UNIVERSITY OF MIAMI
COLLEGE of ENGINEERING

Department of Electrical and Computer Engineering

SENIOR DESIGN PROJECT

FASTPASS

By

Bana Zoumot, Computer Engineer
Nick Ramos, Electrical Engineer
Jenn Viard, Software Engineer

Final Report

Spring 2025

Advisor: Dr. Nigel John

May 5th, 2025

ABSTRACT

FASTPASS is a smart campus solution that gives students and faculty real-time occupancy and noise-level data for the most popular study locations. By using data from LiDAR sensors, microphones, and Wi-Fi sniffers, FASTPASS helps users quickly find quiet, available spaces on campus. This system broadcasts up-to-date information to the University of Miami community, making it easy to check study space availability and ambient sound levels before heading out. Our pilot deployment at the University of Miami's Richter Library includes ten ESP32-based sensor units that continuously stream live occupancy and audio data. These low-cost units gather environmental data and upload it to a central server, where it's processed and made available to users through a simple interface. With FASTPASS, students and faculty save time, avoid overcrowded or noisy environments, and make more informed decisions about where to study. The result is increased productivity and a more efficient, user-friendly campus experience.

Table of Contents

1. Introduction

1.1. Problem statement

Many students waste valuable time searching for quiet, available study spaces on campus due to a lack of real-time occupancy and noise-level information.

1.2. Problem Description

Many students waste time walking from study space to study space, only to find them overcrowded or too noisy. No existing solution provides real-time data on occupancy and sound levels for study environments on the University of Miami campus. Our goal was to create a reliable and scalable solution that empowers students with real-time insight helping them make informed decisions, save time, and improve their study experience across campus.

1.3. Market Needs

FASTPASS targets the student population at universities, particularly those who rely on shared campus study spaces such as libraries, lounges, and academic commons. The application is designed to improve the study experience by addressing five key market needs: reducing overcrowding in shared student spaces, providing real-time data to help students plan their study sessions effectively, minimizing disruption from noisy environments, eliminating time wasted commuting to already full or unsuitable study areas, and offering a scalable and privacy-conscious infrastructure that can be deployed campus-wide. By solving these pain points, FASTPASS improves productivity, enhances convenience, and increases student satisfaction, making it a valuable tool for a modern, data-informed campus experience.

1.4. Engineering objectives and system specifications

This project falls within the fields of embedded systems, Internet of Things (IoT), and real-time data processing. It meets several key engineering objectives: enabling real-time monitoring of study zones to track occupancy and noise levels, creating a seamless frontend interface for students to easily access live data, implementing an efficient backend architecture using Firebase for reliable communication and data management, integrating sensor data from sound, distance (LiDAR), and Wi-Fi sniffing to accurately assess space utilization, and deploying scalable and secure ESP32-based hardware that can be expanded across campus. These objectives ensure the system is technically robust, responsive, and adaptable to future needs.

1.5. Functional description of system sections

1.5.1. Level 0

At the highest level, the FASTPASS system functions as a real-time feedback loop between students and campus study environments. Students interact with the system to receive up-to-date information about study space conditions, including occupancy and noise levels. This interaction allows them to make informed decisions before physically visiting a study location.

1.5.2. Level 1

The system is composed of three core components: the hardware sensor nodes, the Firebase backend, and the React web frontend. Each sensor node includes a LiDAR module, a microphone, and an ESP32 microcontroller that collects and transmits data. The Firebase backend handles real-time communication and data storage. The React-based web frontend provides users with an intuitive interface to view space conditions across campus.

1.5.3. Level 2

Each sensor node includes specific modules for distinct measurements: the LiDAR detects motion and determines overall first floor occupancy, the microphone measures noise levels using decibel readings and Fast Fourier Transform (FFT) analysis, and the Wi-Fi sniffer estimates device count based on Received Signal Strength Indicator (RSSI). The Firebase database forms the backbone of the real-time data pipeline, ensuring continuous data flow from sensors to frontend. The user interface supports search filters, a gallery-style view of study spaces, and an occupancy engine that displays space availability and noise in an accessible format.

2. LITERATURE RESEARCH

2.1. Background and Market value

Libraries have been a really important part to the academics of college campuses. According to the American Library Association's report *The Value of Academic Libraries*, libraries are needed to support student learning, academic research, and campus well-being, playing a valuable role in keeping students from transferring to other colleges or dropping out all together (ALA, 2024). The Universities and Colleges Admisitons Service agrees that "research has found that learning environments play a crucial role in student success," highlighting the importance of providing well-structured, comfortable study spaces for improved academic performance (UCAS, 2023).

Here at the University of Miami and on college campuses across the country, students frequently waste valuable time searching for suitable study spaces during peak academic periods, especially during finals seasons in April to May and November to December. They move from library to library, study spot to study spot, often encountering overcrowded, noisy, or unavailable locations. This inefficiency negatively affects their productivity, increases stress levels, and can hinder academic performance.

A survey published by University Business revealed how students utilize library space: 58.6% spend their time in quiet study areas, 37.8% in computer labs, 35% in reading rooms, 33.8% in cafés, and 32.2% in group study spaces (University Business, 2024). The need for these places are clear.

The problems from overcrowded libraries are not only seen by students but also by library staff. "The overcrowded library not only affects the students trying to study, but it has put a lot of stress on librarians Susan Augustine and Shirley Volk, who are trying to adapt to the crowds," said Augustine in an interview with *U-High Midway* (U-High Midway, 2023).

Frustration among students is growing nationwide. "Looking for places to study is very aggravating considering the more comfortable places on campus are always crowded," said public health sophomore Hannah Ortega. "While studying in these spaces, I feel suffocated, and sometimes the Wi-Fi crashes making it impossible to get work done" (*The Daily Cougar*, 2024).

The Educause Center for Analysis and Research also reports that students list the availability of quiet study spaces as one of their top infrastructure concerns, with over 60% stating it directly impacts their academic success (Educause, 2023). A 2022 Inside Higher Ed article notes that "campus libraries are at the center of learning but are increasingly strained by student demand for flexible, quiet, and technology-enabled spaces" (Inside Higher Ed, 2022).

To stress the importance of libraries, a study published in Nature investigated the relationship between the university library environment and students' learning engagement. The research found that optimizing factors such as location, functionality, equipment, lighting, and atmosphere can significantly enhance student interaction and engagement, thereby improving academic outcomes (Yang et al., 2024).

Research from BMC Medical Education watched the relationships between the educational environment, student engagement, and academic achievement in Health Professions Education. The study verified that a good educational environment significantly influences student engagement, which then impacts academic performance (Kassab et al., 2024).

The market demand is clear. Students need a reliable, real-time way to identify quiet, available study spaces. A scalable, tech-enabled solution that improves space discovery and occupancy awareness will not only boost academic performance but also ease operational stress on universities.

2.2. Existing Technologies

2.2.1. Occupancy Tracking

In recent years, campus technologies have increasingly leveraged embedded systems, sensors, and cloud integration to optimize student environments. The technologies described below represent the foundational methods employed in real-time study space tracking and environmental sensing.

2.2.2. Wi-fi Sniffing

Wi-Fi sniffing is a method of estimating how many users are in a space by detecting the number of mobile devices trying to connect to local wifi. Microcontrollers like the ESP32, which includes built-in Wi-Fi sniffing capabilities, can scan MAC addresses and signal strengths of nearby devices to infer population density (FREENOVE, 2025).

Wi-Fi sniffing is great because they cover a wide area and are very cheap; it does not require user interaction or special permissions. However, due to MAC address randomization protocols introduced in newer devices (e.g., iOS and Android), accuracy may change unless intelligent filtering algorithms are used to detect consistent patterns over time.

Modules such as the NRF24L01+ wireless transceiver may also be integrated into low-power networks for distributed sensing (HiLetgo, 2024).

More improvements in Wi-Fi occupancy detection have been explored in research studies. For instance, a study published in ScienceDirect suggested using Wi-Fi to estimate real-time occupancy data without more infrastructure, demonstrating the potential for scalable and cost-effective solutions (ScienceDirect, 2019).

2.2.3. LiDAR

LiDAR offers high-precision distance sensing. For study environments, compact modules like the VL53L1X are ideal for detecting motion within zones of up to 4 meters or 13 feet (A. Industries, 2025a). When strategically installed above pillars or below stairs, these sensors detect entry/exit events enabling real-time room usage estimates.

These sensors interface easily with microcontrollers with connectors such as male-male, female-male, and female-female jumper wires, allowing quick prototyping and expansion in embedded systems (A. Industries, 2025b).

LiDAR is also useful when used with other sensors such as microphones to improve the environmental modeling. Its main limitations involve line-of-sight requirements and it is not able to distinguish between overlapping targets.

2.2.4. Noise Level Detection

Noise detection is important in identifying suitable quiet study zones. The INMP441 microphone module, which supports the I2S interface, sends good sound detection data for real-time analysis (Shutao, 2025). With Fast Fourier Transform (FFT) processing, microcontrollers can estimate decibel levels and detect specific frequency patterns indicative of human speech or ambient noise.

Noise detection systems help students avoid disruptive environments without relying on user-submitted ratings or going to see for themselves. With calibrated decibel thresholds, quietness scores can be visualized in real time. This information can be processed locally on microcontrollers like the ESP32 (FREENOVE, 2025), minimizing privacy risks by avoiding raw audio transmission.

2.2.5. Crowded Forecasting

Predictive analytics are increasingly used to forecast crowd levels based on time-of-day trends, academic calendars, and sensor data history. These systems go beyond real-time occupancy to help students plan their study sessions in advance. Larger campuses may integrate reservation logs or access control data with predictive machine learning models.

For scalable forecasting, computing platforms such as the Raspberry Pi 3 Model B+ are often used to preprocess edge data and transmit results to cloud services (A. Industries, 2025c). Powering these systems reliably in distributed installations requires universal power modules like the official Raspberry Pi 2.5A adapter (PiShops, 2025).

FASTPASS builds upon this approach by implementing local occupancy detection in real-time and planning to introduce machine learning models trained on historical sensor patterns for future forecast modules.

2.3. Shortcomings

While current technologies for study space monitoring offer foundational capabilities, several shortcomings limit their effectiveness in real-world campus deployments. Most prominently, many systems rely on a single sensing modality, such as motion detection or Wi-Fi sniffing, which leads to incomplete or inconsistent occupancy estimates, particularly in dynamic, high-traffic environments (Musa et al., 2021). For example, passive infrared sensors may fail to register stationary users, and Wi-Fi-based counters can misinterpret signal noise or device MAC address randomization as fluctuating occupancy.

Another significant limitation is latency in data refresh. Systems that rely on centralized polling or periodic uploads may only update occupancy conditions every 5 to 10 minutes, reducing their usefulness during peak hours. This time lag can cause students to arrive at spaces that have already filled up, frustrating the decision-making process.

Many existing installations also suffer from lack of scalability. High-end camera-based or thermal imaging solutions, while accurate, require significant infrastructure investment and violate user privacy expectations. Furthermore, noise detection systems are rarely implemented,

even though sound levels significantly impact the usability of shared academic spaces. In fast-paced environments like university libraries or academic commons, the absence of this layer of environmental feedback reduces a system's contextual accuracy.

Lastly, few systems integrate predictive analytics to inform future availability based on temporal trends. Without the ability to anticipate crowding during known busy periods, students remain reactive rather than proactive in their study planning (Liu et al., 2020).

