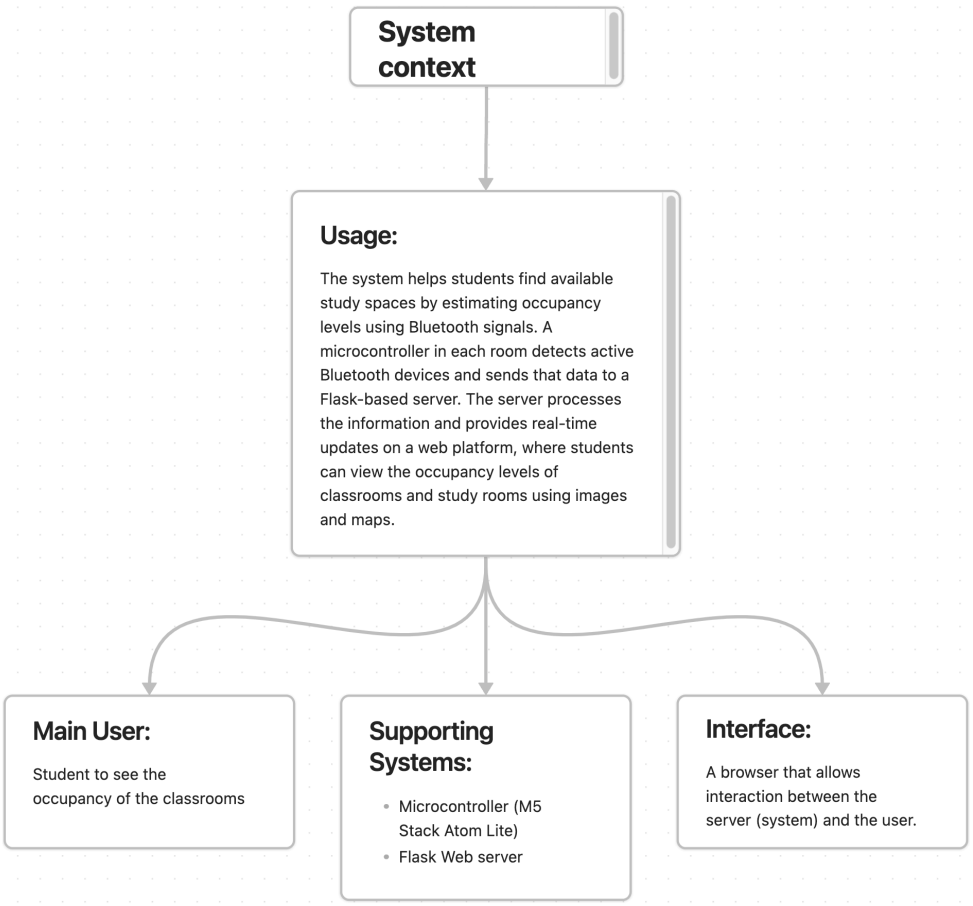
Report – 999

Introduction

This project was developed as part of the System Oriented Programming course, with the goal of creating a system that estimates the number of people in specific university classrooms or study areas. The system uses Bluetooth technology to detect nearby devices, which serves as a proxy to estimate occupancy levels in near-real time.

Ideally, each room would be equipped with a microcontroller (M5Stack Atom Lite) that periodically scans for Bluetooth-enabled devices within its range. The data collected would then be used to estimate the number of people and sent to a central server, which processes the information and displays it on a web page. This would allow students to easily check which rooms are crowded, helping them choose quieter places to study.

The main aim of the project remains to simplify the process of finding available study spaces, using lightweight hardware and an accessible web interface.



