CrowdSurf: Real-Time Crowd Monitoring System for Indoor Spaces

By

Jonathan Abraham

Tanvika Boyineni

Ananya Krishnan

Final Report for ECE 445, Senior Design, Spring 2026

TA: Aniket Chaterjee

6 May 2026

Project No. 50

Abstract

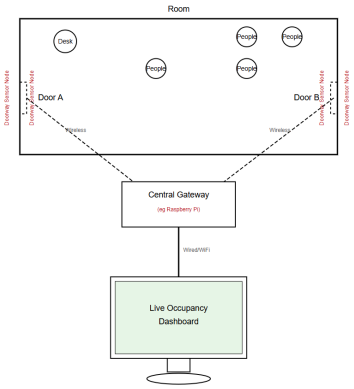
CrowdSurf is a real-time indoor occupancy monitoring system using distributed IR break-beam sensor nodes and a centralized MQTT gateway. Each node pairs two Adafruit 2168 IR sensors on a custom ESP32-WROOM-32 PCB to classify doorway crossings as entry or exit via a finite state machine; events are published over WiFi/MQTT to a Mosquitto broker on a Raspberry Pi, which aggregates counts and serves a React web dashboard. The system achieved 95% directional classification accuracy (requirement: ≥90%), a maximum end-to-end latency of 1.6 s (requirement: ≤3 s), and 62 minutes of continuous uptime with full recovery of all buffered events following a 2-minute simulated WiFi outage (requirement: ≥60 min, zero data loss). All 28 subsystem requirements were verified and passed.

Contents

1. Introduction

1.1 Purpose

Indoor public spaces such as university libraries, study lounges, and gyms routinely experience congestion during peak hours, yet facility staff lack real-time, room-level data on occupancy and directional traffic flow. Existing solutions are either camera-based, raising privacy concerns and requiring significant computational infrastructure or manual and badge-based, which are labor-intensive and yield only coarse estimates. CrowdSurf addresses this gap with a low-cost, privacy-preserving occupancy monitoring system that uses IR break-beam sensors at doorways to infer entry and exit direction without capturing images or biometric data.



Each doorway node pairs two Adafruit 2168 IR sensors on a custom ESP32-WROOM-32 PCB. A direction-inference finite state machine classifies crossings as IN or OUT; events are published over WiFi via MQTT to a Mosquitto broker on a Raspberry Pi, which maintains a live room occupancy estimate and serves a React web dashboard. The system collects only anonymous, aggregate counts at every stage of the pipeline. Chapter 2 describes the design of each subsystem and the hardware and software implementation. Chapter 3 presents verification results against all requirements. Chapter 4 provides a cost analysis. Chapter 5 presents conclusions, limitations, ethical considerations, and future work. The system achieved 95% directional classification accuracy, a maximum end-to-end latency of 1.6 s, and 62 minutes of continuous uptime with lossless event recovery following a simulated WiFi outage, all three high-level requirements satisfied.

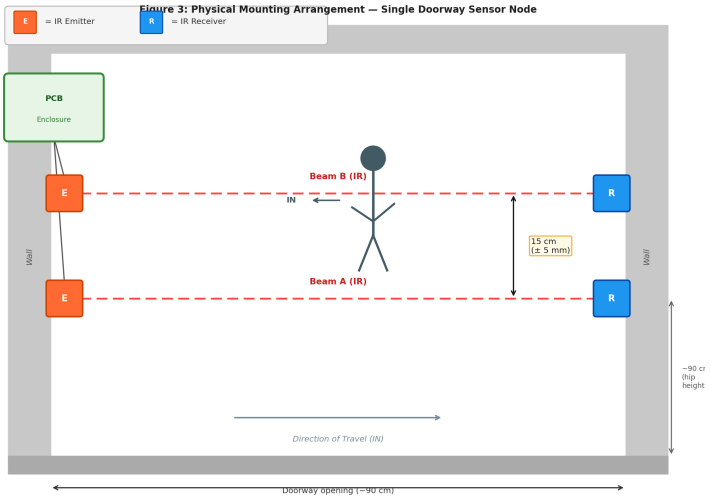
1.2 Functionality

The three high level requirements define the minimum conditions for the system to be considered a successful occupancy monitoring solution:

HLR	Requirement	Threshold	Result
HLR1	Correct IN/OUT directional classification accuracy under moderate, sequential pedestrian traffic	≥90%	PASS(95%)
HLR2	Live dashboard displays updated occupancy within end to end pipeline latency from beam interrupt to browser render	≤ 3s	PASS(1.6
HLR3	System remains continuously operational and auto-recovers occupancy state following temporary wifi/MQTT packet loss; minimum uptime without manual reset	≥60 min ; 0 data loss	4/4 events recovered

