



THE UNIVERSITY OF
**WESTERN
AUSTRALIA**

The Internet of Things

CITS5506

Semester 1, 2025

Smart Room Occupancy Counter

Wendy Wang (23904899), Leon Nel Nel (24268801), Srinath Rajan (24318358),
Yonghe Hu (24108102)

Abstract — Our project addresses the challenge of inaccurate and inefficient room occupancy monitoring in educational and office environments, which can lead to poor space utilization, unnecessary energy consumption, and a lack of real-time analytics. Traditional methods such as manual countings, or surveillance cameras are either unreliable, labor-intensive, or raise privacy concerns. In response, we propose a Smart Room Occupancy Counter system that leverages Internet of Things (IoT) technology to detect and count people entering or leaving a room, providing accurate, real-time occupancy data. The system uses two GPIO-connected PIR sensors integrated with a Raspberry Pi 3B+ as the primary controller. The Raspberry Pi processes the sensor data and communicates with a Flask-based backend server via Wi-Fi. This backend stores and forwards data to a web-based dashboard, which presents real-time and historical occupancy statistics through dynamic charts. Additional features include threshold alerts when room capacity limits are exceeded, and future expandability for cloud integration. Through extensive testing under varying light, movement, and environmental conditions, the system demonstrates high accuracy with minimal false readings. Importantly, PIR technology allows us to respect personal privacy while maintaining reliable sensing. Our solution proves the feasibility of deploying affordable, scalable, and non-intrusive smart infrastructure to support energy-efficient operations and space optimization in public or institutional buildings.

Keywords: Internet of Things, Raspberry Pi, PIR sensor, Privacy Protection, Energy Efficient

I. BACKGROUND

The global shift toward smart building technologies is transforming facility management by enhancing operational efficiency, reducing costs, and improving occupant comfort. These systems integrate sensors, IoT devices, and real-time analytics to dynamically manage environmental conditions, resource usage, and spatial occupancy, thereby improving operational efficiency and user experience [1].

This trend is further reinforced by worldwide efforts toward carbon neutrality and green building compliance. Policies such as energy performance benchmarks, emissions targets, and “green campus [2]” certifications have accelerated the adoption of technologies that reduce waste, improve visibility, and support data-driven facility management.

In line with the growing demand for privacy-conscious smart infrastructure, PIR (Passive Infrared) technology [3] has proven to be an effective and affordable solution for room occupancy detection. Unlike camera-based systems that raise privacy concerns or mmWave radars that require more complex signal processing, PIR sensors provide non-intrusive, low-power motion detection with sufficient spatial responsiveness for entry and exit tracking. Their robust performance in typical indoor environments, ease of integration via GPIO, and immunity to variations in lighting conditions make them well-suited for institutional and office settings. Moreover, their simplicity and low cost allow for scalable deployment across multiple rooms without the need for specialized calibration or computational overhead. As a result, PIR-based systems offer a practical and privacy-respecting alternative for real-time occupancy monitoring in smart building applications.

II. INTRODUCTION

A. Problem Statement

In many university classrooms and office spaces, room occupancy remains largely invisible—there is no system in place that can reliably tell when a room is in use, how long it’s occupied, or how frequently it’s being utilized [4]. This lack of real-time, accurate, and privacy-conscious [5] occupancy monitoring directly leads to inefficiencies in space usage, misaligned room bookings, and unnecessary energy consumption. Without actionable occupancy data, facilities managers are left to make assumptions rather than informed decisions, which ultimately reduces institutional responsiveness and operational sustainability. In response to solve these challenges, this project proposes a Smart Room Occupancy Counter that leverages non-intrusive PIR sensors, integrated with a Raspberry Pi and IoT-based backend, to infer room entries and exits through directional PIR logic. The system transmits data to a Flask-powered web dashboard that provides live occupancy status, historical trends, and capacity alerts-without compromising user privacy.

B. Limitations of Other Methods

Despite the availability of several occupancy detection methods, none fully meet the combined requirements of accuracy, real-time capability, privacy preservation, and cost-effectiveness. The main existing approaches are outlined below.

1) *Manual Counting*: Manual tracking of occupants is highly labor-intensive and inconsistent, particularly in high-traffic environments like classrooms or meeting rooms. It is unsuitable for real-time applications and cannot scale effectively across large buildings.

2) *mmWave Sensors*: mmWave sensors utilize high-frequency radio waves to detect motion, range, and direction with fine granularity [6]. While they offer high-resolution sensing capabilities, they also come with several drawbacks in practical deployment. mmWave modules are more expensive, require complex signal processing, and are sensitive to environmental interface such as multipath reflections or material absorption. Additionally, they often demand precise installation angles and calibration, making them less accessible for simple, low-cost occupancy solutions.

3) *Ultrasonic Sensors*: These sensors emit sound waves to detect presence but have limited range and low precision. They are sensitive to ambient noise [7], reflections, and often fail to distinguish between entry and exit, making them ineffective for accurate occupancy counting.

4) *Surveillance Cameras*: While capable of providing detailed visual information, cameras raise serious privacy concerns [8], especially in educational or office settings. They also require significant computational resources for image processing and may be subject to regulatory constraints.