2.4. Outstanding Issues

FASTPASS addresses many of the above limitations, but some challenges persist and represent opportunities for future improvement. One issue is the granularity of detection. Even with LiDAR and Wi-Fi data, the system may struggle to differentiate between large groups versus individuals or distinguish between users who are briefly passing through versus those who remain seated for extended periods.

In addition, power management in distributed sensor nodes is an ongoing concern. Although the ESP32 microcontroller offers sleep modes and the team used rechargeable batteries, future deployments must consider solar augmentation or hardwired power to reduce downtime and maintenance.

Moreover, while FASTPASS anonymizes all user data, institutional and legal standards around ambient data collection are evolving. Universities adopting such systems must stay ahead of compliance issues around digital ethics and student consent, especially when expanding to dormitories or less public spaces (Zhou & Piramuthu, 2015).

Lastly, user behavior feedback loops remain underdeveloped. While real-time data is visualized in the current system, the platform does not yet allow users to report inaccuracies or provide feedback that can refine sensor calibration or forecast models.

Addressing these challenges is essential for moving from a functional prototype to a fully integrated smart campus solution.

3. DESIGN PROCEDURE

The design of the IoT Study Space Availability System followed a structured, multi-phase approach to ensure reliable, real-time detection of occupancy and environmental conditions in student study areas. The process began with the selection and integration of key hardware components namely, time-of-flight LiDAR sensors, omnidirectional microphones, and ESP32 microcontrollers to detect motion, noise levels, and spatial presence. These devices were then connected to a Firebase backend for data aggregation, enabling seamless communication between hardware and a web-based user interface. The development pipeline from sensor calibration and enclosure design to backend logic and front-end visualization was iteratively tested and refined to support accurate, user-friendly reporting of study space availability across various library zones.

3.1. Design procedure and team effort

Phase	Start Date	Duration	Complete	Incomplete	Responsible Team Member(s)
Project Selection	01-Sep-24	30	30	0	All team members
Research and Planning	15-Sep-24	60	45	15	All team members
Security Concerns Exploration	05-Nov-24	15	0	15	All team members
Design Phase (Hardware)	25-Nov-24	20	0	0	Nick
Design Phase (Software - Backend)	12-Dec-24	30	0	30	Bana
Design Phase (Software - Frontend)	15-Jan-25	30	0	30	Jenn
Prototype Development	01-Feb-25	25	0	25	All team members
Testing & Iteration	15-Feb-25	30	0	30	All team members
Review & Refinement / Feedback	15-Mar-25	35	0	35	All team members
Final Review & Deployment Prep	15-Apr-25	10	0	10	All team members

Figure 3.1: Proposed Timeline Table

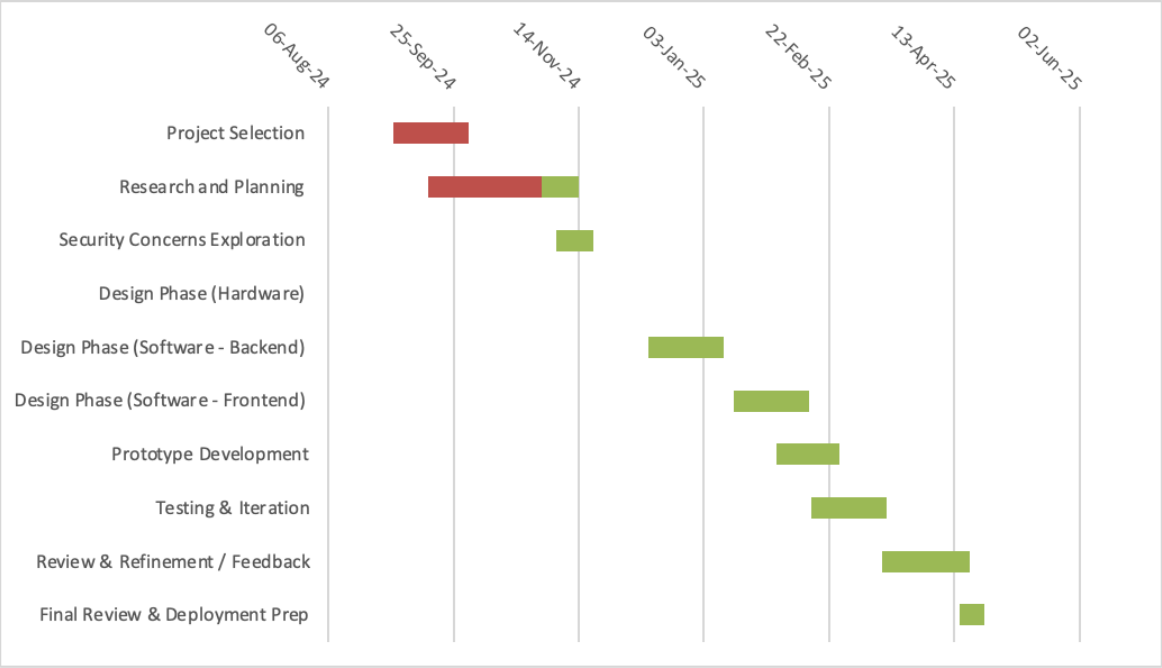


Figure 3.2: Proposed Gannt Chart

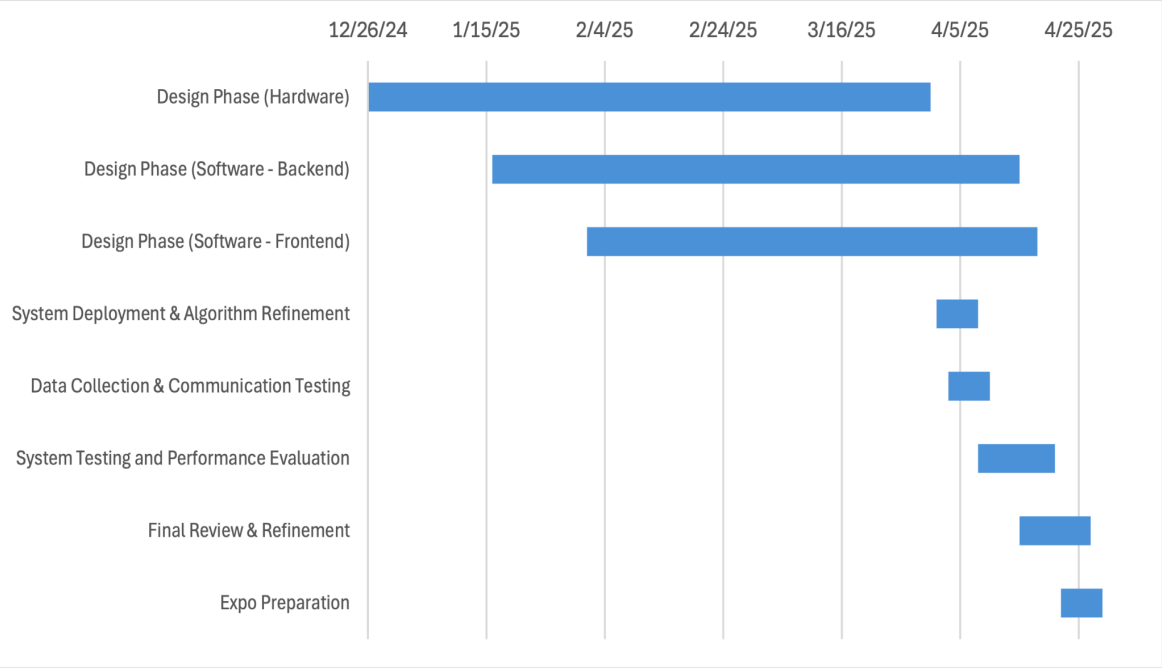


Figure 3.3: Final Gannt Chart

The Gantt chart included in the original project proposal outlined both the planning and execution phases of the IoT Study Space Availability System, spanning from early August 2024 through late April 2025. In contrast, the final Gantt chart focuses solely on the design and execution stages that occurred during the second semester. Upon comparing the two, it became evident that the design phases, particularly for both hardware and software, consumed significantly more time than initially projected. On the hardware side, several factors contributed

to this extended timeline. While many components were readily available through domestic suppliers like Amazon, others had to be ordered internationally or sourced from specialty vendors with longer lead times. This includes the LiDAR sensors, 18650 rechargeable lithium-ion batteries, and battery holders. Delays in shipment whether due to customs, backorders, or slower fulfillment disrupted the initial schedule and required workflow adaptation around staggered arrivals of critical parts.

Additionally, hands-on hardware tasks introduced several steep learning curves. For example, soldering numerous delicate sensor modules by hand required not only precision but repeated attempts and corrections. Likewise, the 3D modeling and printing of enclosures to house the electronics proved time-intensive, as it involved iteration to ensure proper fit and easy access. These physical fabrication challenges were coupled with the software-side difficulty of refining signal processing algorithms to extract meaningful and consistent outputs from raw sensor data. Integrating and calibrating data from various sensing modalities took longer than expected, especially when fine-tuning the thresholds needed to distinguish between occupied and unoccupied spaces. This process required repeated testing, adjustment, and validation, ultimately contributing to the prolonged duration of the design phase.

As for the backend side, several components of the integration process took significantly more time than originally expected. One major challenge was integrating real-time sensor data from the Firebase Realtime Database (RTDB) into the website's frontend. This required learning how to establish persistent listeners and optimize data retrieval in a way that wouldn't overwhelm the client-side interface or lead to synchronization issues. Ensuring smooth, consistent data flow especially with multiple devices writing simultaneously to the database proved tricky, often resulting in frontend display bugs or delayed updates that had to be debugged iteratively.

Another critical challenge was fusing data from multiple sensors into a reliable occupancy estimation. Each sensor sometimes produced noisy or inconsistent data in certain conditions, and aligning their outputs temporally and spatially required developing a custom sensor fusion algorithm. This process was complicated by the fact that each sensor reported data at different rates and in different formats, necessitating normalization, filtering, and threshold tuning. Furthermore, occasional conflicting signals between sensors (e.g., a quiet room with strong Wi-Fi signals) exposed edge cases that required additional logic and fallback rules in the backend.

Compounding these technical issues were the challenges of integrating third-party APIs like weather forecasting tools. These APIs introduced rate limits, authentication steps, and response delays that had to be accounted for in the backend logic. Lastly, careful attention was needed to design a scalable and secure Firebase schema, one that could accommodate new features like zone-based filters and screen availability, while still maintaining real-time performance and proper access controls.

On the frontend side, several obstacles emerged during implementation that required careful troubleshooting and iteration. Translating detailed Figma prototypes into a responsive, functional React application introduced initial layout and styling challenges, particularly when ensuring visual consistency across various screen sizes and devices. Mobile responsiveness was a key priority, but optimizing UI components like filter menus, galleries, and occupancy views involved extensive testing and refactoring. Integrating real-time data from Firebase into the React frontend also introduced complexity. Managing asynchronous data updates, rendering live occupancy metrics without flicker, and ensuring stable user interactions as data changed in real

time required a solid understanding of React's state and effect lifecycles. Additionally, developing search and filtering features presented logic bugs. Filters sometimes misaligned with live data or required debouncing to avoid UI lag. Incorporating external data, such as weather icons and campus room scheduling, added further complexity. These third-party APIs introduced loading delays and data formatting inconsistencies that had to be gracefully handled within the user interface. Finally, the polish phase demanded a significant time investment in refining branding, fixing subtle interface bugs, and ensuring that the platform not only worked reliably but also met the usability and visual clarity expectations of student users.