Project structure

The system architecture follows a C4 model to describe the various levels of the application, from general context to internal components. The corresponding diagram can be found throughout the report. The system is divided into three main parts:

Microcontroller

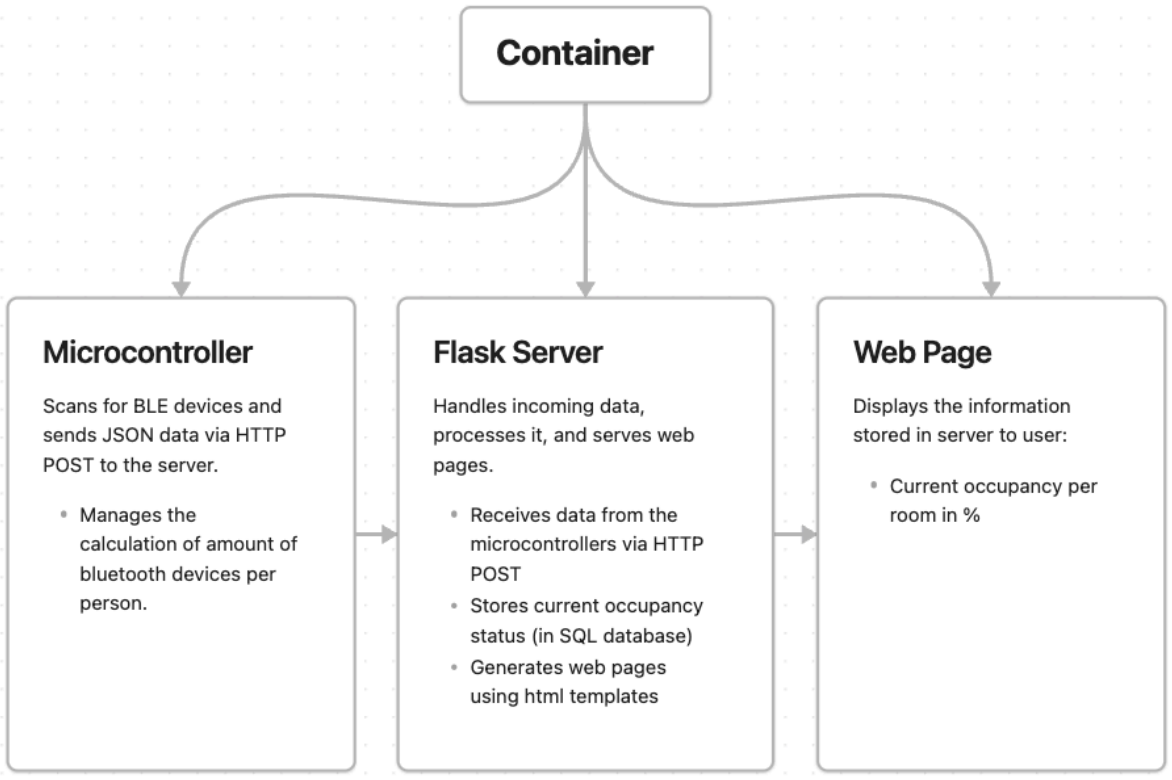
The microcontroller, an M5Stack Atom Lite, would be installed in each classroom and is responsible for collecting real-time occupancy data. It would continuously scan the room for nearby Bluetooth devices using BLE (Bluetooth Low Energy). Based on the number of devices detected, it estimates how many people are present in the room. This data would then be formatted as a JSON object and sent to the central server via HTTP POST requests. The microcontroller also manages the Wi-Fi connection setup and handles all communication logic that would ensure smooth data transmission.

Flask Server

The Flask server functions as the central processing unit of the system. It would receive the occupancy data sent by the various microcontrollers placed in different rooms. Upon receiving the data, the server would update a SQLite database to reflect the most current occupancy count for each room. It also renders and serves web pages using the Jinja templating system. These pages would be updated periodically to display the latest information collected from the microcontrollers.

Web Page (Frontend Interface)

The web page serves as the user interface that students access through their browser. It presents a clear and user-friendly view of the current occupancy levels for each monitored room. The interface includes visual elements representing room occupancy to help students quickly identify available spaces. In addition to showing current data, the system is also designed to accommodate future features, such as displaying daily summaries or weekly trends to help students plan their study sessions more efficiently.