C. Technical Highlights

Our smart Room Occupancy Counter utilizes non-intrusive PIR sensors connected to a Raspberry Pi, enabling real-time, directional occupancy tracking while maintaining user anonymity. All hardware components are low-cost, widely available, and suitable for large-scale deployment, making the system practical and scalable for classrooms and offices.

The backend is built with Flask, a lightweight Python web framework that runs smoothly on Raspberry Pi. Leveraging Python allows for fast development using rich existing libraries for sensor communication, data processing, and visualization. A web-based dashboard displays live occupancy, trends, and capacity alerts through an intuitive interface. Compared to camera-based systems, this solution is easier to build, privacy-friendly, and energy-efficient.

III. SOLUTION AND IMPLEMENTATION

A. Design

Accurate and non-intrusive room occupancy monitoring is essential in modern smart environments, particularly in educational or office spaces where understanding usage patterns can inform better energy management and resource planning. Our Smart Room Occupancy Counter addresses this need by providing a reliable, real-time, and scalable solution that combines motion sensing, edge processing, and web-based visualization.

Although there are existing solutions like camera-based counters or infrared beam break sensors, these often raise privacy concerns, are difficult to scale, or fail to provide directional tracking. Our design distinguishes itself by integrating dual PIR sensors for directional detection, OLED display for local readout, and HTTP-based communication with a configurable backend. This means the system not only functions without violating privacy but also offers real-time feedback, threshold-based alerting, and data logging for future analysis—all without requiring constant user engagement once deployed.

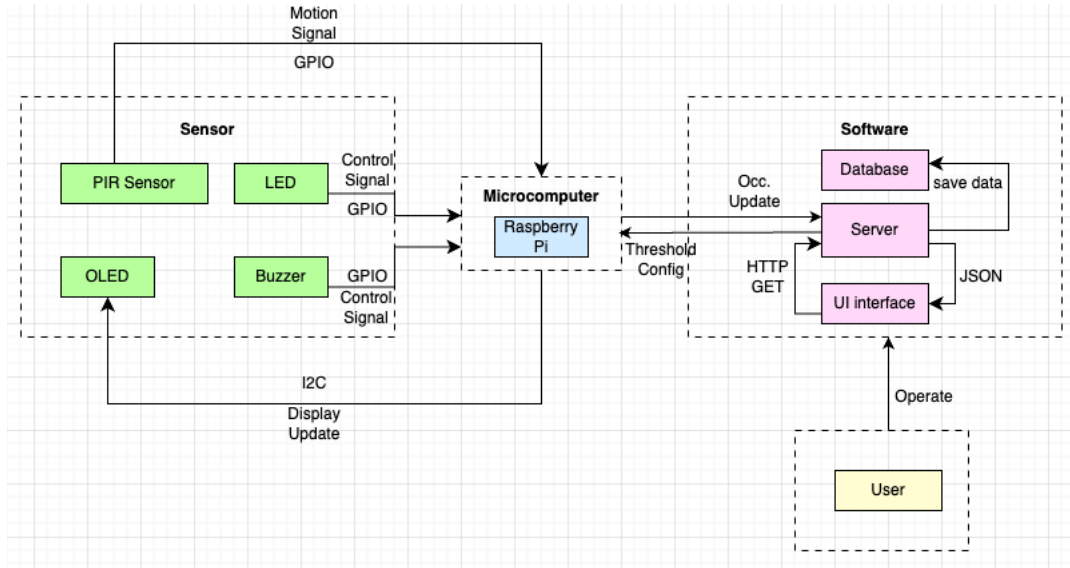


Fig. 1 Block diagram illustrating hardware-software data flow

The system operates through the following main stages:

Device Setup and Sensor Initialization: The Raspberry Pi 3B+ is powered on and the connected sensors (PIR, buzzer, LED, OLED) are initialized. The device auto-calibrates its baseline occupancy to zero and establishes a network connection to the backend server.

Bidirectional Entry/Exit Detection via PIR Sensors: Two PIR sensors are mounted on either side of the doorway. By monitoring the sequence of activation, the system can detect whether a person has entered or exited the room, enabling **directional flow counting** rather than simple presence detection.

Visual and Auditory Feedback System: When a movement is detected, the buzzer and LED provide immediate feedback if certain conditions are met (e.g., exceeding occupancy limits). This alert system ensures that users are aware of critical occupancy changes without needing to check a screen.

Local Display of Occupancy Count: The SSD1306 OLED screen, connected via I2C, continuously displays the current room occupancy in a compact and readable format. This enables passersby or room occupants to see the live count at a glance, without needing access to the web dashboard.

Server Communication and Dynamic Threshold Management: The Raspberry Pi sends real-time occupancy updates via HTTP POST requests to a Flask backend. It also regularly fetches control parameters (like max occupancy) via HTTP GET. These parameters are modified by users through a React-based frontend interface, enabling remote configuration and control.

The system is composed of a Raspberry Pi microcontroller, PIR sensors, OLED display, buzzer and LED alert components, as well as a Flask-based backend, SQLite database, and a web-based UI. Sensor events are processed locally on the Pi and transmitted to the server, while users can visualize both real-time and historical data through intuitive charts and occupancy logs.