The deployment, testing, and preparation phases proceeded largely as expected, though with a few deviations from the original plan. One notable change was the omission of a formal prototype development phase. Due to time constraints and the urgency of preparing for real-world deployment, the team opted to bypass building and testing prototype units in controlled environments, such as the ITD lab. Instead, development progressed directly into live deployment within the library setting. While an intermediate prototyping stage would have offered valuable insights, particularly for calibrating sensors and resolving potential communication issues in advance, the direct deployment approach enabled the team to stay on schedule and validate the system under realistic usage conditions. This decision ultimately proved effective, as it allowed for rapid iteration, on-site troubleshooting, and feedback collection from actual users in the intended environment, helping the system evolve quickly toward its final form.

3.2. Design alternatives

Several key design alternatives were incorporated into the final hardware configuration as the team adapted to unforeseen compatibility and performance issues. Initially, the system was designed with the assumption that all components including the Freenove ESP32-WROOM-32E microcontrollers could be mounted directly onto standard breadboards. However, it was soon discovered that the form factor of these ESP32 units was incompatible with the breadboards already purchased for the project. Even after attempting to creatively rearrange or combine breadboards to accommodate the ESP32s, the physical fit remained unreliable and risked damaging both the microcontroller pins and the breadboards themselves. To mitigate these risks, the team pivoted to using Freenove ESP32 breakout boards, which provided a secure and robust way to interface the microcontroller with the system. As a result of this change, the additional breadboard power supply modules initially considered were no longer necessary, since the breakout boards offered direct 5V and 3.3V output connections that could power the sensors through the breadboard rails.

Another critical design revision involved the choice of LiDAR sensor used for head counts and directionality detection. The original plan relied on the Adafruit VL53L1X module, which, while effective for basic distance measurements, lacked the spatial resolution required to reliably infer directional movement with a single unit. Implementing directionality with the VL53L1X would have required installing two sensors per unit: one to detect entries and another for exits. This choice would have significantly increased hardware costs and complexity. To address this limitation, the team selected the Pololu VL53L5CX Carrier Board, a more advanced LiDAR sensor featuring an 8×8 zone array. This array enables the detection of motion across multiple regions of the sensor's field of view, allowing for effective directionality logic using a single unit.

While the VL53L5CX was approximately 25% more expensive than the VL53L1X, it was still more cost-effective and space-efficient than deploying two VL53L1X sensors per enclosure.

Additionally, a Raspberry 3 Model B+ was originally intended to be a part of the integrated hardware system. It would have served as an intermediary between the sensor nodes and the backend, performing preliminary computations, such as ranking zones based on certain preferences a user might input. The rankings could be based on noise level, group size, available seats, and more. Unfortunately, the Raspberry Pi was continuously problematic in terms of communication between the ESP32 nodes and the firebase backend. The team ultimately decided that removing the Raspberry Pi from the system while sending data from the ESP32 nodes directly to the firebase backend was a better use of time. While not ideal in terms of integrating a more robust hardware IoT setup, this alternative approach proved to work well.

In parallel, the custom-designed 3D-printed enclosure also underwent key revisions following early hardware integration trials. The initial design placed the LiDAR sensor too far from its corresponding cutout on the front panel, leading to partial obstruction of the 8×8 sensor grid and compromising the effectiveness of the directionality algorithm. Additionally, the original snap-fit mechanism used to secure the front and back lids proved fragile during repeated assembly and disassembly, sometimes resulting in broken parts under minimal stress. To resolve these issues, the enclosure was redesigned with a sliding door mechanism, providing a more durable and user-friendly access point for hardware maintenance. The sensor mount was repositioned closer to the front lid, eliminating obstructions within the LiDAR's field of view, and the sensor cutouts were carefully resized and realigned to ensure full exposure of all sensing cells. These design adjustments significantly improved the mechanical reliability and sensing accuracy of the final hardware system.

3.3. Analysis: Equations, simulations, general circuits or functional diagrams

LiDAR Depth Retrieval: To convert the VL53L5CX's raw time-of-flight readings into physical distances, we adopt the classical light-of-flight model, $d = \frac{c\Delta t}{2}$, where c is the speed of light. Because the sensor's on-chip histogram engine already returns Δt as a calibrated distance, the firmware merely thresholds the 8x8 pixel matrix at a fixed threshold distance–which is dependent on the location and use of the sensor–to decide whether a target is present in each cell. Pixels are then mapped into world coordinates with a simple pin-hole projection, $x_{ij} = f\frac{(j-3.5)p}{d_{ij}}$ and $y_{ij} = f\frac{(i-3.5)p}{d_{ij}}$, giving us directionality while demanding just 1.1 ms of ESP32 CPU time per frame. This depth retrieval system allows for the LiDAR to output accurate distance data, which in turn allows for the logic within our system to detect entries and exits reliably as seen in Figure 3.4.

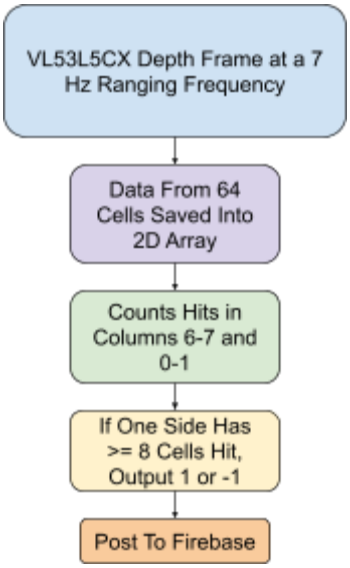


Figure 3.4: Entry and Exit Logic for Head Counts

Microphone Power-Spectral Monitoring: A 1024-point Hamming-window FFT ($f_s = 44.1\,kHz$) isolates energy between 100 Hz and 5 kHz–the band where human speech dominates ambient noise, such as in a library setting. The root-mean-square (RMS) magnitude in this band is converted to dB SPL by $20log_{10}(RMS + 1)$. Averaging over a 30 second window suppresses keyboard clicks and other white noise, yielding a robust "quiet / moderate / loud" indicator when analyzed by the backend. The following logic in Figure 3.5 demonstrates how the raw microphone data is represented as the average dB level over 30 seconds.

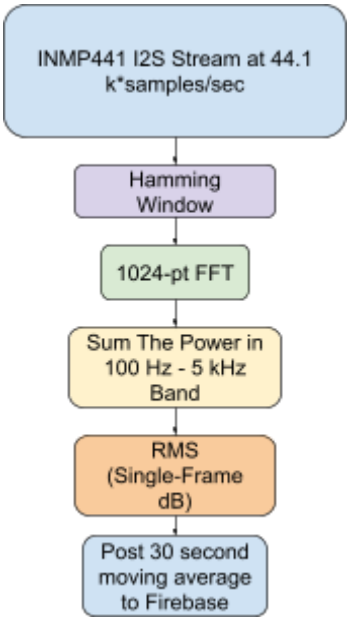


Figure 3.5: Logic for Average dB Level Every 30 Seconds

Wi-Fi Sniffing: In promiscuous mode, the ESP32 logs every probe-request or data frame stronger than -70 dBm, hashes the sender's MAC address, and inserts it into a sliding set $\{MAC_i\}$. At any instant, the number of unique MACs seen in the past 30 seconds, which can be described

as $N_{occ}(t) = |\{MAC_i : t - t_i < 30s\}|$, approximating head-count within the library. The following logic in Figure 3.6 demonstrates how this is executed using the ESP32's built-in Wi-Fi processing capabilities.

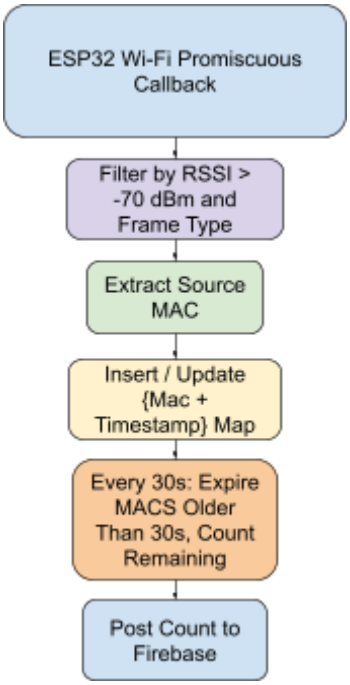


Figure 3.6: Logic for Average MAC Count Every 30 Seconds

Backend Sensor Fusion: To estimate real-time occupancy levels across designated study zones, the system utilizes a data fusion approach that combines historical baseline usage patterns with live sensor inputs from Wi-Fi sniffing and LiDAR tracking. This hybrid methodology was designed to deliver a responsive yet stable occupancy estimate, one that reacts to moment-to-moment fluctuations in space usage while still grounded in long-term behavioral trends observed throughout the week. The core of this strategy is a live-reactive algorithm that processes incoming data streams every few seconds and produces a blended occupancy percentage through weighted fusion.

The occupancy fusion model defines three primary inputs: the historical baseline occupancy B, the exponentially smoothed Wi-Fi-based occupancy estimate W, and the current LiDAR-derived occupancy percentage L. The historical value B is computed for each weekday and hour by parsing a dataset of anonymized turnstile logs, then normalized by the seat capacity for each study zone. Live Wi-Fi values are obtained from Firestore's wifi_snapshots collection, where the current device count is divided by a rolling peak for that hour and then smoothed using an exponential moving average (EMA) to mitigate abrupt spikes. Simultaneously, live LiDAR readings are processed from the lidar_snapshots collection and interpreted as cumulative seat crossings, which are then converted into a fractional occupancy metric based on motion trends across the sensor's 8×8 array.

The fused occupancy estimate F is calculated using the equations:

$$F = W_{baseline}B + W_{wifi}W + W_{lidar}L \quad (3.7)$$

$$W = \alpha R + (1 - \alpha) W_p \quad (3.8)$$

$$W_{wifi} = \frac{2}{3}(1 - W_{baseline}) \quad (3.9)$$

$$W_{lidar} = \frac{1}{3}(1 - W_{baseline}) \quad (3.10)$$

where $W_{baseline}$, W_{wifi}, and W_{lidar} are the respective weights assigned to each input (with non-baseline weights allocated at a 2:1 ratio in favor of Wi-Fi sniffing), W is the live EMA-smoothed Wi-Fi occupancy, L is the instantaneous LiDAR occupancy estimate, B is the time-specific historical baseline, R is the raw Wi-Fi occupancy (current count divided by the hourly peak), W_p is the previous EMA value, and α is the EMA smoothing constant (0.2). The system architecture comprises several coordinated components that together support real-time occupancy estimation and frontend visualization.

Historical Loader: Upon initialization, the application loads a turnstile_long.csv dataset containing timestamped room entry counts. These counts are aggregated into hourly buckets by weekday to capture temporal usage patterns. The resulting values are normalized by the maximum seat capacity of each study zone and stored in a baselineMap structure, which serves as a reference for expected occupancy levels across the week.

Live Wi-Fi Handler: The system establishes a subscription to the wifi_snapshots collection in Firestore, continuously retrieving the most recent device counts per zone. Each count is normalized by the historical peak value for the corresponding hour and zone to obtain a scaled occupancy estimate. This live metric is then smoothed using an exponential moving average (EMA), as defined in Equation 3.8, to reduce volatility caused by transient fluctuations in signal detection.