Implementation

Bluetooth detection and data transmission

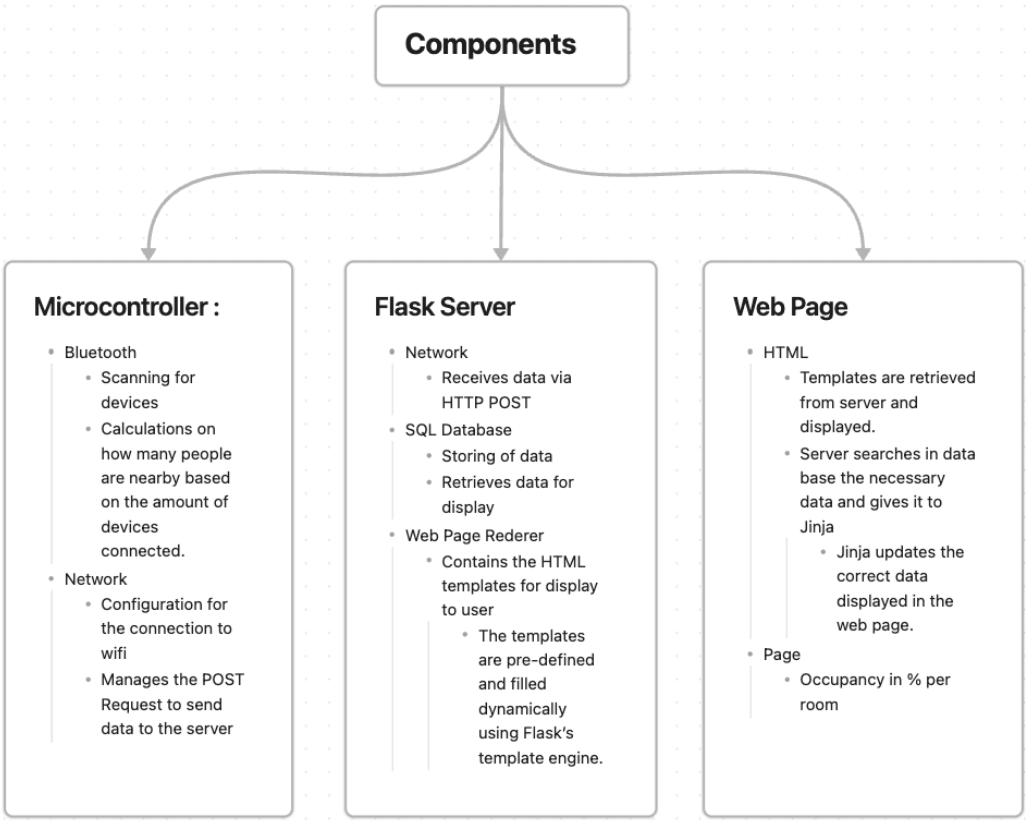
The M5Stack Atom Lite, programmed in C/C++, scans for nearby Bluetooth Low Energy (BLE) devices at regular intervals. Once the number of devices is detected and filtered, the microcontroller would package this data (including the room identifier and the estimated number of people) into a JSON object. It would then connect to the server through TCP and send the data over HTTP using a POST request. To achieve this, the microcontroller first needs to connect to the university's Wi-Fi network. However, due to compatibility issues with the university's Wi-Fi network, this functionality could not be tested in the intended environment, even though it worked correctly on personal hotspots.

Data reception and storage on the server

The Flask server, developed in Python, receives the incoming POST requests via a specific API endpoint. Once received, the server would parse the JSON data and update or insert the corresponding room occupancy in a lightweight SQLite database. The server acts as the backend of the system and contains HTML templates that can be dynamically rendered using Flask's built-in Jinja2 templating engine.