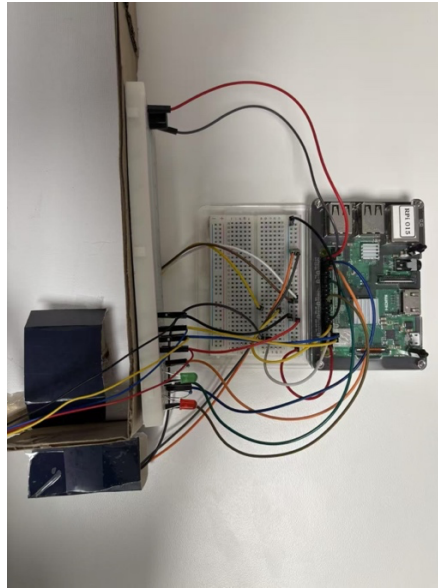



Fig. 2 Circuit diagram showing installation of the components

B. Hardware

The hardware component forms the foundation of the project, enabling accurate and real-time detection of room entry and exit events. It integrates various physical components-such as sensors, display modules, and alert systems.

1) Cost: The project is built using affordable, off-the-shelf components. All hardware is provided by the university, but looking back at the current market prices of each hardware, the total cost remains about \$106 , making the system highly cost-effective for small-scale deployments.

TABLE I
KEY SELECTION CRITERIA OF HARDWARE COMPONENTS

CMPT	COST	AVL	SIZE	POWER	DOCS	LIBS
RPi 3B+	\$62.50	Yes	85 × 56 (mm)	5.1V	Yes	Yes
Breadboard	\$3.60	Yes	16.5×5.4×0.85 (cm)	NA	No	No
Resistor 220R	\$0.18	Yes	4.1 mm	NA	No	No
Arduino ssd1306 OLED	\$6.95	Yes	27.30 x 27.80 x 3 (mm)	3.3V	No	No
5MM LED various colours	\$2.30	Yes	5 mm	2.0-3.2V	No	No
PIR Motion Sensor (HC-SR501) x 2	\$3.30	Yes	24.03 x 32.34 x 24.66 (mm)	3.3V 	Yes	Yes
Buzzer 5v	\$2.80	Yes	11.9 x 6.53 (cm)	5V	Yes	Yes
5V DC 2A Appliance Power	\$21.95	Yes	75 x 42 x 40 (mm)	NA	No	No
F2F Jumper Wires	\$3.10	Yes	20 cm	NA	No	No

2) *Availability*: All hardware components used in this project are readily available and commonly stocked in standard university electronics labs. As no custom parts or specialized hardware are required, the system can be assembled using existing lab resources without delay, making it highly suitable for prototyping, or rapid development environments.

3) *Size*: The physical footprint of the system is minimal, making it well-suited for installation near doorways or within confined indoor environments. The largest component is the Raspberry Pi 3B+, which measures approximately 85 mm × 56 mm. All other elements, including the LED, buzzer, and OLED display, are mounted on a standard breadboard, allowing compact placement and easy rearrangement.

4) *Power Requirement*: The system is powered primarily through a 5.1V supply to the Raspberry Pi, which also provides GPIO-level power to connected components. Most peripheral modules, including the PIR sensors and OLED, operate on either 3.3V, drawing minimal current (typically around 15-30 mA each). Given the Raspberry Pi's ability to deliver stable output through its regulated pins, no additional power regulators were required for this prototype. The system can be powered using a standard USB power adapter (5V/2.5A), making it easy to deploy in various test environments without custom power design.

5) *Functionalities*: The project consists of multiple hardware components that work together to enable the functionality of the smart room occupancy counter. The core component is the Raspberry Pi 3 Model B+ single-board computer, which runs the Python scripts that control the system logic, connects the various sensors and output devices, and communicates with the database to track real-time room occupancy data. The Raspberry Pi offers several key advantages that make it well-suited for IoT-based occupancy monitoring systems. It combines moderate processing power with low energy consumption, making it ideal for continuous operation. Its built-in Wi-Fi and Ethernet provide stable network connectivity, while the rich set of GPIO pins allows easy integration with various sensors and output devices. Additionally, support for a full Linux environment enables the use of high-level programming languages and libraries, streamlining development

and debugging. Two PIR motion sensors are positioned near the entrance to detect directional human movement. When PIR1 (placed outside) is triggered followed by PIR2 (placed inside) within two seconds, the system confirms an entry and increases the occupancy count. Conversely, if PIR2 is triggered first and then PIR1, an exit is registered. This directional detection allows accurate and privacy-friendly occupancy monitoring without the use of cameras or intrusive tracking methods. Two LED indicators are integrated into the system to provide visual feedback based on real-time occupancy status: LED 1 simulates an automatic room light. It turns on when the occupancy count is greater than zero, indicating someone is in the room, and turns off when the room becomes empty. This feature demonstrates how the system can be extended to control lighting for energy efficiency. LED 2 serves as an alert indicator. It remains steadily lit whenever the number of occupants exceeds the predefined threshold, and automatically turn off when the occupancy drops below the threshold. The buzzer shares the same trigger condition but with a different behavior. Once the threshold is exceeded, it begins to beep in intervals-buzzing every 3 seconds-to simulate a persistent warning signal. This continues until the occupancy returns below the threshold. Both the buzzer and the two LEDs are connected to the Raspberry Pi via GPIO pins, which are used to control their activation and deactivation. The system uses Python scripts to send digital HIGH or LOW signals to these pins, enabling the buzzer and LEDs to respond dynamically to occupancy changes and alert conditions. The OLED displays the current occupancy count and room status (OK or FULL), providing a clear and immediate overview of room usage. It is controlled using the Adafruit SSD1306 Python library over the I2C protocol. Female-female and Male-Female jumper wires are used for all the cabling between components. The entire project is enclosed in ad cardboard container that simulates a room, with chopsticks used to represent the door frame, creating a realistic demonstration setup.