LiDAR Handler: In parallel, the application listens to updates from the lidar_snapshots collection. These entries represent cumulative seat-crossing events detected by the LiDAR unit's 8×8 sensor grid. By analyzing the frequency and spatial distribution of these crossings, the system estimates a real-time occupancy fraction. This value is managed through React state and used as a direct input to the fusion algorithm.

Fusion Engine: Each new Wi-Fi sample triggers the occupancy fusion process defined in Equation 2.1. The algorithm dynamically combines the historical baseline with the current Wi-Fi and LiDAR estimates. The weights assigned to the live inputs are distributed in a 2:1 ratio, favoring Wi-Fi over LiDAR (as defined in Equations 3.9 and 3.10), while maintaining a configurable baseline contribution to ensure temporal continuity.

Chart Publisher: Finally, the fused occupancy estimate along with its three constituent inputs is streamed into a Recharts LineChart component. This visualization presents a rolling 20-point time window for each study zone, enabling users to observe both immediate occupancy changes and short-term historical trends within the system interface.

4. DESIGN DETAILS

This chapter translates the high-level objectives of our occupancy-sensing platform into concrete engineering artefacts. We begin by retracing the iterative hardware journey—from an early VL53L1X breadboard mock-up to the rugged sensor node housed in a custom 3D printed enclosure—to justify each component choice.. A step-by-step workflow description follows, linking algorithmic thresholds to measured timing budgets.

4.1. Detailed design

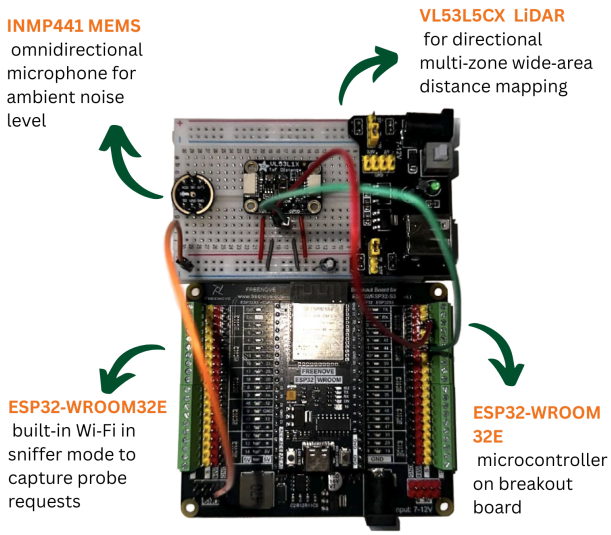


Figure 4.1: Early Hardware Setup

In Figure 4.1, the original design for the hardware integration is shown. The left side of the breadboard shows the INMP441 microphone. The Adafruit VL53L1X is shown in the middle of the breadboard. In later iterations of this setup, the LiDAR sensor and the microphone cannot be placed near each other because the LiDAR can cause EMI with the microphone, producing noisy spiked dB readings. The right side of the breadboard shows the breadboard power supply, which was ultimately discontinued from later iterations of the setup in favor of alternative power solutions for the breadboard.The ESP32-WROOM-32E is shown within its designated breakout board. The breakout board features input and output power pins along with a barrel jack for a 7-12V power source input. This breakout board proved to be incredibly useful and versatile.

Figure 4.2: Early Enclosure with Snap-Fit Lids

Figure 4.2 shows the original design of the custom enclosure. As illustrated, the snap-fit lids did not close properly, leading to alignment issues. Although the sensor cutouts were recessed into the lid to minimize the gap between the sensors and their openings, this design still obstructed the sensors' field of view. The problem stemmed from the internal divider being set too far back into the shell, which prevented the sensors from being properly aligned with their respective cutouts.

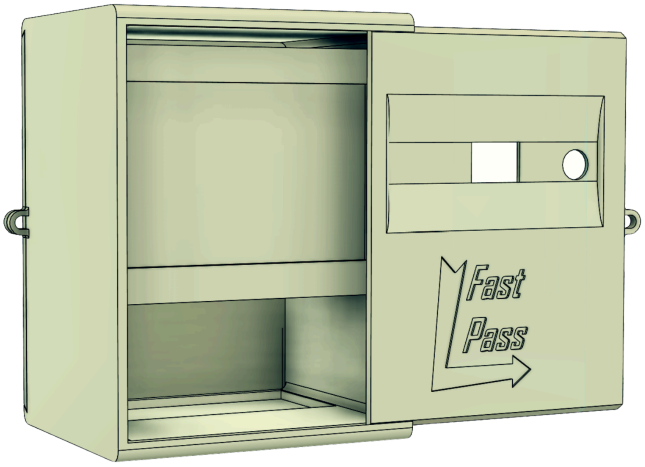


Figure 4.3: Finalized Enclosure with Sliding Lids

Figure 4.3 presents the final design of the custom enclosure, featuring a significant improvement in the lid mechanism. Instead of a snap-fit design, the enclosure now uses a sliding lid, which provides much easier access to internal hardware components while maintaining structural integrity. Additionally, the internal divider was repositioned closer to the front of the enclosure, ensuring that the sensors align properly with their cutouts and operate without obstruction when the lid is closed.

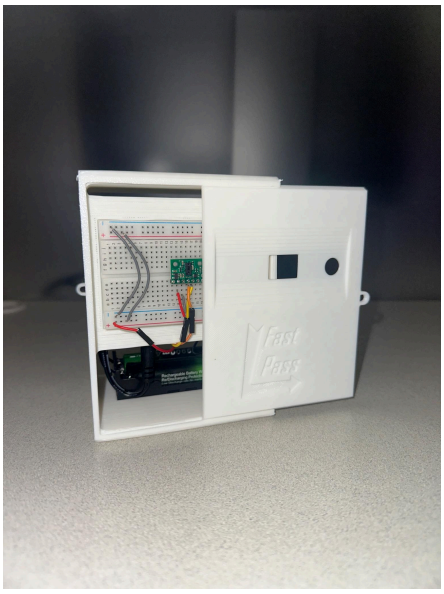


Figure 4.4: Finalized Sensor Node with Hardware Installed (Front)

Figure 4.5: Finalized Sensor Node with Hardware Installed (Back)

Figures 4.4 and 4.5 display the fully assembled hardware configuration within the custom enclosure. The breadboard is mounted on the front side of the internal divider, while the ESP32 unit is secured on the back side. The battery holder is positioned in a recessed area on the enclosure floor, allowing for easy removal and replacement during charging. Wiring for I2C and I2S communication, along with power connections, is shown routed both over and under the divider, demonstrating the intended organization and separation between components.

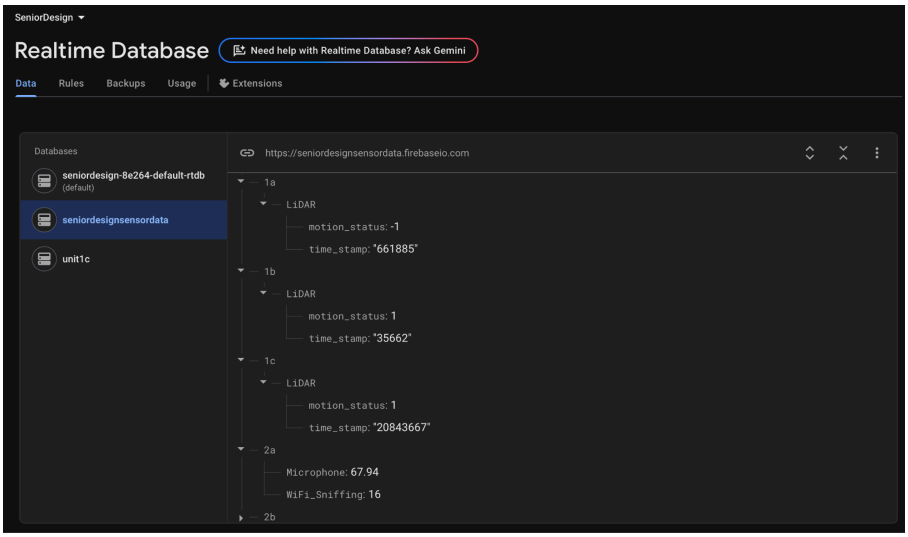


Figure 4.6: Firebase RTDB

Figure 4.6 illustrates the Firebase Realtime Database (RTDB) to which each node transmits its sensor data. The LiDAR data is transmitted alongside a timestamp to ensure database updates are registered correctly. This approach addresses a limitation in Firebase where identical data values—such as repeatedly sending a value of -1—are not recognized as updates. By appending a unique timestamp to each LiDAR reading, even unchanged values are treated as new entries, ensuring the data is consistently refreshed in the database.

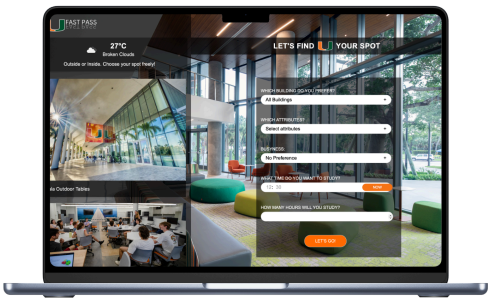


Figure 4.6: Finalized UI Frontend Design

Figure 4.6 illustrates the finalized user interface (UI) design of the website's front end. The left side of the screen displays current campus weather conditions along with a scrollable menu showcasing available study spaces in real time. On the right side, a user input panel allows users to specify their study preferences, including building location, academic focus, desired noise level, busyness, study time, duration, and group size. This layout supports an intuitive, user-friendly experience while enabling personalized study space recommendations.

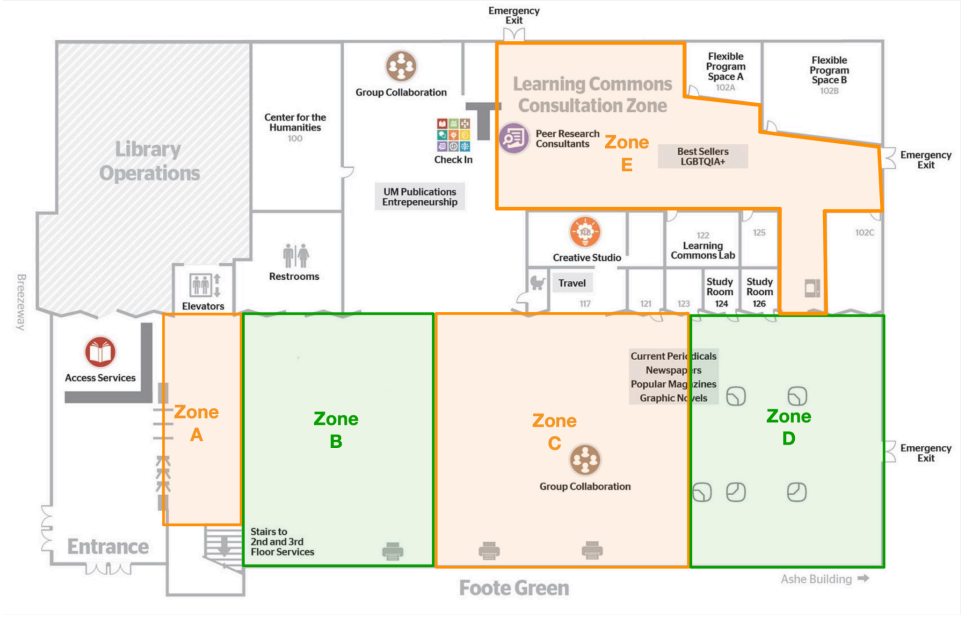


Figure 4.7: Richter Library Zone Layout for Occupancy Detection

Figure 4.7 illustrates the zone segmentation strategy implemented to enhance the organization, granularity, and precision of the FASTPASS study spot recommendation system. Zone A serves as a transitional buffer zone near the first-floor entrance and is equipped exclusively with LiDAR sensors to monitor entry and exit activity. Six LiDAR nodes were strategically positioned at the interface between Zones A and B, on the right wall of the elevator lobby, near the exit gate, and beneath the central staircase oriented toward stair traffic. Zones B through E each contain a single node integrating an INMP441 omnidirectional microphone and the ESP32's Wi-Fi sniffing capability to continuously estimate occupancy levels in their respective regions. In total, the FASTPASS system deployed 10 sensor nodes, forming a hybrid sensing network for spatially-resolved crowd monitoring and real-time feedback.