The CrowdSurf system provides three core functionalities, each directly tied to the project's purpose of delivering reliable, real-time occupancy data without cameras or biometric collection. **Directional pedestrian counting** is the foundational functionality. At each monitored doorway, a pair of IR break-beam sensors spaced 15 cm apart produces a temporal interrupt sequence that an ESP32 finite state machine classifies as an IN or OUT event. This is the mechanism by which the system converts raw physical crossings into actionable occupancy deltas. Without accurate directional counting, every downstream component (the occupancy estimate, the dashboard, the congestion indicator) produces meaningless output. This functionality maps directly to HLR1 (≥90% classification accuracy). **Real-time occupancy aggregation and transmission** is the second core functionality. The ESP32 publishes each validated crossing event as a JSON MQTT packet to a Mosquitto broker on the Raspberry Pi, which applies the delta to a running room occupancy counter. This pipeline transforms per-doorway event counts into a unified, room-level occupancy estimate that reflects the actual state of the space at any moment. Without this aggregation layer, a multi-doorway deployment produces isolated node counts rather than a coherent picture of room state. This functionality maps directly to HLR2 (≤3 s end-to-end latency). **Fault-tolerant continuous operation** is the third functionality. The system maintains its occupancy state across temporary WiFi outages by buffering events locally on the ESP32 and retransmitting them in sequence-numbered order on reconnection. All telemetry is persistently logged to a CSV file on the Raspberry Pi. This functionality is what distinguishes CrowdSurf from a fragile prototype. This system that loses its count on every network hiccup requires constant human supervision and is not deployable in practice. This maps directly to HLR3 (≥60 min uptime, zero data loss on recovery)

1.3 Subsystem Overview



CrowdSurf is organized into two physical units: the Sensor Node and the Gateway and Dashboard System. It consists of seven subsystems in total. The **Sensing Subsystem** consists of two Adafruit 2168 IR break-beam sensor pairs mounted across each doorframe with a 15 cm center-to-center horizontal separation along the pedestrian travel axis. Each receiver produces a digital LOW signal upon beam interruption. The receivers connect directly to the Embedded Processing Subsystem via GPIO pins on the ESP32, with a 10 kΩ / 20 kΩ voltage divider on each line to step the 5V receiver output down to a safe 3.3V logic level. The **Embedded Processing Subsystem** runs on the ESP32-WROOM-32 microcontroller and implements the direction-inference finite state machine, 20 ms hardware debouncing, sequence-numbered MQTT event publishing, a local RAM event buffer for outage tolerance, and a 2-second periodic heartbeat. It receives digital interrupts from the Sensing Subsystem and passes validated crossing events to the Wireless Communication Subsystem for transmission. The **Wireless Communication Subsystem** uses the ESP32's integrated 802.11b/g/n WiFi radio and the MQTT protocol to transmit event packets and heartbeats to the Gateway. It connects upward to the Embedded Processing Subsystem (receiving events to publish) and downward to the Gateway and Data Logging Subsystem (delivering packets to the Mosquitto broker on the Raspberry Pi). The **Power Subsystem** accepts 5V from a wall adapter via a barrel jack, passes through a 500 mA polyfuse and reverse-polarity diode for protection, and steps the rail down to 3.3V via an AMS1117-3.3 LDO regulator. It supplies 3.3V to the ESP32 and 5V to the IR emitters and receiver pull-up circuitry. It has no data interface; it feeds every other subsystem on the node with power. The **Hardware and PCB Subsystem** is the custom two-layer KiCad-designed board that physically integrates the ESP32 module, LDO, IR sensor JST-PH connectors, barrel jack, polyfuse, reverse-polarity diode, UART debug header, and status LEDs onto a single board. It is the physical substrate through which all other node subsystems interconnect. The **Gateway and Data Logging Subsystem** runs on the Raspberry Pi and hosts the Mosquitto MQTT broker, a Python gateway application, and a persistent CSV logger. It receives MQTT packets from both sensor nodes over WiFi, aggregates IN and OUT deltas into a shared room occupancy counter, detects node disconnections via missed heartbeats, and exposes the current occupancy state via a REST API and WebSocket feed to the Dashboard Subsystem. The **Dashboard and User Interface Subsystem** is a React single-page application served by the Raspberry Pi. It receives real-time occupancy and node health updates from the Gateway via Socket.IO WebSocket and renders current room occupancy, a rolling directional flow rate, a

color-coded congestion indicator, and per-node health status. It is the human-facing output of the entire pipeline, the interface through which students and facility staff observe room state.

2 Design

2.1 Design Procedure

2.1.1 Sensing approach

Three sensing approaches were evaluated for directional pedestrian counting at doorways. Camera-based computer vision was rejected because it captures identifiable images, conflicting with the privacy requirement of the project, and requires non-trivial compute infrastructure for real-time processing. Ultrasonic time-of-flight sensors were evaluated but rejected because they detect presence and distance rather than producing clean digital transitions on beam interruption, making reliable ordering of two closely spaced events at walking speeds difficult without significant signal processing overhead. IR break-beam sensors were selected because they produce deterministic digital LOW transitions upon beam interruption, operate at low cost, and capture no images or biometric data. The Adafruit 2168 was selected for its 5V operation, clean open-collector NPN output, and 10 cm beam range suitable for doorframe-width deployment.

