CrowdSurf: Real-Time Crowd Monitoring for Indoor Spaces

Team #50
Johnathan Abraham, Tanvika Boyineni, Ananya Krishnan
ECE 445 Spring 2026

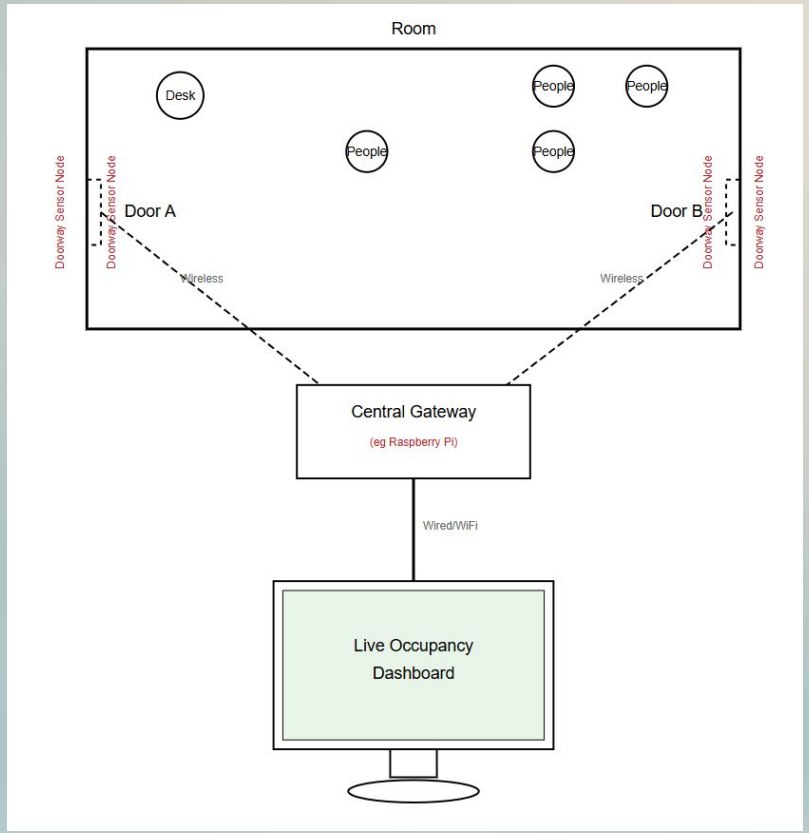
Introduction

Problem

- Indoor spaces like libraries, gyms, lounges, and student centers often get crowded during peak hours
- Students and staff usually do not have access to real-time occupancy information
- Existing solutions have major drawbacks:
  - Camera systems raise privacy concerns and require more computation
  - Manual or badge-based systems are labor-intensive and often inaccurate

Our Idea

- Build a low-cost, privacy-preserving, real-time occupancy monitoring system using doorway sensor nodes



Objective

Our system:

- Detects whether a person is entering or exiting through a doorway
- Tracks room occupancy in real time
- Displays occupancy and node health on a live dashboard
- Preserves privacy by using non-imaging sensors only

High-Level Requirements

- **Accuracy**: ≥ 90% correct IN/OUT directional classification per doorway under moderate, sequential pedestrian traffic
  - Systematic miscounts compound over time → occupancy estimate becomes useless
- **Latency**: Dashboard updates within 3 seconds of any crossing event (full pipeline: sensor → ESP32 → WiFi → Pi → browser)
  - A lagging dashboard misleads users about current state
- **Reliability**: Continuous operation ≥ 1 hour; automatic recovery from WiFi/MQTT packet loss without manual reset
  - A system that loses count on every dropout requires constant human supervision