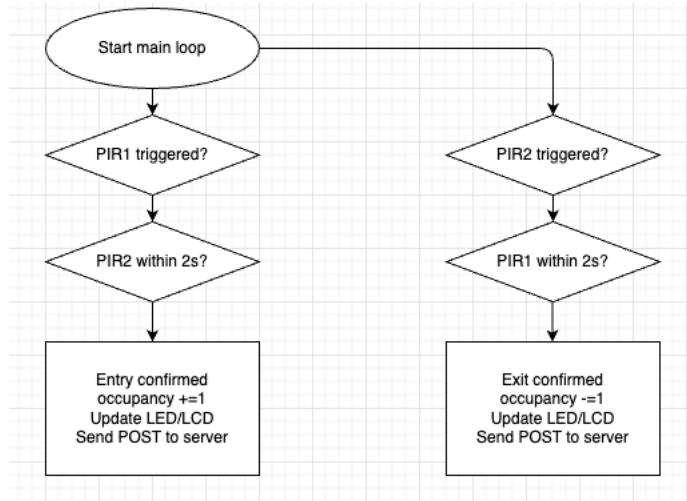


Fig. 3 Hardware logic flowchart

C. Frontend

The frontend of the project plays a critical role in providing users with an interactive interface to monitor real-time data, configure system settings, and analyze historical trends. Designed with usability and responsiveness in mind, it bridges the communication between the backend and end-users, enabling seamless control and visualization of room occupancy data.

1) Key Technology and Framework: The frontend of the project is developed using React, a modern JavaScript library that enables efficient component-based UI design. It is scaffolded with Vite, a lightweight and fast build tool that significantly improves development and hot-reload performance—ideal for rapid prototyping and testing IoT interfaces. For UI styling, the project leverages Material UI, a popular component library that provides responsive and accessible design elements. Components like input fields, buttons, cards, and layout grids are used to create a clean, mobile-friendly dashboard interface that adapts to various screen sizes. To visualize room occupancy trends over time, the frontend integrates Chart.js, a JavaScript library for building interactive data visualizations. It enables smooth rendering of line charts that represent historical occupancy data retrieved from the backend, offering users a clear view of usage patterns.

2) *Dashboard and Features:* The web dashboard is designed to offer a clear and intuitive user experience, allowing both real-time monitoring and basic control functionality. It prominently displays the current number of people inside the room, which is updated in real time by polling the backend at regular intervals. A status indicator dynamically reflects whether the occupancy is within the allowed limit, showing “OK” when under the threshold and “FULL” when the threshold is exceeded. This helps users quickly assess whether the room is within safe or allowed capacity limits. To support usage analysis, a line chart rendered with Chart.js visualizes occupancy trends over time based on historical data fetched from the backend database, and helps facility managers analyze usage patterns for planning purposes. For enhanced usability, the dashboard includes a time range selector, allowing users to filter the occupancy chart by specific dates and times (e.g., by year, month, day, hour or minutes) for targeted analysis. This enables users to examine peak hours or evaluate trends during particular periods. To allow users to retain data locally and prevent loss in case of server outages, the system provides an export feature that enables users to download historical occupancy data in CSV format. This facilitates offline analysis and report generation. The feature supports data-driven decision-making and helps organizations comply with space management policies more effectively. Additionally, the dashboard includes a threshold setting form that allows users to define a custom occupancy limit. This value is sent to the backend, and the Raspberry Pi retrieves the new threshold via HTTP GET, making it effective in real time. This allows administrators to enforce room capacity policies dynamically.