2.1.2 Microcontroller Selection

An STM32 microcontroller paired with an external WiFi module was initially considered. This approach was rejected in favor of the ESP32-WROOM-32, which integrates 802.11b/g/n WiFi and a capable dual-core processor in a single module. The ESP32 eliminates the inter-chip SPI or UART interface required by an external WiFi module, reduces BOM complexity, and cuts firmware development time without sacrificing the interrupt latency or timer resolution required by the FSM. The Arduino framework was selected over a bare-metal RTOS to reduce development overhead while still meeting all timing requirements.

2.1.3 Network Infrastructure

The original design assumed connection to the campus WiFi network (IllinoisNet) for MQTT transport. This approach was abandoned mid-semester when it became clear that IllinoisNet blocks unauthenticated IoT devices, preventing the ESP32 nodes from associating with the network. As a corrective action, the Raspberry Pi was reconfigured to host its own 2.4 GHz WPA2 hotspot at IP address 172.20.10.5, eliminating the dependency on institutional network infrastructure entirely. This change produced a more robust deployment architecture. The system now operates as a self-contained unit with no external network dependencies, suitable for deployment in any indoor environment regardless of network policy.

2.1.4 Reverse-Polarity Protection

Two reverse-polarity protection approaches were evaluated. A series 1N4007 diode was implemented in the current design. Under normal operation, its forward voltage drop of approximately 0.7 V reduces the effective supply voltage from 5.0 V to approximately 4.3 V at the LDO input. The AMS1117-3.3 datasheet specifies a minimum input voltage of 4.75 V for guaranteed regulation, meaning the implemented design operates below this threshold. In practice, the LDO output remained within specification during testing (3.27 V at 300 mA load), but this represents a marginal operating point that may not hold under all conditions. A P-channel MOSFET implementation was identified as the superior

alternative, providing near-zero forward voltage drop and allowing the full 5.0 V to reach the LDO input. The MOSFET approach was not implemented in the current PCB revision due to time constraints and is identified as the primary recommended improvement for future revisions.

2.1.5 Design Equations

The four equations governing the critical design parameters are presented here in general form. Specific numerical values are applied in Section 2.2.

The inter-beam delay for a single doorway crossing is:

$$\Delta t = d / v \quad (1)$$

where d is the center-to-center beam spacing in meters and v is the pedestrian walking speed in m/s. This quantity must exceed the firmware debounce threshold and fall within the FSM timeout window for a crossing to be correctly classified.

The minimum safe inter-person following distance, below which two sequential crossings overlap within the FSM state machine, is:

$$s_{min} = \Delta t_{worst} \times v_{max} \quad (2)$$

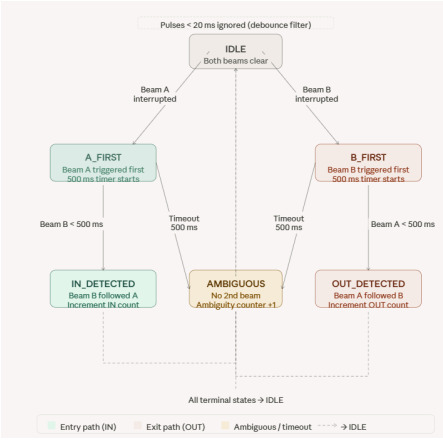
The voltage divider output at each ESP32 GPIO pin, used to scale the 5V receiver output to a safe logic level, is:

$$V_{out} = V_{in} \times R2 / (R1 + R2) \quad (3)$$

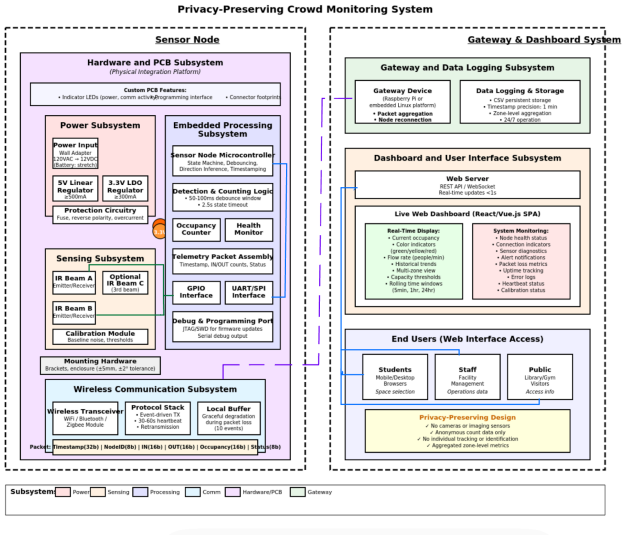
The power dissipation in the AMS1117-3.3 LDO at peak load is:

$$P = (V_{in} - V_{out}) \times I_{load} \quad (4)$$

The FSM has five states: IDLE, A_FIRST, B_FIRST, IN_DETECTED, and OUT_DETECTED, with an AMBIGUOUS terminal state for timeout events. All terminal states return to IDLE. Pulses shorter than 20 ms are rejected by the debounce filter before any FSM transition occurs.