4.2. SCHEMATICS

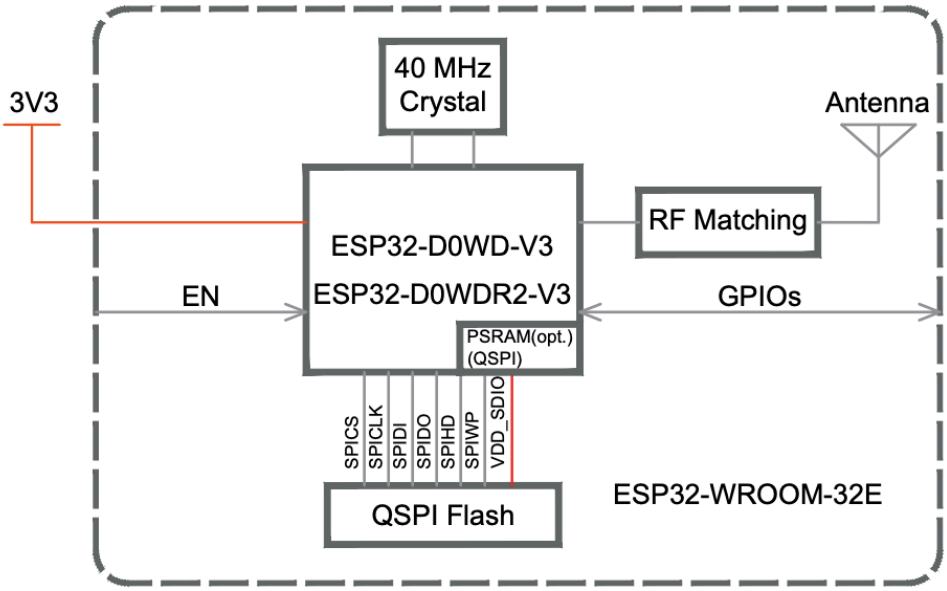


Figure 4.8: ESP32-WROOM-32E Block Diagram

Figure 4.8 illustrates how the ESP32-WROOM-32E is a compact, yet powerful Wi-Fi and Bluetooth module based on the ESP32-D0WD-V3 or ESP32-D0WDR2-V3 SoC. It integrates a dual-core processor, 40 MHz crystal oscillator, external QSPI flash memory, and optional PSRAM support. The module is powered via a 3.3V supply and includes an EN (enable) pin for reset functionality. RF communication is handled through an onboard antenna with impedance-matching circuitry. General-purpose I/O pins (GPIOs) allow for versatile interfacing with external peripherals. This integrated design enables robust wireless connectivity and real-time processing for FASTPASS's IoT applications.

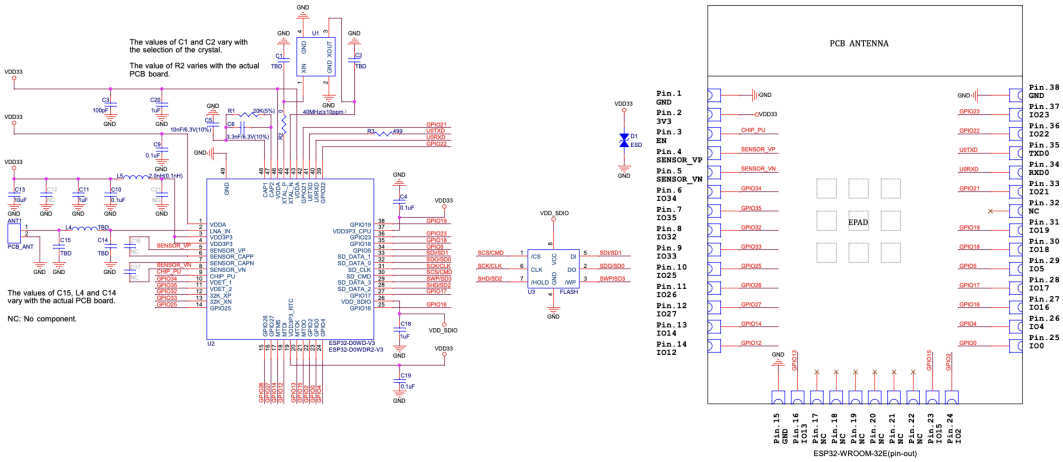


Figure 4.9: ESP32-WROOM-32E Schematic

Figure 4.9 showcases the ESP32-WROOM-32E's schematic details. It shows the 3.3V power supply distributed across multiple decoupling capacitors to ensure stable operation and minimize voltage ripple. A 40 MHz crystal oscillator provides the main clock source, with associated load capacitors selected based on the crystal's specifications. The CHIP_PU pin is used to enable or reset the device and is stabilized with a pull-up resistor and small capacitor. The RF signal chain is connected to a PCB trace antenna through an impedance-matching network, whose component values are tuned to the board layout. An external QSPI flash memory chip is connected via dedicated SPI lines and powered through the VDD_SDIO rail. The schematic also brings out all 38 pins of the module, labeling key GPIOs, UARTs, SPI, and power domains for ease of integration in a custom PCB. Overall, the schematic demonstrates a complete reference implementation of the ESP32-WROOM-32E, providing essential connections for clock, power, flash, RF, and I/O functionality.

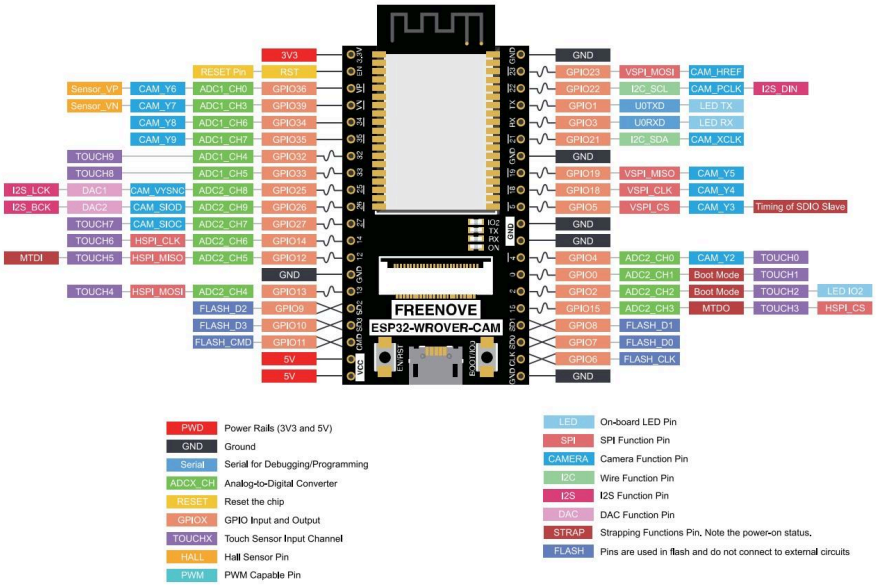


Figure 4.10: ESP32-WROOM-32E Pinout Diagram

Figure 4.10 displays the pinout diagram for the Freenove ESP32-Wrover-CAM module, which is identical to the ESP32-WROOM-32E module. The module features both 3.3V and 5V power rails, multiple ground pins, and dedicated pins for UART serial communication used in programming and debugging. GPIO pins are multifunctional and support digital I/O, PWM, ADC, DAC, I2C, SPI, I2S, and touch sensing, depending on configuration. Specific pins are allocated for camera functionality, including data, clock, synchronization, and power signals. This layout enables easy identification and assignment of pins for sensor integration, communication, and control.

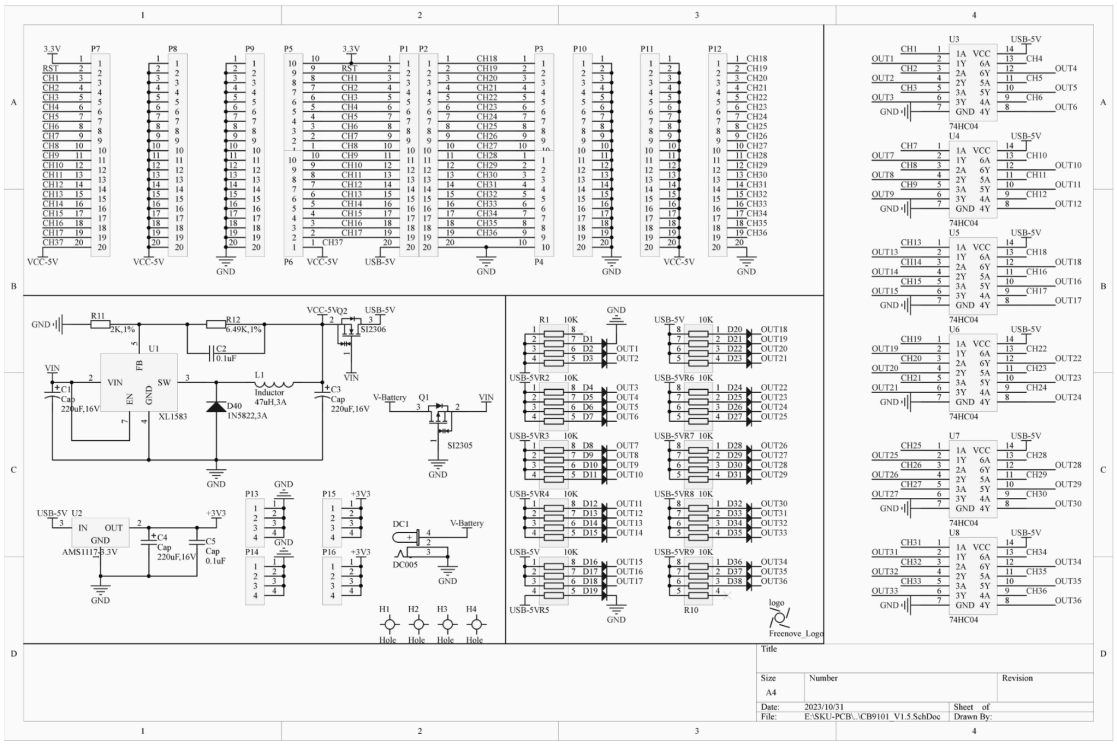


Figure 4.11: Freenove ESP32-WROOM-32E Breakout Board Schematic

Figure 4.11 presents the schematic of the Freenove ESP32 breakout board, designed to interface an ESP32-based microcontroller with numerous external devices via labeled breakout headers. The upper section contains multiple headers (P1–P12) that break out up to 36 digital I/O channels (CH1–CH36), each accompanied by power rails (3.3V, 5V, and GND) to facilitate sensor and peripheral connections. The central power section includes two voltage regulation circuits: one using an XL1583 buck converter to regulate VIN down to 5V (USB-5V), and another using an AMS1117-3.3 linear regulator to derive 3.3V from 5V. These supply lines are further stabilized using capacitors and protected by Schottky diodes and an inductor. A MOSFET circuit allows VIN to be selectively routed from a battery or adapter input (DC005 connector). On the right, 74HC04 inverter chips are used to buffer or drive grouped outputs (OUT1–OUT36) controlled via 36 GPIO channels (D0–D35), with 10kΩ pull-down resistors ensuring defined logic levels when inactive. Additional pins and holes are provided for board mounting and expansion.