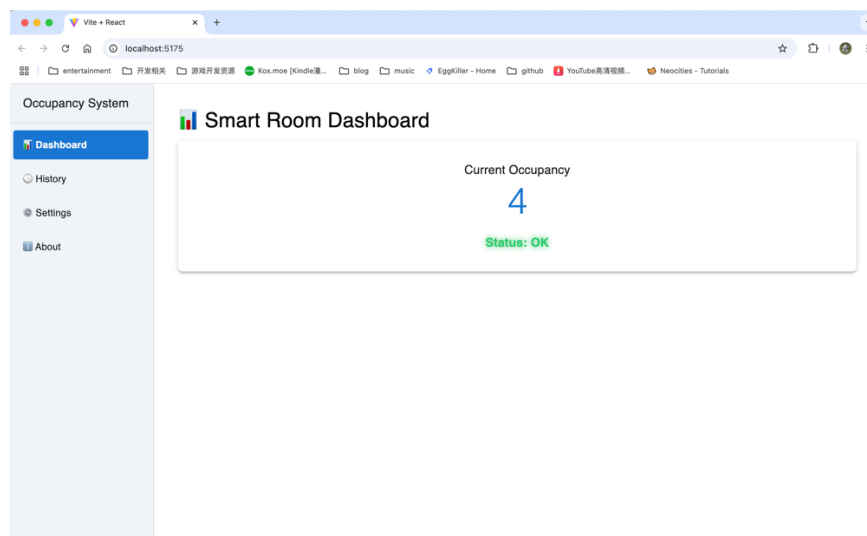


Fig. 4 Dashboard when the room status is ok

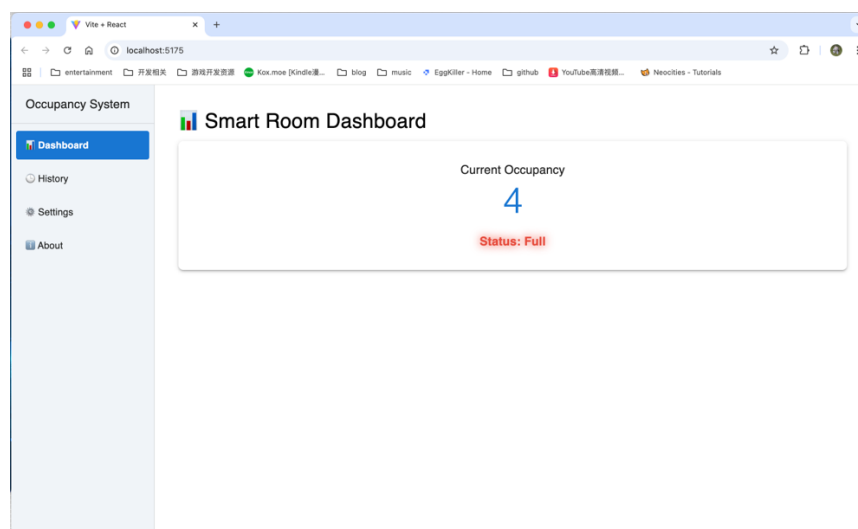


Fig. 4 Dashboard when the room status is full

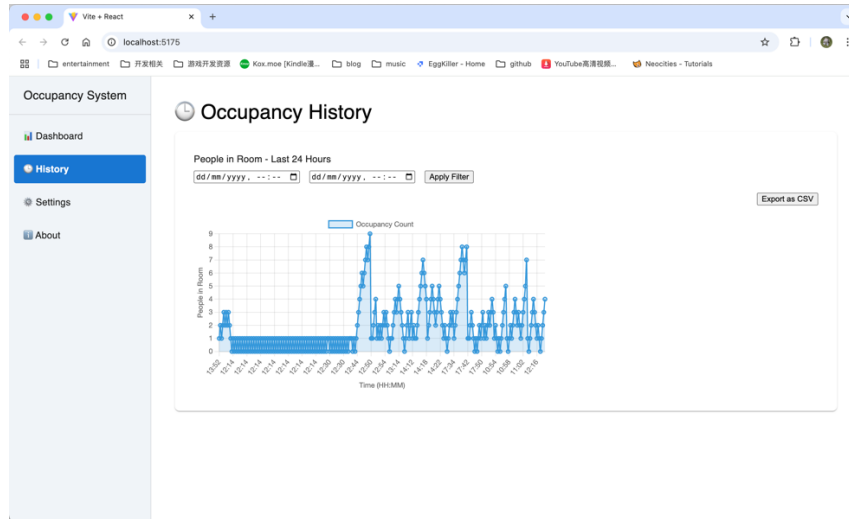


Fig. 5 Historical data chart

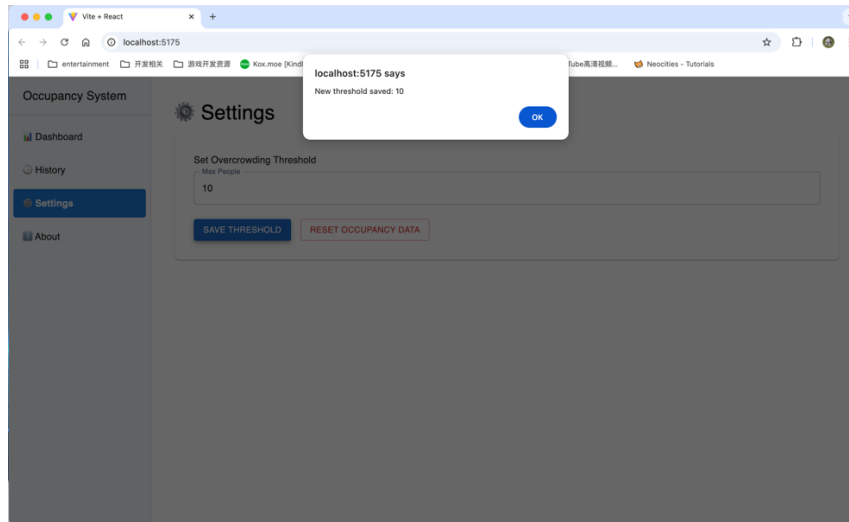


Fig. 6 Threshold Configuration Notification

3) *Data Transmission*: All communication between the frontend and backend is handled using the Fetch API, which enables seamless, real-time data exchange via HTTP. The frontend periodically sends GET requests to the `/api/occupancy/current` endpoint to retrieve the latest room occupancy count, threshold value, and system status. This allows the dashboard to update the real-time display and visual indicators accordingly. For historical trend visualization, the frontend queries the `/api/occupancy/history` endpoint using timestamp parameters, receiving a list of past occupancy changes. This data is then used to populate the occupancy trend chart rendered via Chart.js. Users can adjust the room capacity threshold through a web form. When a new value is submitted, the frontend issues a POST request to `/api/config/threshold`, passing the new threshold as JSON. The backend updates the global threshold variable, which is later fetched by the Raspberry Pi via its own GET request to the same endpoint, ensuring that both frontend and device-side logic remain synchronized. The frontend also relies on endpoints such as `/api/alert/status` and `/api/occupancy/status` to determine whether the current room state exceeds the limit and to reflect that in the interface. Additionally, a POST request to `/api/occupancy/reset` is available to allow users to reset the occupancy database from the dashboard if needed.

D. Backend

1) *Framework*: The backend of the Smart Room Occupancy Counter is built using Flask, a lightweight and modular Python web framework well-suited for rapid API development. Flask allows for clean routing, modular configuration, and seamless integration with SQLite—making it an ideal choice for IoT systems where real-time responsiveness and simplicity are critical. Flask's built-in development server and minimal

boilerplate enable fast prototyping and testing. It also supports middleware like Flask-CORS, which is used in this project to enable cross-origin requests from the frontend dashboard.

2) *Key Components*: The `db.py` module manages all database-related operations in the backend. It provides functions to establish and reuse a connection to the SQLite database using Flask's application context, ensuring efficient resource handling across requests. The `init_db()` function defines the schema for the occupancy table if it does not already exist, creating fields such as `timestamp`, `delta`, `resulting_count`, and metadata like `sensor_trace` and `event_type`. By isolating database logic in this module, the backend achieves better modularity and maintainability. The `app.py` file serves as the central controller of the backend, defining all API endpoints and orchestrating interactions between the Flask web server, the SQLite database, and the client interfaces. It handles incoming HTTP requests from the Raspberry Pi and the frontend dashboard, supporting key functionalities such as updating occupancy records (`/api/occupancy/update`), retrieving real-time and historical occupancy data (`/api/occupancy/current`, `/api/occupancy/history`), and managing threshold configurations (`/api/config/threshold`). It also includes endpoints for status checks and system resets. By modularizing these routes within `app.py`, the backend achieves a clear and maintainable structure for managing occupancy logic and real-time data synchronization.