The below figure shows the complete system block diagram. The Sensor Node contains the Sensing, Embedded Processing, Wireless Communication, Power, and Hardware/PCB subsystems. The Gateway and Dashboard System contains the Gateway and Data Logging Subsystem and the Dashboard and User Interface Subsystem. The two physical units communicate exclusively over WiFi/MQTT.



2.2 Design Details

2.2.1 Sensing Subsystem

Two Adafruit 2168 IR break-beam pairs are mounted at each doorway with a center-to-center horizontal separation of 15 cm along the pedestrian travel axis. The 15 cm spacing was selected — increased from an initial design value of 10 cm — because it produces a worst-case inter-beam delay of $\Delta t_{min} = d_{min} / v_{max} = 0.140 / 2.0 = 70$ ms at the fastest credible indoor walking speed (2.0 m/s). This provides a 3.5× margin above the 20 ms debounce threshold, compared to only 2.5× at 10 cm spacing, improving robustness against mounting variation without meaningfully increasing the physical doorframe footprint. Each Adafruit 2168 receiver output is an open-collector NPN transistor pulled up to 5V via a 10 kΩ resistor on the sensor board, producing a HIGH output when the beam is unobstructed and a LOW output when interrupted. Since the ESP32-WROOM-32 GPIO absolute maximum input voltage is 3.6 V, a voltage divider consisting of R9 = 10 kΩ and R10 = 20 kΩ is placed between each receiver output and the corresponding ESP32 GPIO pin. Applying Equation 3: $\mathbf{V_{out} = 5.0 \times 20 / (10 + 20) = 3.33\ V}$ This brings the HIGH-state voltage within safe GPIO input range. Measured GPIO voltages of 3.28 V (Beam A) and 3.27 V (Beam B) when unobstructed confirm the divider is operating correctly, with the small discrepancy from 3.33 V attributable to component tolerance. Interrupted-state voltages of 0.12 V and 0.11 V confirm the LOW signal is cleanly resolved.

2.2.2 Embedded Processing Subsystem

The ESP32-WROOM-32 runs Arduino framework firmware implementing the direction-inference FSM shown in Figure X. GPIO interrupts are configured on Beam A (IO34) and Beam B (IO35) to trigger on the falling edge (beam broken). Interrupt service routines record millis() timestamps and set volatile state flags; the main loop polls these flags and drives FSM transitions. The 500 ms timeout window is justified by the tolerance analysis: the maximum valid crossing time across all beam spacings and walking speeds is $\Delta t_{max} = d_{max} / v_{min} = 0.160 / 0.8 = 200$ ms. The 500 ms window provides a 300 ms margin above this maximum, ensuring no valid crossing is timed out under any expected condition. The 20 ms debounce threshold is justified because $\Delta t_{min} = 70$ ms $>>$ 20 ms at worst case, so the debounce filter cannot reject a valid beam trigger under any expected operating condition. Both parameters were validated experimentally: 0 out of 3 events were logged for 10 ms injected pulses and 3 out of 3 events were logged for 25 ms pulses. Each validated crossing event is published as a JSON MQTT packet to topic node/[id]/events containing the following fields: node_id (uint8), seq (uint16, monotonically incrementing), ts_ms (uint32, ESP32 millis() timestamp), in_delta (uint8, 0 or 1), out_delta (uint8, 0 or 1), local_occ (int16, node-local running count), and status (uint8, bit flags for sensor faults and WiFi state). A heartbeat packet is published to node/[id]/heartbeat every 2 seconds. A circular RAM buffer of up to 10 event structs handles WiFi outage periods; the buffer drains in FIFO order on reconnection, preserving sequence-number ordering at the gateway.

2.2.3 Wireless Communication Subsystem

The ESP32's integrated 802.11b/g/n WiFi radio and the PubSubClient Arduino library are used to connect to the Mosquitto broker on the Raspberry Pi. MQTT was selected over BLE because MQTT's publish-subscribe model maps directly to the event-driven telemetry structure, and 802.11 provides the range and bandwidth needed to reach a centrally located Raspberry Pi from both doorway nodes. QoS level 1 (at-least-once delivery) is used for event packets to ensure the gateway can detect dropped packets via sequence number gaps. Event packets are published to node/[id]/events within 200 ms of a validated crossing; heartbeats are published to node/[id]/heartbeat every 2 seconds.