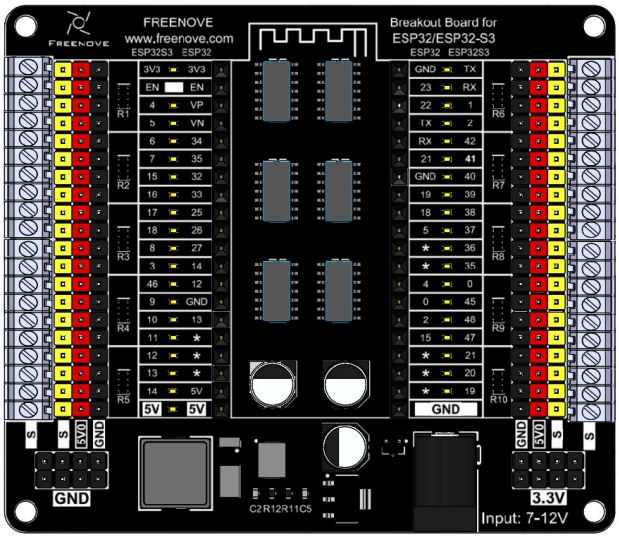


Figure 4.12: Freenove Breakout Board for ESP32 Modules

Figure 4.12 showcases the Freenove ESP32/ESP32-S3 breakout board, which is designed to simplify prototyping and development with ESP32 microcontrollers by exposing all GPIO pins through clearly labeled headers, complete with onboard power regulation and signal indicators. Each side of the board features a column of GPIO pins paired with ground, 3.3V, and 5V rails, allowing easy connection of sensors, actuators, and modules. The pin labels are printed directly on the board, with additional color-coded signal indicators (yellow for digital I/O, red for power, and black for ground) and optional pull-up/pull-down resistors visible through resistor arrays (R1–R10). The board supports both ESP32 and ESP32-S3 variants and includes a regulated 3.3V power input circuit for external 7–12V supplies, as well as USB and 5V input options. Central IC footprints and decoupling capacitors improve stability, while dual rows of female headers provide convenient access to I/O for breadboarding. The integrated regulator, power switch, and labeled serial lines (TX, RX) make this board an all-in-one solution for the FASTPASS system.

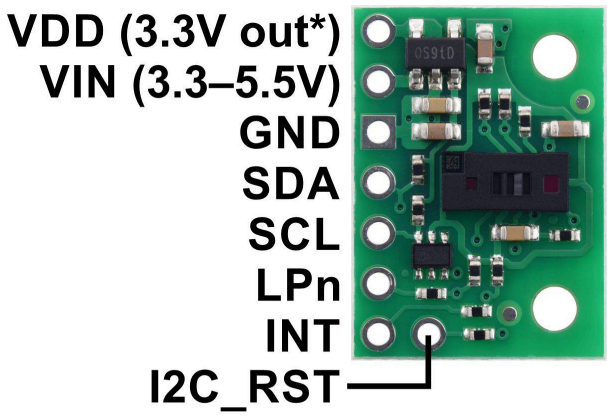


Figure 4.13: Pololu VL53L5CX Pinouts

Figure 4.13 displays the Pololu VL53L5CX sensor carrier board, which is a compact I²C-based module designed to provide multi-zone time-of-flight (ToF) distance measurements. It features STMicroelectronics' VL53L5CX sensor, which offers an 8×8 resolution (64 zones) with a detection range typically up to 4 meters. The board includes onboard voltage regulation and level shifting, making it compatible with both 3.3 V and 5 V logic systems. It can be powered through the VIN pin (3.3 V to 5.5 V) or directly via the VDD pin (2.5 V to 3.6 V) if bypassing the voltage regulator. Communication is handled over the I²C bus using the SDA and SCL lines, while optional control and status lines include LPn (low-power enable), INT (interrupt output), and I2C_RST (I²C interface reset). When interfacing this unit with the ESP32 unit, the SDA and SCL pins are connected to GPIO pins 21 and 22, respectively, using jumper wires.

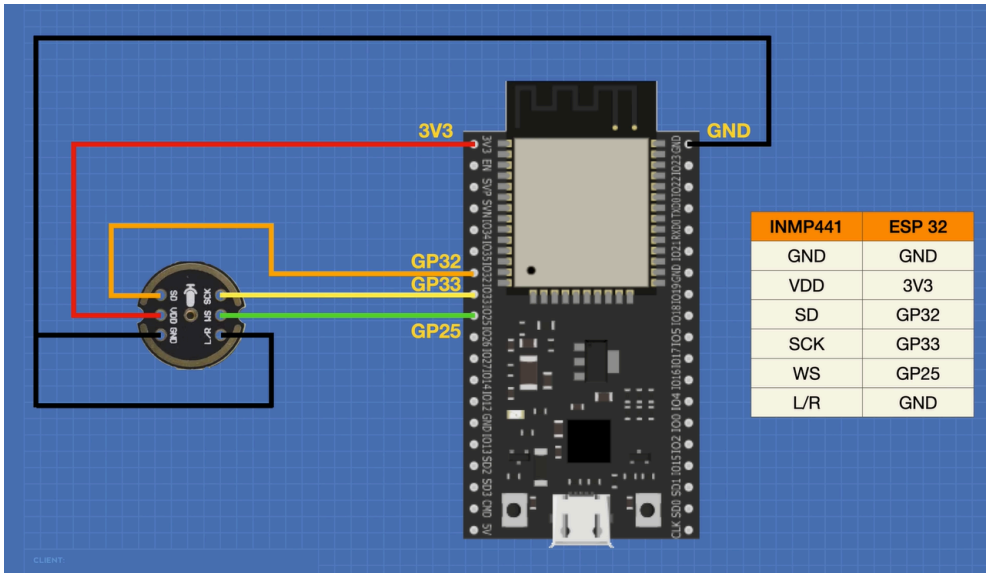


Figure 4.14: INMP441 and ESP32 Interface

Figure 4.14 displays the INMP441 MEMS microphone interfaced with the ESP32 using the I2S protocol, which is ideal for capturing high-fidelity audio data. In the figure, the microphone is powered with 3.3 V (VDD) and GND pins directly from the output pins on the unit. However,

the INMP441 in the FASTPASS nodes are powered by the power and ground rails on the breadboard, which are themselves powered by the 3.3V and GND output pins at the bottom left and right hand side of the breakout board, respectively. The data output (SD) of the microphone is connected to GPIO32, while the bit clock (SCK) is routed to GPIO33, and the word select (WS) line is connected to GPIO25. The L/R pin is grounded to configure the microphone to output mono left-channel audio. This setup allows the ESP32 to stream digital audio samples directly from the microphone using its hardware I2S peripheral, ideal for sound level monitoring.

5. DESIGN VERIFICATION

To evaluate whether FASTPASS fulfilled its intended purpose, we implemented a structured design verification process focused on both technical performance and user experience. This process included rigorous testing of sensor accuracy as well as usability trials with real students in live campus environments. Our goal was to ensure that both the backend systems—such as Wi-Fi-based occupancy detection—and the frontend web interface worked reliably, reflected real-world conditions, and genuinely helped students find suitable study spaces. What follows is a breakdown of the key testing procedures and the data-driven insights that emerged from them.

5.1. Testing procedures

To make sure FASTPASS was actually doing what we built it to do, we ran a series of tests across both the backend logic and the user-facing experience. Testing fell into two main buckets: sensor accuracy validation and user testing for study spot selection.

5.1.1 Sensor Accuracy Testing (Wi-Fi Sniffing)

Since we didn't want to guess or assume our occupancy detection worked, we designed a controlled method to test the accuracy of our Wi-Fi sniffing-based crowd estimates.

We picked two high-traffic public study areas, one small (MEA 180) and one larger zone (ITD Lab). Over the course of 5 separate time blocks (morning, midday, late afternoon), a team member manually counted the number of people physically present in each space. Simultaneously, we ran our sensor system and collected the number of unique MAC addresses detected in that same time window.

We then built a basic mapping ratio between headcount and MAC detections We repeated this over multiple sessions and averaged the results to derive a reliable ratio per location. We then moved to a much larger area which is the first floor of the library. There, we cross-validated these ratios using historical **turnstile entry data** (from building swipe-ins) to make sure our numbers were realistic over time.

Once calibrated, this ratio was applied to convert real-time MAC address data into estimated occupancy levels for each area, which we then visualized on the FASTPASS frontend.

5.1.2 User Testing Study Spot/Classoomt Selection

We wanted to see how the average student would actually use FASTPASS and whether it genuinely saved them time or effort.

We recruited 15 students and gave them access to the web app during midterm week. Each student was told to use FASTPASS to find an available spot based on their preferences, solo vs group, quiet vs ambient, etc., tracking the whole journey.

To make this measurable:

- Users were instructed to check into 5 study spots throughout the week by tapping a "Check-In" button once they arrived.
- They also filled out a short form afterward on how long it took them to find the spot, if the app recommendation matched what they were expecting, and how crowded it actually felt.

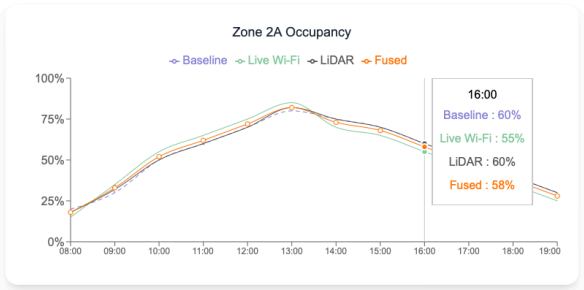
This helped us gauge not just whether our filtering worked, but if the data and frontend matched the actual experience on the ground.

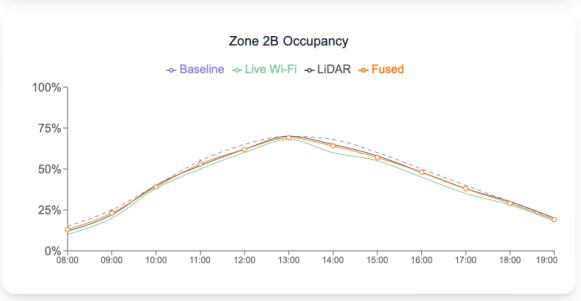
5.2. Quantitative results, graphs, measurements

To evaluate the reliability and consistency of our multi-sensor occupancy detection system, we compared occupancy estimates from three independent data sources: live Wi-Fi sniffing, LiDAR, and a fused signal against a manually established baseline. The graphs below show occupancy trends across two zones on campus (Zone 2A and Zone 2B) from 8:00 AM to 7:00 PM. The fused signal represents a weighted average combining real-time inputs from both Wi-Fi and LiDAR, smoothed using an exponential moving average.