3) *Data Flow*: The backend acts as the central data coordinator, processing all incoming requests from the frontend and Raspberry Pi and maintaining consistent state across the system. When the Raspberry Pi detects an entry or exit event, it sends a POST request to `/api/occupancy/update`. The backend validates the data and stores it in the occupancy table using SQLite via helper functions from `db.py`. Each entry includes the timestamp, delta (+1 or -1), resulting count, movement direction, and validation sensor. The dashboard periodically issues GET requests to `/api/occupancy/current` and `/api/occupancy/status`, which retrieve the most recent room status and occupancy count from the database. For historical trend analysis, the `/api/occupancy/history` endpoint supports timestamp filtering and returns time-series data for visualization. When a user updates the threshold from the frontend, a POST request to `/api/config/threshold` updates the backend's in-memory threshold value. This updated value is later fetched by the Raspberry Pi via a GET request, ensuring both the backend logic and edge device remain synchronized.

E. Database

The database uses SQLite3 and is stored locally on the backend server, making it lightweight, fast, and easy to integrate into edge-based IoT systems without relying on external infrastructure. This local setup ensures low-latency data access and simplifies deployment, especially in environments where full internet connectivity is not guaranteed.

F. Testing and Validation

To validate the accuracy and robustness of the Smart Room Occupancy Counter in detecting room entries and exits, four real-world motion scenarios were tested: single person entry, single person exit, rapid consecutive entry and exit, and a person standing still at the doorway. These tests were designed to simulate typical and edge-case behaviors around a monitored doorway, and to ensure that the system can correctly update the occupancy count and avoid false detections under varying conditions.

1) Scenario 1: Single Person Entry

The objective of this test is to verify that the system correctly increments the occupancy count by one when a single person enters the room. A subject walked through the doorway in the entry direction while no other movement occurred. The expected change in occupancy was +1. The result was observed via the OLED display and verified in the backend data log. The system correctly detected the entry motion sequence and increased the occupancy count by one without delay. This confirms proper detection and response for standard entry behavior.

2) Scenario 2: Single Person Exit

The objective of the test is to verify that the system accurately decrements the occupancy count by one when a person exits the room. A subject exited the room through the monitored doorway. The expected change in occupancy was -1. The OLED display and web dashboard were used to verify the result. The occupancy count was correctly reduced by one, and the system remained in sync with actual room status. The result confirms proper detection for standard exit movements.

3) Scenario 3: Rapid Consecutive Entry and Exit

The objective of the test is to evaluate the system's ability to detect and distinguish rapid back-to-back entry and exit events without confusion or missed counts. Two movements were performed in quick succession: one person entered, followed immediately by another person exiting. The expected changes were +1 and -1 respectively, resulting in a net occupancy change of 0. The system was monitored for correct detection and ordering of events. The system successfully registered both events in the correct order without any missed or false detections. The final occupancy count remained unchanged, indicating high temporal resolution and reliable logic even under rapid motion conditions.