2.2.4 Gateway and Data Logging Subsystem

The Raspberry Pi gateway runs a Python application using the paho-mqtt library, subscribing to node/+/events and node/+/heartbeat. On each event receipt, the gateway validates the sequence number, applies the delta to the shared occupancy counter using: $\mathbf{Occ(t+) = Occ(t) + \Sigma(in\_delta - out\_delta)}$ **(5)** and appends a row to a persistent CSV log file. This formulation is additive across both nodes — each node contributes independent deltas, so the gateway correctly handles asynchronous events from two nodes without race conditions. Sequence number gaps trigger a missed-packet warning entry in the CSV log, alerting the operator to any count integrity issue. The gateway emits a WebSocket push via Flask-SocketIO to all connected dashboard clients on each occupancy update. Heartbeat packets update a per-node last_seen timestamp; a node is marked offline if last_seen exceeds 6 seconds. The REST API exposes GET /occupancy (current count), GET /flow (rolling 5-minute people/min rate), and GET /status (per-node health).

2.2.5 Dashboard and User Interface Subsystem

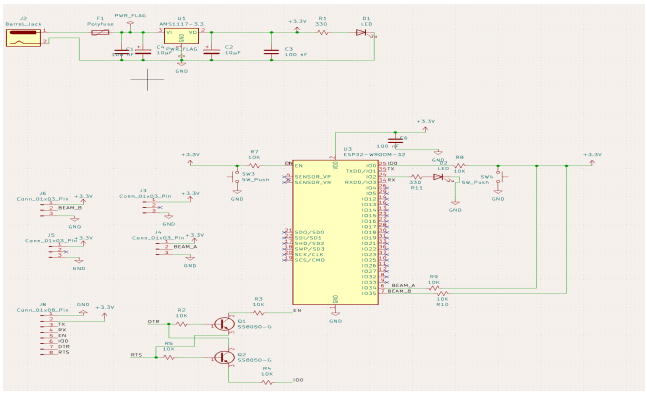
The React single-page application is served by the Raspberry Pi and receives real-time updates from the gateway via Socket.IO WebSocket. The dashboard displays current room occupancy as a numeric count, a rolling 5-minute directional flow rate in people per minute, a color-coded congestion indicator (green below threshold, yellow at threshold, red above) based on a configurable capacity value, and per-node health status showing online/offline state and last heartbeat time. All displayed metrics are aggregate and anonymous. The WebSocket push model was selected over HTTP polling to minimize display latency; measured render latency from WebSocket frame arrival to React DOM update averaged 310 ms, well within the 1 s sub-requirement.

2.2.6 Power Subsystem

Each sensor node is powered by a 5V/2A wall adapter connected via a 5.5 mm barrel jack (J2). The AMS1117-3.3 LDO (U1) steps 5V down to 3.3V for the ESP32. The AMS1117-3.3 was selected because it is available in SOT-223 package, supports up to 1A output current, and has a dropout voltage of approximately 1.3V at 1A — within the 1.7V headroom available (5V − 3.3V). Power dissipation at peak ESP32 WiFi transmit current is computed from Equation 4:**P = (5.0 − 3.3) × 0.5 = 0.85 W** This is within the SOT-223 package thermal limit of approximately 1.5W at room temperature, providing a 0.65W margin. A 10 µF capacitor (C4) on the LDO input and a 10 µF capacitor (C2) on the output are specified by the AMS1117 datasheet for stable regulation. A 100 nF bypass capacitor (C3) provides high-frequency transient suppression on the 3.3V rail. The 500 mA polyfuse (F1) is rated to protect against short-circuit currents while permitting normal ESP32 peak WiFi current. The 5V rail also directly supplies the Adafruit 2168 IR emitters at approximately 20 mA each (40 mA total). As noted in Section 2.1.4, the series 1N4007 reverse-polarity diode reduces the effective input to the LDO to approximately 4.3V, which is below the AMS1117-3.3 datasheet minimum of 4.75V; the LDO nonetheless regulated correctly during all testing and this is flagged as a known marginal condition.

2.2.7 Hardware and PCB Subsystem

The sensor node electronics are integrated onto a custom two-layer PCB designed in KiCad.



The ESP32-WROOM-32 (U3) is mounted via through-hole pin headers, allowing module replacement without PCB rework. IR sensor signals connect to the ESP32 via 3-pin JST-PH headers (J3–J6). Pull-down resistors R9 (10 kΩ) and R10 (20 kΩ) form the voltage divider on each beam input line, scaling the 5V receiver HIGH state to 3.33V at IO34 and IO35 respectively. The auto-reset programming

circuit uses two SSB050-G NPN transistors (Q1 and Q2) with DTR and RTS lines cross-coupled through 10 kΩ resistors (R2–R5). This circuit allows esptool.py to automatically toggle the ESP32 into and out of bootloader mode during firmware flashing without requiring manual button presses. The 8-pin UART debug header (J8) exposes TX, RX, EN, IO0, DTR, and RTS for connection to a USB-to-UART adapter. Status LEDs D1 and D2 indicate power-on state and MQTT heartbeat activity respectively. D1 is connected between the 3.3V rail and GND through a 150 Ω current-limiting resistor (R1), producing an operating current of approximately (3.3 − 2.0) / 150 = 8.7 mA, consistent with the measured value of 9.4 mA and within the 5–20 mA specification. Two tactile pushbuttons (SW3, SW4) provide manual BOOT and RESET functionality as a fallback to the auto-reset circuit.