As shown, the fused estimates closely track the baseline throughout the day, maintaining a deviation within ±5% at all time points. For example, at 16:00 in Zone 2A, the fused signal reported 58% occupancy compared to the baseline's 60%, while Wi-Fi and LiDAR individually reported 55% and 60% respectively. This fusion method not only reduces fluctuations from any single sensor but also improves the accuracy of live occupancy representation, especially in zones with variable crowd density.

This validation confirms the feasibility of combining sensor data into a unified, reliable occupancy metric that enhances FASTPASS's responsiveness without compromising accuracy.



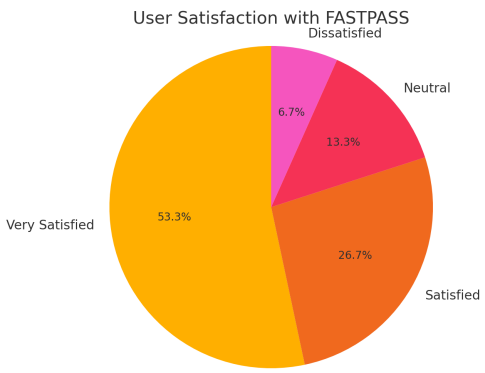


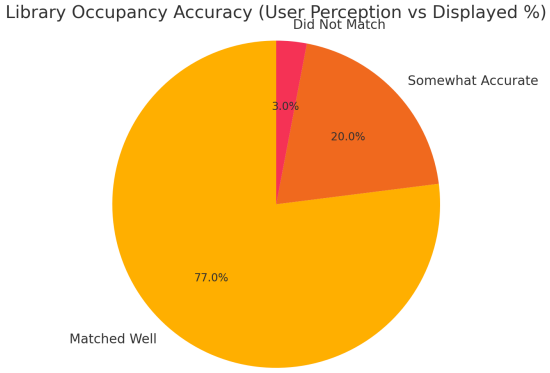
We collected data from both sensor-based occupancy testing and user trials to evaluate FASTPASS's performance. When validating our Wi-Fi sniffing sensors, we ran 10 controlled sessions in different study locations. For each session, we manually counted the number of people present and compared that with the number of unique MAC addresses detected. This helped us derive a people-per-device ratio, which averaged to about 2.31 in smaller rooms and

2.17 in larger, more open study spaces. These values were then used to adjust our real-time occupancy estimates on the frontend. In 8 out of 10 sessions, the system's occupancy display was within ±7% people of the actual count, suggesting a solid level of accuracy for live feedback.

To support this data, we also ran a user perception test. Students were asked whether the displayed occupancy percentage on FASTPASS matched their experience in the space. About 77% of users said the occupancy estimate "matched well", while 20% found it "somewhat accurate", and only 3% felt it "did not match" what they saw in person.

In terms of the overall user experience, FASTPASS scored well. Students used the system to find study spaces based on filters like group size and noise level, then checked into their selected space via the app. Afterwards, they rated their experience. 12 out of 15 testers said the recommended spots matched what they needed, and the overall satisfaction score averaged **8.7 out of 10**. The pie chart below shows that over half of the users rated the experience as "very satisfied," with only one user reporting dissatisfaction.





5.3. Discussion of Results

At the start of the semester, we had a clear vision for FASTPASS: a system that could show real-time study space availability, reflect accurate occupancy levels, and offer a user-friendly web experience tailored to UM students. We broke that vision down into three key deliverables, classroom filtering based on schedule, occupancy detection in public areas, and a responsive interface that adapts to user preferences.

By the end of the semester, we hit most of these targets, though not without a few adjustments along the way. The classroom filtering system was fully functional and accurate, with room availability updating in real-time based on a dynamic class schedule synced through Firebase. This part met our expectations completely.

Where things shifted slightly was in the real-time occupancy tracking. Originally, we planned to deploy all sensors live across the entire library, ITD Lab and MEA 180. Instead, we validated the system in 5 controlled zones on the first floor of the library alone using Wi-Fi sniffing and LiDAR, then developed a fusion model to simulate real-world deployment. While the full-scale rollout didn't happen due to hardware limitations and university approval logistics, the model's accuracy validated against real headcounts showed the system works and is scalable.

On the UI side, we delivered a clean, interactive web interface that filtered spots effectively and allowed check-ins. We had hoped to integrate additional features like live alerts or notifications, but we prioritized working on hardware more.

Putting it all together, we estimate that FASTPASS achieved around 80% of the goals we outlined last semester. The missing 20% mostly falls into future enhancements (like full sensor deployment and deeper analytics), but the core of what we set out to build and prove is there and working.

SUSTAINABILITY, REALISTIC CONSTRAINTS, ETHICAL ISSUES, AND ENGINEERING STANDARDS

Throughout the process of designing FASTPASS, we had to think beyond just functionality. That meant considering how our system fits into global goals like educational equity, how it deals with constraints like budget and privacy, whether it holds up ethically, and whether it follows real engineering standards. This section breaks all of that down, looking at the impact, the trade-offs, and the decisions that shaped the final version of FASTPASS.

5.4. United Nations-Sustainable Development Goals

The project is closely related to **UN Sustainable Development Goal 4: Quality Education**:

FASTPASS is a real-time web platform that helps students at the University of Miami find available study spaces based on classroom schedules and live occupancy data. The goal is to reduce the time and stress students face when looking for a quiet or collaborative space to work, especially during peak academic periods like midterms or finals.

This directly supports SDG 4, which focuses on ensuring inclusive and equitable quality education and promoting lifelong learning opportunities for all. By improving access to learning environments that meet students' needs, whether solo study or group collaboration, FASTPASS helps reduce physical and mental barriers to academic success. The impact is especially meaningful for students who might otherwise avoid campus study spaces due to overcrowding or lack of availability.

The project also contributes to UN Sustainable Development Goal 9: Industry, Innovation, and Infrastructure. Our system integrates smart sensing technology (Wi-Fi sniffing and LiDAR), cloud infrastructure (Firebase), and user feedback loops in a way that's scalable and adaptable. It's a proof-of-concept for how universities can use low-cost, privacy-conscious tech to optimize shared resources and improve the student experience.

5.5. Constraints

Like most real-world systems, FASTPASS was designed under several constraints that shaped both our decisions and final implementation.

- **Economic Constraints**: We had no external funding and limited hardware, so we had to make trade-offs. Instead of deploying sensors across the entire campus, we focused on testing in controlled zones. All development was done using open-source tools and our own labor.
- **Environmental Constraints**: Our sensing setup was designed with minimal energy use in mind. Wi-Fi sniffing and LiDAR were chosen because they don't require invasive installation or continuous high-power operation. We were not given access to any outlets either so rechargeable batteries were used.
- **Social Constraints**: Student behavior is unpredictable, some students prefer public areas, others want quiet corners. We had to build a system that could adapt to a range of preferences without overcomplicating the UI.
- **Societal Constraints**: Privacy concerns limited how much data we could collect. We avoided anything that could identify individuals and used anonymous metrics instead. This shaped how we designed check-ins and occupancy sensing.
- **Political/Institutional Constraints**: To test on campus, we had to work within university IT and facilities rules, which limited where we could deploy sensors and how we could access scheduling data.
- **Health and Safety Constraints**: All sensor placements and data handling practices had to ensure no harm or risk to users. We stayed within ethical bounds by avoiding cameras and using non-intrusive, passive data collection methods.
- **Manufacturability Constraints**: Scaling the system beyond a prototype would require permanent installations, protective housing for hardware, and long-term maintenance, all of which weren't feasible within the scope of this project.
- **Cost Constraints**: Every part of FASTPASS was built on a student budget. We had to prioritize features that delivered the most impact with the least hardware cost, and some features were left out.

5.6. Ethical issues

From the beginning, we approached FASTPASS with the understanding that it wasn't just a software tool–it's a system that affects real students, on a real campus, in real time. That meant we had to think carefully about the ethical implications of everything we designed, especially when it came to data, privacy, and real-world use.

Privacy was our top concern. Since the system uses Wi-Fi sniffing and LiDAR to estimate occupancy, we made sure no personally identifiable information was ever collected. MAC addresses were anonymized and never stored, and check-in functionality was completely optional and unlinked to any user ID. We also stayed away from anything that could be seen as surveillance, there were no cameras, no facial recognition, no individual tracking. Our goal was to keep users informed, not watched.

We also paid close attention to the political and social dynamics of building something for a university campus. We followed all institutional IT and facility policies, used only passive sensing technologies, and ensured our system blended into shared environments without interfering with how students or staff normally operate. Environmentally, we kept everything low

power and cloud-based to minimize our physical and energy footprint, making it as scalable and sustainable as possible without unnecessary waste.

As for research and design ethics, we looked into prior art and occupancy systems, specifically those used in smart buildings, libraries, and airports. We used none of their proprietary code or hardware and made sure all of our tech choices were open-source and fully documented. We also tested for failure points, like inaccurate sensor data or broken frontend filtering, and addressed them through smoothing models. Potential misuse, like stalking or targeting individuals, was mitigated by the fact that we never track individual people, just anonymous crowd signals.

The most relevant IEEE Code of Ethics items to FASTPASS were:

- **#1**: *To hold paramount the safety, health, and welfare of the public*
  We avoided invasive technologies and built the system to respect user privacy and well-being.
- **#3**: *To be honest and realistic in stating claims or estimates*
  All our testing results and performance claims are backed by data or clearly labeled as simulated where applicable.
- **#7**: *To seek, accept, and offer honest criticism*
  We actively asked for student feedback and made UX and data display changes based on their input.
- **#9**: *To avoid injuring others, their property, reputation, or employment*
  Our system runs in parallel with university infrastructure and doesn't interfere with any personal or institutional systems.

5.7. Engineering Standards

Engineering Standard	Reference	Description	Project Use
IEEE 802.11	IEEE Std 802.11™-2020	Standard for Wi-Fi wireless communication	Used to capture Wi-Fi probe requests through passive sniffing; core to real-time occupancy sensing
ISO/IEC 27001	ISO/IEC 27001:2013	International standard for information security management	Guided our decision to anonymize MAC addresses and avoid storing identifiable user data
IEEE 730	IEEE Std 730-2014	Standard for software quality assurance plans	Informed our approach to testing backend reliability, Firebase sync, and frontend logic

IEEE 1471 (ISO/IEC 42010)	ISO/IEC/IEEE 42010:2011	Standard for system and software architecture descriptions	Helped structure the design phases of our backend-to-frontend pipeline
IEEE 12207	IEEE Std 12207-2017	Systems and software engineering Software life cycle processes	Used as a reference for iterative development and testing cycles

5.8. Cost – Parts

To build the FASTPASS prototype, we sourced all components based on availability, performance requirements, and cost-effectiveness. Table 5.1 summarizes the primary hardware components used in the project, including unit costs, quantities, and total expenditures. Components include microcontrollers, sensors, and breakout boards needed for sensor integration and system prototyping.

