrara, "Smart Alarm IoT system: Monitoring elevator traffic and meteorological data on job sites using MQTT and InfluxDB integrated with Grafana," in *Proc. Int. Conf. Information and Communication Systems (ICICS)*, 2024, pp. 1-6. doi: 10.1109/ICICS63486.2024.10638309
- [3] R. McCollam, *Getting Started with Grafana: Real-Time Dashboards for IT and Business Operations*. Apress, 2022. doi: 10.1007/978-1-4842-8309-7

- [4] Mouser Electronics, "DHT11 Humidity & Temperature Sensor - Technical Data Sheet," Datasheet 1143054. [Online]. Available: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>
- [5] R. K. Kodali and A. Valdas, "MQTT Based Monitoring System for Urban Farmers Using ESP32 and Raspberry Pi," in *Proc. 2nd Int. Conf. Green Computing and Internet of Things (ICGCIoT)*, Bangalore, India, 2018, pp. 395-398. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/8752995>
- [6] E. Salituro, "Extracting and visualizing data with InfluxDB and Grafana," in *Learn Grafana 10.x*. Packt Publishing, 2023.
- [7] A. Zare and M. T. Iqbal, "Low-Cost ESP32, Raspberry Pi, Node-Red, and MQTT Protocol Based SCADA System," in *Proc. IEEE Int. IoT, Electronics and Mechatronics Conf. (IEMTRONICS)*, Vancouver, BC, Canada, 2020, pp. 1-5. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9216412>
- [8] M. F. Usmani, "MQTT Protocol for the IoT - Review Paper," 2021. doi: 10.13140/RG.2.2.26065.10088
- [9] Seeed Studio, "Grove - Gas Sensor (MQ3)," Jan. 3, 2023. [Online]. Available: https://wiki.seeedstudio.com/Grove-Gas_Sensor-MQ3/
- [10] Espressif Systems, "ESP32 Wi-Fi & Bluetooth SoC." [Online]. Available: <https://www.espressif.com/en/products/socs/esp32>