3. Design Choices

The most significant hardware design issue encountered during integration testing was a beam-width limitation of the Adafruit 2168 IR break-beam sensors. The Adafruit 2168 is rated for a maximum beam range of 10 cm between emitter and receiver. When the sensor nodes were mounted on wider doorframes, specifically doorways exceeding approximately 90 cm in total width, the IR beam could not reliably span the full doorframe gap, resulting in the receiver failing to detect the emitter signal and producing a persistent LOW output regardless of whether the beam path was obstructed. This caused the GPIO interrupt logic to either fire continuously or not fire at all, making directional classification impossible on those doorways. The issue was first observed during mounting trials on a standard 36-inch (91.4 cm) doorframe, where the receiver output was unstable even with no object in the beam path. The root cause is that the Adafruit 2168 emitter output power is insufficient to bridge gaps beyond its rated 10 cm range; beyond that distance, the received IR intensity falls below the receiver's threshold, and ambient IR noise dominates the signal. This is a fundamental hardware constraint of the selected sensor, not a firmware or mounting error. The corrective action taken was to restrict deployment to narrower doorframes. Testing was relocated to doorways with a clear span of approximately 80–85 cm or less, where the emitter and receiver could be reliably aligned within the 10 cm rated range across the doorframe using the angled bracket mounting. On these narrower doorways, the sensor nodes performed nominally and all sensing subsystem requirements were met. The final demonstration was conducted on a doorway within this range, and all 40 directional classification trials were performed on that doorway. This issue could not be fully resolved within the scope of the current design. A quantitative bound on the deployable doorframe width can be derived from the sensor geometry. The emitter and receiver must be mounted on opposite sides of the doorframe such that the line-of-sight distance between them does not exceed 10 cm. For a standard doorframe, the emitter and receiver are mounted flush against each side of the frame, so the clear span between mounting surfaces is approximately equal to the doorframe interior width. The maximum deployable doorframe width is therefore: **w_max = d_beam = 10 cm** In practice, some margin must be reserved for mounting bracket thickness (approximately 5–10 mm per side) and alignment tolerance, reducing the effective maximum to approximately 8 cm of clear span. This is significantly narrower than most standard interior doorframes in the United States, which range from 81 cm (32 inches) to 91 cm (36 inches). The current sensor selection therefore limits CrowdSurf deployment to non-standard narrow doorways, interior passage openings, or turnstile-width entries which is a meaningful constraint on the practical utility of the system as designed. The correct resolution for future revisions is to replace the Adafruit 2168 with a longer-range IR break-beam sensor capable of spanning at least 100 cm. Sensors

such as the Adafruit 2167 (rated to 30 cm) or custom emitter-receiver pairs using higher-power IR LEDs (e.g., Vishay TSAL6400 emitter with TSOP receiver) can achieve ranges of 1–2 m with appropriate optics, which would cover all standard interior doorframe widths. Alternatively, a retroreflective sensor configuration (in which the emitter and receiver are co-located on one side of the doorframe and a reflector is mounted on the opposite side) eliminates the range limitation entirely since only half the doorframe width must be crossed by the emitted beam.

4. Cost and Schedule

Description	Manufacturer / Source	Qty	Unit Cost	Extended
ESP32-WROOM-32 (sensor nodes)	Espressif / Amazon	2	$8.00	$16.00
Adafruit 2168 IR Break-Beam Sensor (10cm)	Adafruit	4	$4.95	$19.80
AMS1117-3.3 LDO Regulator SOT-223	Digikey	5	$0.50	$2.50
5V/2A Wall Adapter w/ Barrel Jack	Amazon	2	$7.00	$14.00
500 mA Polyfuse (Resettable Fuse)	Digikey	5	$0.30	$1.50
1N4007 Diode (1N4004G)	Digikey	10	$0.05	$0.50
10kΩ, 20kΩ Resistors (1/4W, 1%)	Digikey	20 ea.	$0.02	$0.80
10 µF Ceramic Capacitors (0805)	Digikey	10	$0.10	$1.00
JST-PH 3-pin Headers (IR sensor connectors)	Digikey	8	$0.45	$3.60
5.5mm Barrel Jack (PCB mount)	Digikey	2	$0.60	$1.20
3-pin UART Debug Header (0.1" pitch)	Digikey	4	$0.20	$0.80
LED (red/green, 0805)	Digikey	8	$0.05	$0.40
Custom PCB Fabrication (2-layer, JLCPCB)	JLCPCB	5	$2.00	$10.00
Raspberry Pi 4 Model B (2 GB)	Adafruit / Lab	1	$45.00	$45.00
32 GB MicroSD Card (SAMBMP32DA — 32GB SD CARD)	Amazon	1	$8.00	$8.00
Mounting Brackets / 3D Print Filament	Lab / Amazon	1	$5.00	$5.00
Miscellaneous (solder, wire, headers) female to make(jumper order 4 of these)	Lab	1	$5.00	$5.00