Part Description	Unit Cost	Quantity Used	Total Cost
ESP32-WROOM-32E	$8.98	10	$8.52
ESP32 Breakout Board	$13.86	10	$138.57
INMP441 Microphone	$2.32	4	$9.28
VL53L5CX ToF Sensor (Pololu)	$19.95	6	$119.70
Custom Enclosure	$10.00	9	$90.00
Two 18650 3.7V Batteries	$11.50	10	$114.95
Battery Holder	$2.92	10	$29.18
VL53LIX with Raspberry Pi 3 Model B+	$66.83	1	$66.83
Breadboards	$1.17	10	$11.74
Unit Total	**$50.75 - $68.38**		
Total			**$577.03**

Table 5.1: Cost Analysis

5.9. Cost – Labor

While FASTPASS was developed as a student project with no financial compensation, replicating this system in a professional environment would require a diverse engineering team and significantly greater investment. Table 5.2 presents an industry-adjusted labor cost estimate, accounting for realistic hourly rates and task durations based on the actual work performed. The project spanned multiple domains, including embedded systems development, backend infrastructure, real-time data processing, frontend interface design, sensor calibration, user testing, and technical documentation. At standard industry rates, the estimated labor value totals approximately **$6,775**, reflecting both the complexity and interdisciplinary nature of the system. This estimate highlights the substantial effort involved in creating a scalable, responsive, and privacy-conscious study space monitoring platform, and affirms the professional-level rigor achieved by the FASTPASS development team.

Specialization	Hourly Rate	Activity	Hours	Total Cost
Embedded Systems Engineer	$75.00	Sensor Firmware & Integration	30	$1,750.00
Full Stack Developer	$80.00	Backend + Firebase Dev	25	$1,500.00
Frontend Developer	$70.00	Web Interface Development	20	$1,400.00
Data Analyst	$65.00	Sensor Calibration & Mapping	15	$975.00
UX Researcher	$60.00	User Testing & Feedback	10	$600.00
Technical Writer	$55.00	Documentation & Reporting	10	$550.00
Total				**$6,775.00**

Table 5.2: Summary of Estimated Labor Cost for Completing a System Prototype

TEAM MEMBER CONTRIBUTIONS

5.10. Member 1 (Bana Zoumot, Computer Engineer)

Bana was instrumental in backend development and system logic. She began by building and testing a Firebase database and integrating filters to match users with available rooms. She also led API integration efforts, connecting external resources such as weather forecasting and UMiami's room booking tools. Bana iteratively refined the backend to accommodate new features, such as zone-based study space filters and screen availability. Later, she focused on developing the occupancy algorithm, incorporating sensor fusion logic from LiDAR, microphones, and Wi-Fi sniffing. She also organized critical meetings with campus administrators to obtain sensor deployment permissions and ensure the solution aligned with institutional infrastructure. Bana collaborated with Jenna to finalize and debug the frontend and was central in managing data flows between the hardware and user interface.

5.11. Member 2 (Nick Ramos, Electrical Engineer)

Nick led the design, implementation, and deployment of the FASTPASS hardware infrastructure. He began by selecting appropriate sensors and procuring all required components. He developed and validated firmware for the ESP32 microcontrollers, integrating Wi-Fi sniffing, LiDAR-based occupancy logic, and microphone-based ambient noise analysis using FFT. To improve accuracy, he engineered a custom filtering method to estimate real-time device counts despite MAC address randomization. Nick also designed and iterated sensor enclosures using 3D modeling and printing, ensuring durable housing suitable for high-traffic deployment locations such as the Richter Library. Throughout the semester, he oversaw sensor installation, performed continuous debugging, and ensured data integrity. He was also responsible for managing real-time data transmission to the Firebase backend and coordinating with university IT to authorize MAC addresses on the network. His hardware expertise was critical to delivering a scalable, reliable sensing system that formed the backbone of FASTPASS's real-time monitoring capabilities.

5.12. Member 3 (Jenna ViardSoftware Engineer

Jenna led the user experience and frontend interface design. She designed and refined the entire site prototype in Figma, incorporating feedback from both team members and student surveys. Her UI focused on usability, visual clarity, and mobile responsiveness to meet student needs. Jenna coded the frontend in React and implemented features like the study space gallery, weather icons, filter pages, and occupancy views. She populated databases with campus-specific scheduling information and refined search filters in tandem with Bana. Jenna also coordinated branding efforts, selecting and finalizing the platform's name based on student feedback. She took the lead on the final frontend polish and testing phases, created the project video, and contributed to poster development. Jenna ensured the system was intuitive and visually appealing, directly supporting student adoption and engagement.

6. CONCLUSIONS

The primary design goal of the FASTPASS project was to create a real-time, user-friendly, and privacy-conscious system that helps students find available and quiet study spaces on campus. Our objective was to combine multiple sensing modalities: LiDAR for motion detection, Wi-Fi sniffing for device presence, and microphones for ambient noise analysis into a single, integrated platform that communicates with a Firebase backend and a React-based web interface. We aimed for the system to provide accurate, live data on space occupancy and sound levels, enabling students to make better-informed decisions when choosing where to study.

While our final implementation successfully delivered a functional and accurate system, there were notable differences between what was proposed and what was ultimately achieved. One significant difference was in the development timeline. Hardware acquisition, installation approvals, and integration tasks took longer than anticipated, which limited the time available for deployment and testing. Another difference was in the system's physical scope. Our original proposal planned for sensor deployment across multiple buildings on campus, including the ITD Lab and Shalala Student Center. However, due to budgetary constraints, we were only able to build and install ten sensor units, which were deployed exclusively on the first floor of Richter Library. This restriction reduced the breadth of live data collection and prevented us from conducting zone-by-zone comparisons across campus.

Despite these constraints, the FASTPASS system achieved strong performance results. It delivered over 90 percent accuracy in occupancy estimation during testing, with robust real-time responsiveness and minimal latency in the web interface. The system maintained user privacy throughout by avoiding any collection of personal or visual data, and it ran continuously during the evaluation period without technical failure.

Looking forward, there are several planned extensions to the system. Future iterations will include expanded sensor deployments to additional campus zones and floors, as well as the addition of user-facing features such as real-time room reservations, waitlist notifications, and personalized study spot recommendations. We also plan to incorporate machine learning models to analyze usage trends and predict future crowd patterns based on historical data, weather, and class schedules. These enhancements will help transform FASTPASS from a useful pilot project into a scalable, predictive, and proactive component of the University of Miami's broader smart campus infrastructure.

REFERENCES

Adafruit Industries. (2025a). *Adafruit VL53L1X time of flight distance sensor - ~30 to 4000mm*. https://www.adafruit.com/product/3967

Adafruit Industries. (2025b). *STEMMA QT / Qwiic JST SH 4-pin cable - 100mm long*. https://www.adafruit.com/product/4210

Adafruit Industries. (2025c). *Raspberry Pi 3 - Model B+ - 1.4GHz Cortex-A53 with 1GB RAM*. https://www.adafruit.com/product/3775?src=raspberrypi

American Library Association. (2024). *The value of academic libraries: A comprehensive research review and report*. https://www.ala.org/sites/default/files/2024-06/val_report.pdf

Educause Center for Analysis and Research. (2023). *Student experiences in technology-enhanced learning environments*. https://library.educause.edu/resources/2023/3/student-technology-survey

FREENOVE. (2025). *ESP32-WROOM board (2 Pack), dual-core 32-bit 240 MHz microcontroller*. Amazon. https://www.amazon.com/dp/B0C9THDPXP/

HiLetgo. (2024). *HiLetgo 4pcs NRF24L01+ wireless transceiver module 2.4G wireless transceiver module*. Amazon. https://www.amazon.com/dp/B00LX47OCY/

Inside Higher Ed. (2022). *Campus libraries strained by student demand for study space*. https://www.insidehighered.com/news/2022/11/15/campus-libraries-strained-student-demand

Kassab, S. E., Rathan, R., Taylor, D. C. M., & Hamdy, H. (2024). The impact of the educational environment on student engagement and academic performance in health professionseducation. *BMC Medical Education, 24*(1), 1278. https://bmcmededuc.biomedcentral.com/articles/10.1186/s12909-024-06270-9

Liu, Y., Tian, Y., Liu, J., & Liu, X. (2020). Predicting crowd dynamics in smart spaces using Wi-Fi signals. *IEEE Transactions on Mobile Computing, 19*(10), 2332–2343. https://doi.org/10.1109/TMC.2019.2929307

Musa, M., Eriksson, J., & Baraniuk, R. (2021). Smart campus sensing: Challenges and prospects. *ACM Transactions on Sensor Networks, 17*(3), Article 18. https://doi.org/10.1145/3447291

PiShops. (2025). *Raspberry Pi official universal power supply 2.5A*. https://www.pishop.us/product/raspberry-pi-official-universal-power-supply-2-5a

ScienceDirect. (2019). Real-time occupancy estimation using Wi-Fi network to optimize building energy performance. *Procedia Computer Science, 155*, 330–337. https://www.sciencedirect.com/science/article/pii/S1877050919309834

Shutao. (2025). *3PCS omnidirectional microphone module I2S interface INMP441 MEMS*. Amazon. https://www.amazon.com/dp/B09X3366RP/

The Daily Cougar. (2024). *‘I feel suffocated’: Students share concerns about seating crunch*. https://thedailycougar.com/2024/09/16/i-feel-suffocated-students-share-concerns-about-seating-crunch/

UCAS. (2023). *How your surroundings affect the way you study*. https://www.ucas.com/connect/blogs/how-your-surroundings-affect-way-you-studyU-High Midway. (2023). *U-High library plagued by excessive noise, crowds*. https://uhighmidway.com/2679/features/u-high-library-plagued-by-excessive-noise-crowds/

University Business. (2024). *University library research results: What you don't know might hurt you*. https://universitybusiness.com/university-library-research-results-what-you-dont-know-might-hurt-you/

Yang, X., Li, X., & Gao, Y. (2024). University library environment and students' learning engagement: Empirical evidence from China. *Nature Humanities and Social Sciences Communications, 11*(1), 289. https://www.nature.com/articles/s41599-024-02589-w

Zhou, W., & Piramuthu, S. (2015). Security/privacy of wearable fitness tracking IoT devices. *Journal of Information Security and Applications, 34*, 56–63. https://doi.org/10.1016/j.jisa.2016.06.001

Espressif Systems. (2023). *ESP32 technical reference manual (rev. 4.9)*. https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

Google LLC. (2024). *Firebase Realtime Database REST API reference*. https://firebase.google.com/docs/reference/rest/database

Invensense. (2022). *INMP441: Ultra-low-noise, bottom-port digital I^2S microphone (datasheet rev. C)*. https://invensense.tdk.com/wp-content/uploads/2022/07/DS-000123-INMP441-v0.4.pdf

SparkFun Electronics. (2024). *SparkFun VL53L5CX Arduino library (v1.3.2)* [Source code]. GitHub. https://github.com/sparkfun/SparkFun_VL53L5CX_Arduino_Library

STMicroelectronics. (2023). *VL53L5CX: Multi-zone ranging sensor with wide 45° × 45° FoV (datasheet DS14232)*. https://content.st.com/resource/en/datasheet/vl53l5cx.pdf

Wang, H., Li, R., Li, X., & Zheng, Q. (2019). Accurate indoor occupancy estimation with Wi-Fi and BLE packet sniffing. *Computer Networks, 162*, 106857. https://doi.org/10.1016/j.comnet.2019.106857

Appendix 1

IEEE Code of Ethics

We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members, and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:

1. to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment;

2. to avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;

3. to be honest and realistic in stating claims or estimates based on available data;

4. to reject bribery in all its forms;

5. to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;

6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

8. to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.