Web interface and display

The frontend is a basic HTML page served by Flask. It is designed to display real-time data pulled from the SQL database, a visual representation of the classrooms or study spaces and the information of the occupancy levels in each room. This web page is accessible through any browser on the university's Wi-Fi network, allowing students to easily check room availability.



Conclusion

This project demonstrates a practical and lightweight solution for monitoring classroom and study room occupancy using Bluetooth-based detection. By combining a microcontroller (M5Stack Atom Lite) with a Flask web server and a simple frontend interface, the system provides a real-time, user-friendly platform to help students find available spaces more efficiently.

Although this version represents a solid foundation, several improvements remain necessary to fully realize the project's potential. These include enhanced device filtering to avoid false detections, richer data visualization to make occupancy information more intuitive, and the integration of advanced features such as historical data tracking and a more interactive user interface. Most importantly, establishing stable connectivity between the microcontroller and the university’s Wi-Fi network is essential in order to enable real-time data collection from each room. This final step is critical for the system to operate as intended in its actual deployment environment.

Overall, this project combines hardware and software elements effectively to meet a real-world need, using accessible and open-source technologies that are adaptable and easy to maintain.

Individual Report - Han Kieu

What were the most challenging parts of the course, including the project?
Some of the most challenging topics during the course were memory allocation and working with Bash. These concepts were relatively new to me and took more time to fully grasp. Additionally, I found the format of the course to be slightly difficult to follow. While I appreciated the use of “the island” as a way to visualize the structure and content of the course. However, it was sometimes unclear how best to engage with the material. I often felt uncertain about whether I should be reading the content, taking notes, or focusing on listening during lectures.

One significant difficulty was preparing for the final exam. Since it was closed-book and the assignments were mostly practical, it was hard to anticipate the types of theoretical questions that might appear on the exam. I’m grateful that a mock exam was eventually provided, as it helped clarify expectations.

What I liked about the course
The use of “the island” was one of my favorite aspects, it made the course feel more cohesive and helped me track our progress. Despite the challenges in navigating the format, the overall design was engaging and creative.

What I liked about the project
What I appreciated most about the project was the autonomy we had. The freedom to define our own direction allowed our team to pursue an ambitious idea, which was both motivating and rewarding. It gave the work a sense of purpose and relevance. However, that same freedom also presented a challenge: we may have taken on a project that was too complex, particularly for group members with less programming experience. One of the hardest parts for me was understanding the server-side component.

What I learned during the project
I gained hands-on experience with modifying websites using HTML, CSS, and the JustGage library. I also learned how to program a microcontroller and use Platform.io effectively. Most importantly, I got valuable practice with GitLab, which I know will be a crucial tool in my future work.

Note: this text has been reworded by an AI for a more professional language. It has been reviewed by a human (me) for accuracy.

Individual Project Report – Léonard Clément

This report details my contributions, challenges, and learning experiences in the project, focusing on networking and team coordination.

Challenges Encountered

My primary role involved implementing the **networking layer** for both the microcontroller (Atom-Lite) and the server. I also served as a **team coordinator** as I happened to have more experience.

The main technical challenge was connecting the Atom-Lite to the university's **eduroam Wi-Fi**. Despite extensive efforts, a connection couldn't be established, though the code functioned on standard WPA-Personal networks. From a **project management** standpoint, coordinating a team with varied programming expertise posed difficulties in task distribution. Nonetheless, I believe every team member contributed effectively, aligning with their individual skills and available time.

Lessons Learned

While I was already familiar with **C programming**, this project significantly improved my fluency. My most significant new learning area was **embedded and microcontroller programming**, a subject I'd long wanted to explore. Debugging and dependency management proved the most challenging aspects in this domain.

Ironically, **network programming** is where I learned the most. On the microcontroller side, I gained practical experience with **TCP sockets** and **HTTP POST requests**. For the server, I learned **Flask, Jinja2, and Docker**, skills I anticipate will be invaluable in my future studies. Setting up the university server also offered a chance to learn basic **system administration**, though its utility was ultimately limited by the microcontroller's inability to connect to the university's intranet. I would have liked to learn Docker Compose, but time constraints prevented it.