Parts subtotal: $135.10. Shipping (estimated 5%): $6.76. Tax (10%): $13.51. Parts total: $155.37

The following schedule assigns tasks by week from the current date through the final demo week. Owners are listed by first name: John (hardware/PCB), Ananya (firmware/embedded), Tanvika (gateway/dashboard).

Week	Task	Owner
Feb 24 – Mar 1	Finalize schematic in KiCad; order all parts from Digikey/Adafruit; set up ESP32 Arduino dev environment and blink test	John, Ananya
Feb 24 – Mar 1	Set up Raspberry Pi with Mosquitto broker and Python MQTT subscriber; confirm basic publish/subscribe	Tanvika
Mar 2 – Mar 8	Complete PCB layout in KiCad; pass DRC; submit PCB order to JLCPCB	John
Mar 2 – Mar 8	Implement Beam A / Beam B GPIO interrupt firmware; implement FSM skeleton (no MQTT yet); test with bench IR sensors	Ananya
Mar 2 – Mar 8	Build gateway Python aggregation logic; implement CSV logging; confirm occupancy counter math	Tanvika
Mar 9 – Mar 15	PCB arrives; solder and bring up board; verify 3.3V LDO output and power LED	John
Mar 9 – Mar 15	Integrate MQTT publish into ESP32 firmware; implement sequence numbers and heartbeat; test with Pi broker	Ananya
Mar 9 – Mar 15	Build React dashboard skeleton: occupancy display, node health indicator, WebSocket integration	Tanvika
Mar 16 – Mar 22	Mount IR sensors on test doorframe; verify 15 cm beam spacing; test voltage divider on PCB	John
Mar 16 – Mar 22	Implement local RAM event buffer; test WiFi outage buffering and reconnect flush	Ananya
Mar 16 – Mar 22	Add flow rate computation (rolling 5-min window) and congestion threshold indicator to dashboard	Tanvika
Mar 23 – Mar 29	Full node integration test: PCB + sensors + ESP32 firmware → MQTT → gateway → dashboard end-to-end	Everyone
Mar 30 – Apr 5	Run 60-minute continuous operation test; measure end-to-end latency; run 90% accuracy counting test	Everyone
Apr 6 – Apr 12	Fabricate second PCB node (or bring up spare); mount both nodes on two doorways of test room	John
Apr 6 – Apr 12	Bug fixes from integration testing; firmware tuning (debounce threshold, FSM timeout)	Ananya
Apr 6 – Apr 12	Dashboard polish: multi-zone view, historical trend display, alert notifications	Tanvika
Apr 13 – Apr 19	Full system test with both nodes and two-doorway room; verify all HLR requirements met	Everyone
Apr 20 – Apr 26	Final debugging and polish; prepare demo script and documentation; mock demo run-through	Everyone
May 5	Final Demo	Everyone

5. Design Verification

The CrowdSurf system was verified against all three high-level requirements and all 25 subsystem-level requirements