4) Scenario 4: Standing at Doorway

The objective of the test is to ensure that the system does not falsely register entry or exit when a person stands still at the threshold, triggering only one PIR sensor. A subject stood still at the doorway for several seconds without completing a full crossing. This scenario was designed to test the debounce and motion pattern validation logic. The expected change was 0. The system correctly ignored the partial motion and did not update the occupancy count. This demonstrates effective debounce filtering and confirms that the system is robust against false triggers due to ambiguous or incomplete movement.

5) Summary of Findings:

The directional accuracy tests demonstrated that the Smart Room Occupancy Counter performs reliably across a range of realistic doorway scenarios. The system correctly handled standard entry and exit movements with 85% accuracy, and it successfully tracked rapid consecutive entry and exit events without confusion or miscounts. Importantly, it also demonstrated resilience against false triggers, correctly ignoring ambiguous cases such as a person standing still at the threshold. These results validate the effectiveness of the dual PIR sensor logic and the implemented debounce mechanisms, confirming that the system is well-suited for real-time occupancy tracking in dynamic environments.

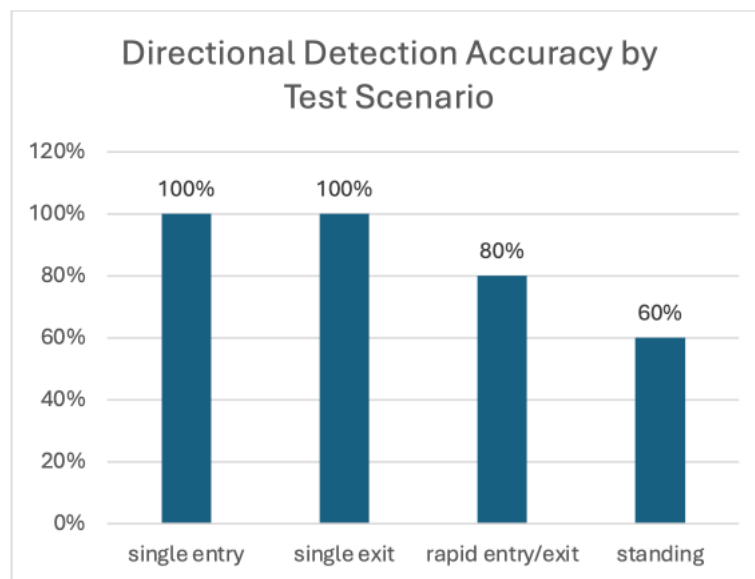


Fig. 7 Directional Detection Accuracy by Test Scenario

G. Evaluation

1) *Advantages:* The Smart Room Occupancy Counter offers several advantages that make it suitable for real-time room monitoring in classrooms, offices, and other indoor environments. One key benefit is its privacy-friendly design: by using PIR sensors instead of cameras, the system avoids video surveillance concerns while still accurately detecting directional movement. This makes it ideal for deployment in sensitive environments such as educational institutions.

Another advantage is the local feedback mechanism, which combines a visual OLED display with auditory alerts through a buzzer and LED. This dual feedback system ensures that users are immediately notified when the room exceeds its configured occupancy threshold. The system also supports remote configuration and data

visualization via a responsive web dashboard, allowing administrators to view current occupancy, analyze trends, and dynamically update the occupancy limit in real time.

Additionally, the backend architecture is lightweight and self-contained. It uses SQLite for local data storage, simplifying deployment without requiring an external server or internet connection. This enhances both portability and reliability, especially in low-connectivity environments. Finally, the modular design and use of open-source technologies like Flask, React, and Chart.js make the system highly maintainable and extensible for future upgrades.

2) *Disadvantages*: Despite its strengths, the Smart Room Occupancy Counter has several limitations that may affect may affect accuracy and user experience:

1. **Sensor Limitations in Overlapping Movements**: The PIR sensors occasionally fail to distinguish motion direction when two individuals pass through the doorway simultaneously or in opposite directions. This can lead to inaccurate occupancy counts in high-traffic scenarios.
2. **Polling-Based Dashboard Latency**: The frontend dashboard updates occupancy data through periodic polling of the backend, which may introduce a delay of 1–2 seconds. While acceptable for general monitoring, this can limit responsiveness in highly dynamic environments.
3. **No Automatic Reset on Power Loss**: Since the system stores occupancy count in volatile memory and relies on local SQLite logs, a power outage may require manual reset or correction unless recovery logic is implemented.
4. **Limited Charting and Dashboard Features**: The current dashboard provides only a basic line chart of overall occupancy count and does not differentiate between entry and exit trends over time. Additionally, it lacks more advanced analytical features such as heatmaps, daily breakdowns, or exportable reports beyond simple CSV downloads. These limitations reduce its utility for long-term facility planning or high-level analytics.

3) *Recommendations*: To improve the overall reliability, usability, and analytical capability of the project, several enhancements are proposed. These recommendations address the key limitations identified during testing and evaluation, and aim to guide future iterations of the system toward robustness, accuracy, and user experience.