General Project Feedback

I found this project highly engaging and beneficial, and I recommend its continuation. Starting it earlier in the semester could allow for the development of larger systems, as such projects offer substantial learning opportunities beyond the course's immediate scope.

Note: this text has been reworded by an AI for a more professional language. It has been reviewed by a human (me) for accuracy.

Individual Report - Mathilda Muster Labeau

This report reflects on the experience gained during the System Oriented Programming course project, focused on building a websensor-based room occupancy estimation system.

- General challenges encountered:

1. The number of Bluetooth devices detected varies greatly between individuals, making estimation approximate.

2. We could not connect to the university's intranet Wi-Fi, which limited live testing and deployment.

3. Some Bluetooth signals in rooms came from university equipment, adding noise to the data.

4. The effective range of the Bluetooth detection was unclear, which impacted accuracy.

- Involvement and Technical Experience

With limited prior knowledge in system-oriented tools such as Flask, HTML, and JSON routing,the project required significant self-learning and adaptation. The server and web interface were studied and tested locally. An experimental version of the site was built from scratch, which led to a better understanding of the structure of a Flask project: routing logic, connection to templates, and the role of Jinja2 in dynamic rendering.

The server and web interface were studied and tested locally. An experimental version of the site was built from scratch, which led to a better understanding of the structure of a Flask project: routing logic, connection to templates, and the role of Jinja2 in dynamic rendering. The full system workflow became clearer with time, particularly how data flows from the backend logic through SQLAlchemy to the visual HTML output.

- Lessons learned

This project offered a first practical insight into the structure of a system-oriented application, from sensor-based data acquisition to full backend-to-frontend integration. The coordinated interaction between components—microcontroller, server, and client interface—highlighted the principles of modular design and data flow.

Even without directly contributing to the firmware development, observing the structure and communication protocols (HTTP POST with JSON, database updates) clarified the role of system orchestration. The diversity of tools introduced during the project, particularly open-source technologies, provided a valuable foundation for future work in backend systems and web development.

Individual Report - Vania Piller

The System Oriented Programming course was my first real experience working with the C programming language and learning how it interacts directly with the system. Coming from higher-level languages like Python or Java, I found this course both challenging and eye-opening. C gave me a deeper understanding of how computers actually manage memory, interact with hardware, and handle low-level operations, something I had never really considered before.

One of the most difficult parts of the course for me was understanding pointers. The idea of directly referencing memory addresses and manipulating them felt abstract at first. It took time for me to grasp how pointers worked, how they could be passed between functions, and how pointer arithmetic could be used to navigate arrays or structures. The connection between arrays and pointers also caused some confusion, but slowly it started to make sense through practice and exercises.

Another area that I struggled with was dynamic memory allocation, especially using `malloc` and `free`. In higher-level languages, memory management is usually automatic, so manually reserving and releasing memory was entirely new to me. At first, I made many mistakes but through debugging and trial and error, I began to understand the importance of careful memory handling in C. I now see how memory leaks and segmentation faults happen, and I have a much deeper appreciation for what's happening "under the hood" in programs.

During the project, I encountered some difficulty understanding how the C and C++ code on the microcontroller integrated with the different parts of the project. Although I was not directly responsible for implementing the core functionalities, I made a conscious effort to follow the overall structure and logic, particularly regarding how the microcontroller was programmed to detect nearby Bluetooth devices and transmit the data to the Flask server developed in Python. This experience gave me a better understanding of how lower-level operations performed by hardware components can be integrated with higher-level systems responsible for processing data and delivering a usable interface to the end user.

Overall, this course was challenging but valuable. I learned how powerful and dangerous system-level programming can be, and how much control C gives you over memory and performance. Although I still feel I have much more to learn, especially with pointers and more complex memory structures, I now have a strong foundation to build on in future systems courses.