System Design Overview

Two IR break-beam sensors are mounted at each doorway

ESP32 processes beam triggers and infers direction

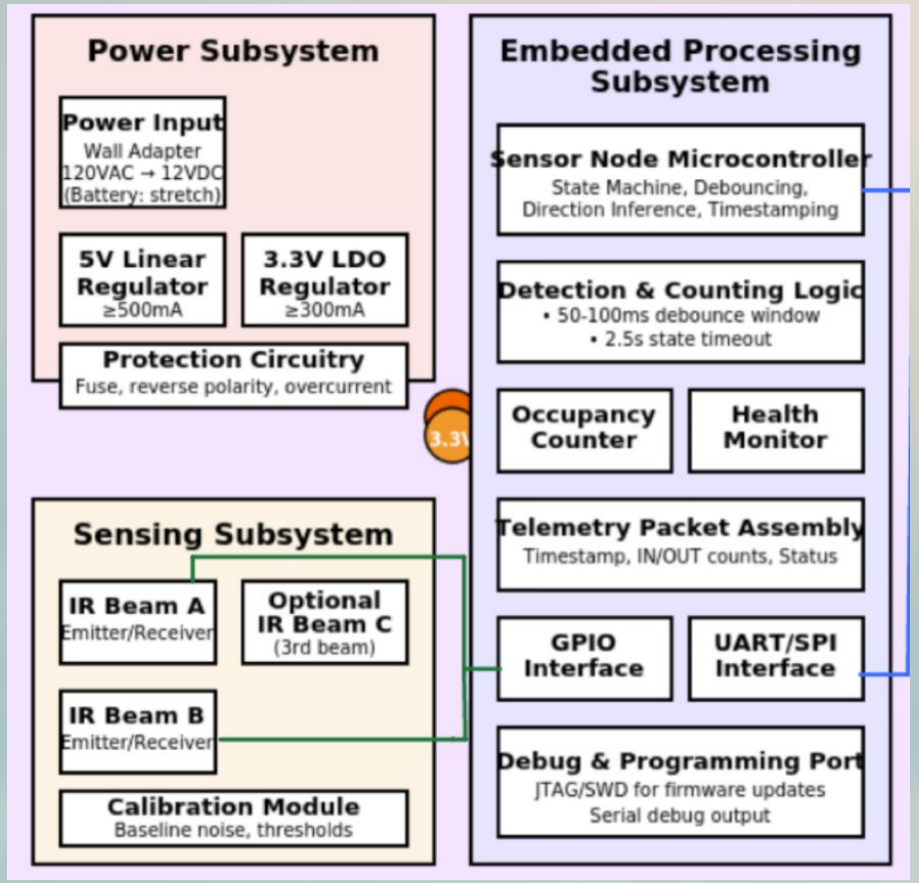
ESP32 publishes event packets over WiFi using MQTT

Raspberry Pi runs:

- Mosquitto broker
- gateway/data logging
- dashboard backend

React dashboard shows:

- live occupancy
- flow rate
- congestion status
- node health



Doorway Sensing + Physical Design

Doorway Sensing Design

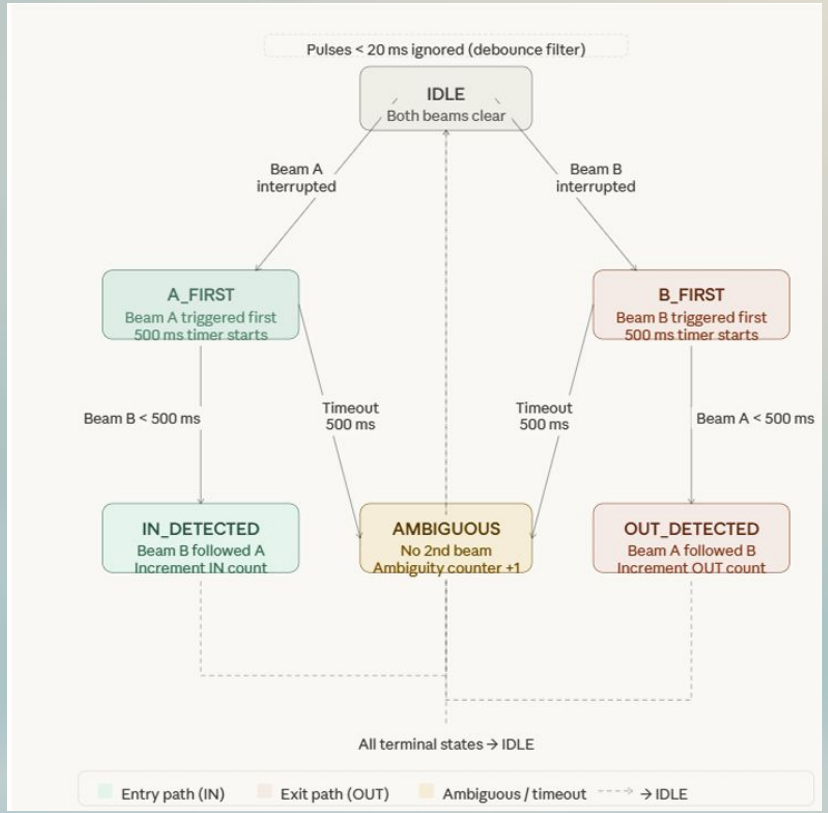
- Two Adafruit 2168 IR break-beam sensor pairs per doorway
- Mounted at about 90–100 cm from the floor
- Beam order determines direction:
  - Beam A then Beam B = IN
  - Beam B then Beam A = OUT