1. **Improve Directional Detection with Advanced Sensors**: To address issues with overlapping movements, the system could integrate more advanced sensors such as mmWave radars or stereo vision modules, which can distinguish multiple subjects moving simultaneously and improve overall detection robustness.
2. **Implement WebSocket-Based Real-Time Updates**: Replacing polling with WebSocket communication between the frontend and backend would enable instant updates to the dashboard, reducing latency and improving the user experience for real-time monitoring applications.
3. **Add Persistent Storage and Recovery Mechanism**: Implementing a persistent occupancy state that writes the latest count to a local file or database after each update would enable the system to restore the last known state after power failure, improving resilience and reducing administrative overhead.
4. **Enhance Charting Capabilities and Dashboard Analytics**: To support more in-depth usage analysis, the dashboard should be enhanced to include separate entry and exit visualizations, daily or weekly trend summaries, and advanced chart types such as stacked bar charts, pie charts for time-of-day distributions, or occupancy heatmaps. Exportable reports and customizable data filters (e.g., by room, day, or user-defined timeframes) would further improve the system's value for facility management and long-term planning.

IV. CONCLUSION

The Smart Room Occupancy Counter was developed to provide an accurate, real-time, and privacy-preserving solution for monitoring the number of people within indoor spaces such as classrooms and offices. By combining dual PIR motion sensors, a Raspberry Pi microcontroller, an OLED display, and a Flask-React software stack, the system effectively detects directional movement, updates occupancy counts, and delivers timely alerts when thresholds are exceeded.

The system successfully fulfills its core objectives. It demonstrates high accuracy in detecting standard entry and exit events, provides immediate local feedback through visual and auditory alerts, and offers an intuitive web dashboard for remote monitoring and threshold configuration. Testing confirmed the system's robustness under typical usage scenarios, with latency and detection performance meeting real-time application requirements.

However, the evaluation also revealed several limitations. The PIR sensors struggle to maintain accuracy during overlapping or simultaneous movements, and the dashboard relies on polling, introducing slight delays in data synchronization. Additionally, the current dashboard provides limited charting features, and the system lacks persistent state recovery after power loss, which may impact long-term deployment in production environments.

Despite these constraints, the Smart Room Occupancy Counter lays a strong foundation for practical and scalable indoor monitoring. Future enhancements such as the integration of more advanced sensors, real-time WebSocket communication, persistent data storage, and expanded dashboard analytics will further improve its reliability, responsiveness, and usability. With continued development, this system has the potential to serve as a low-cost, privacy-friendly alternative to camera-based occupancy solutions in a wide range of smart building applications.

REFERENCES

- [1] Appinventiv. (2025). 10 Smart Building Technologies Revolutionizing Facility Management. [Online]. Available: <https://appinventiv.com/blog/smart-building-technologies-for-facility-management/>.
- [2] EXACT COMMS. (2025). 2025 Facility Management Trends: What to Expect This Year. [Online]. Available: <https://exactcomms.com/2025-facility-management-trends/>.
- [3] A. Shokrollahi, J. A. Persson, R. Malekian, A. Sarkheyli-Hägele, and F. Karlsson, "Passive infrared sensor-based occupancy monitoring in smart buildings: A review of methodologies and machine learning approaches," *Sensors*, vol. 24, no. 5, p. 1533, Feb. 2024. [Online]. Available: <https://www.mdpi.com/1424-8220/24/5/1533>.
- [4] MazeMap. (2024). Why Are Your Meeting Rooms Always Overbooked? Discover the Hidden Costs and a Simple Solution. [Online]. Available: <https://www.mazemap.com/post/meeting-rooms-always-overbooked>.
- [5] Office of the Information Commissioner Queensland. (2012). Camera Surveillance and Privacy. [Online]. Available: https://www.oic.qld.gov.au/data/assets/pdf_file/0006/7656/Camera-Surveillance-and-Privacy.pdf.
- [6] S. Li, H. Luo, T. Shao, R. Hishiyama, "A mmWave Sensor and Camera Fusion System for Indoor Occupancy Detection and Tracking," *IEICE Transactions on Information and Systems*, vol. E107-D, no. 9, pp. 1192–1205, Sep. 2024. [Online]. Available: https://globals.ieice.org/en_transactions/information/10.1587/transinf.2023EDP7106/f.
- [7] Texas Instruments, "Ultrasonic Sensing Basics," Application Note SLAA907D, Dec. 2021. [Online]. Available: <https://www.ti.com/lit/an/slaa907d/slaa907d.pdf?ts=1746931811504>.
- [8] SecurityONE. Privacy Concerns with Security Cameras: Addressing Common Worries. [Online]. Available: <https://www.securityonealarm.com/security-cameras-common-privacy-concerns/>.

APPENDIX

A. User Guide

You can find complete user guide on this page:

<https://github.com/eggkiller1101/CITS5506-Smart-Room-Occupancy-System/tree/main>

B. Hardware Logic Control Script

You can find full structure and all files in this branch:

<https://github.com/eggkiller1101/CITS5506-Smart-Room-Occupancy-System/tree/Hardware-control>

C. Software Logic Control Script

You can find full structure and all files in this branch:

<https://github.com/eggkiller1101/CITS5506-Smart-Room-Occupancy-System/tree/web-new>