Why this matters

- No camera or biometrics
- Simple, fast, and privacy-preserving sensing method

Direction Inference FSM

- Implemented on ESP32-WROOM-32
- Uses interrupt-driven GPIO inputs from Beam A and Beam B
- Finite State Machine classifies events:
  - A first, then B within 500 ms → IN
  - B first, then A within 500 ms → OUT
  - No second trigger in time → AMBIGUOUS
- 20 ms debounce
- Local RAM buffer stores up to 10 unacknowledged events
- Heartbeat published every 2 seconds



Communication, Gateway, and Dashboard

Communication + Backend

- ESP32 publishes MQTT packets to:
  - node/[id]/events
  - node/[id]/heartbeat
- Event packet includes:
  - timestamp
  - node ID
  - sequence number
  - IN delta / OUT delta
  - local occupancy
  - status flags

Gateway Functions

- Raspberry Pi runs Mosquitto + Python gateway
- Computes occupancy:
  - **Occ(t+) = Occ(t) + Σ(in_delta - out_delta)**
- Logs telemetry to CSV
- Tracks node online/offline status
- Pushes live updates to dashboard using WebSocket

Dashboard

- Current occupancy
- Rolling flow rate
- Congestion color indicator
- Per-node health status

Tolerance Analysis

1. Direction-Inference Timing Analysis

The crossing delay between the two beams is:
$\Delta t = d/v$

2. Design Values

Nominal beam spacing: $d_{nom} = 0.15$ m

Spacing tolerance: $d_{min} = 0.14$ m , $d_{max} = 0.16$ m

Debounce threshold: $t_{db} = 20$ ms

FSM timeout: T = 500 ms

Worst-case fast crossing:

$\Delta t_{min} = d_{min} / v_{max} = 0.14 / 2.0 = 0.07$ s = 70 ms

70 ms > 20 ms = So a valid fast crossing is still above the debounce threshold

Worst-case slow crossing:

$\Delta t_{max} = d_{max} / v_{min} = 0.16 / 0.8 = 0.20$ s = 200 ms

200 ms < 500 ms = So a valid slow crossing still finishes within the FSM timeout window.

Nominal case:

$\Delta t_{nom} = 0.15 / 0.4 = 0.107$ s = 107 ms

20 ms < Δt < 500 ms = Across expected walking conditions, the inter beam delay stays above debounce and below timeout, which supports reliable IN/OUT classification

Tolerance Analysis

LDO power dissipation

P = (5.0 - 3.3)(0.5) = 0.85 W

This is below the approximate 1.5 W thermal limit of the AMS1117 package at room temperature.

GPIO protection voltage divider

$$V_{out} = V_{in} (R_2 / (R_1 + R_2))$$

Using R_1 = 10kΩ and R_2 = 20kΩ :

$$V_{out} = 5.0 \ (20 / (10 + 20)) = 3.33 \text{ V}$$

This reduces the 5 V sensor output to a safe ESP32 GPIO level while preserving a valid digital HIGH.

Verification

1. Direction Accuracy

- 20 IN trials + 20 OUT trials
- Pass criterion: **at least 18/20 correct in each direction**

2. End-to-End Latency

- Measure time from crossing event to MQTT broker receipt
- Dashboard must reflect updated occupancy within required time budget
- Beam response ≤ 5ms after physical interruption

3. Outage Recovery

- Simulate WiFi outage
- Buffer events locally
- Reconnect and verify:
  - no events lost
  - correct order preserved
  - sequence numbers increase properly

4. Reliability

- Heartbeat timing
- CSV logging
- continuous 60-minute operation test

5. Key equipment

- Saleae Logic 8 : beam response timing
- USB-to-UART adapter and serial terminal : FSM classification log
- 5V bench supply and current limit for safe bring-up

Conclusion

- CrowdSurf provides a privacy-preserving alternative to camera-based occupancy monitoring
- Converts doorway crossings into real-time room occupancy
- Combines:
  - IR sensing
  - ESP32 embedded processing
  - WiFi/MQTT communication
  - Raspberry Pi aggregation
  - live dashboard visualization

Future Work

- Improve handling of tailgating / simultaneous crossings
- Expand to more doorways and more rooms
- Refine congestion analytics and dashboard features
- Improve packaging and mounting robustness