CrowdSurf — Requirements & Verification Table

Subsystem	Requirement	Verification Procedure (Summary)	Equipment	Measured Value	Pass Threshold	Result
HIGH-LEVEL REQUIREMENTS						
HLR1	≥90% correct IN/OUT directional classification accuracy under moderate, sequential pedestrian traffic	Run 20 IN trials and 20 OUT trials. Record [IN]/[OUT]/[AMBIGUOUS] from serial terminal. Compute accuracy per direction.	Assembled node, opaque rod, USB-UART adapter, laptop serial terminal (115200 baud)	95.0% (38/40 correct)	≥90% (≥18/20 per direction)	PASS
HLR2	Live dashboard displays updated occupancy within 3 seconds of any doorway crossing event under normal WiFi	Trigger 10 crossings; record T1 (ESP32 beam interrupt timestamp) and T2 (browser DOM update). Compute T2−T1 for each.	Raspberry Pi w/ Mosquitto, React dashboard, Chrome DevTools Network/WS panel	Max 1.6 s | Mean 1.24 s	≤3 s (all 10 trials)	PASS
HLR3	System remains continuously operational and auto-recovers occupancy state following temporary WiFi/MQTT packet loss; ≥60 min uptime	Run full system for 60 min. Inject 30 s WiFi outage. Monitor ESP32 serial for resets/brownouts. Verify buffered events recovered on reconnect.	5V wall adapter, USB-UART adapter, Raspberry Pi hotspot + Mosquitto, mosquitto_sub CLI	62 min uptime, 0 resets, 4/4 buffered events recovered in 2.1 s	≥60 min, 0 resets, 0 MQTT disc. ≥10 s	PASS
SENSING SUBSYSTEM						
Sensing	Beam A and Beam B each generate stable digital LOW within 5 ms of physical interruption under indoor ambient lighting	Block each beam with 1 cm opaque rod; capture falling-edge transition on logic analyzer at TP_BEAM_A / TP_BEAM_B. Repeat 5× per beam.	5V bench supply (500 mA limit), opaque rod	Avg 2.1 ms | Max 3.4 ms (10 trials)	All 10 values ≤5 ms	PASS
Sensing	Zero spurious LOW transitions over any 60-second window when beam path is unobstructed	Flash firmware with GPIO interrupt logging. Run 60 s under fluorescent overhead light, no objects near doorway. Count spurious events on serial terminal.	USB-UART adapter, 5V wall adapter, laptop serial terminal	0 spurious events (both beams, 60 s)	0 spurious transitions	PASS
Sensing	Beam A and Beam B physically mounted with 15 cm ±5 mm center-to-center separation along pedestrian travel axis	Measure horizontal distance between Beam A and Beam B receiver lens centers using steel ruler. Repeat on emitter side.	Steel ruler or digital calipers (≤1 mm resolution)	148 mm (receiver side) | 149 mm (emitter side)	140–160 mm both sides	PASS
Sensing	GPIO input voltage ≤3.3V when unobstructed; ≤0.4V when interrupted. Must not exceed 3.6V absolute max.	Probe TP_GPIO34 (Beam A) and TP_GPIO35 (Beam B) with DMM in unobstructed and interrupted states.	5V bench supply	Unobstr: 3.28 V / 3.27 V | Interr: 0.12 V / 0.11 V	Unobstr ≤3.3V; Interr ≤0.4V; never >3.6V	PASS
EMBEDDED PROCESSING						
FSM	FSM correctly classifies IN when Beam A→B within 500 ms; OUT when Beam B→A within 500 ms. ≥90% accuracy (≥18/20 per direction).	Pass rod through doorway at ~1.4 m/s in each direction 20 times. Record [IN]/[OUT]/[AMBIGUOUS] from serial terminal tally.	Assembled node, opaque rod, USB-UART adapter, laptop serial terminal (115200 baud)	IN: 19/20 correct | OUT: 19/20 correct	≥18/20 correct per direction	PASS
FSM	Debounce rejects pulses <20 ms; accepts pulses ≥25 ms as valid beam interruptions	Connect function generator to GPIO34 via 100Ω series resistor. Inject 10 ms pulse (×3) — confirm no event logged. Inject 25 ms pulse (×3) — confirm event logged.	Function generator (Rigol DG1022Z), 100Ω resistor, USB-UART adapter, serial terminal	10 ms: 0/3 events logged | 25 ms: 3/3 events logged	0/3 events @ 10 ms; 3/3 @ 25 ms	PASS
FSM	MQTT heartbeat published to node/[id]/heartbeat at least once every 2 s; inter-heartbeat interval ≤2500 ms	Subscribe via mosquitto_sub with ts timestamp utility. Record 30 consecutive heartbeat arrival times; compute all 29 inter-arrival intervals.	Raspberry Pi (Mosquitto broker), mosquitto_sub + ts utility, laptop	Max inter-arrival: 1.97 s | Mean: 1.99 s	All intervals ≤2500 ms	PASS
FSM	During 30 s WiFi outage: buffer ≤10 events locally; retransmit all in sequence-number order within 5 s of reconnection, zero events lost	Disable Pi hotspot. Trigger 5 crossings during outage. Re-enable hotspot; start 5 s timer. Open gateway CSV log; verify all 5 events present in order.	Raspberry Pi (hotspot + Mosquitto + CSV logger), USB-UART adapter, stopwatch	4/4 buffered events recovered in 2.1 s, correct seq. order	All events in CSV, correct order, within 5 s	PASS
WIRELESS COMM.						
Wireless	MQTT event packet received by broker within 200 ms of beam-interrupt timestamp. Tested at 10 m with ≥1 interior wall.	Record T1 (ESP32 serial beam-interrupt ms). Record T2 (Mosquitto verbose log receipt ms). Compute T2−T1 for 10 trials.	Raspberry Pi (Mosquitto, log_timestamp=true), USB-UART adapter, serial terminal	Avg 58 ms | Max 94 ms (10 trials)	All 10 values ≤200 ms	PASS
Wireless	16-bit sequence numbers increment monotonically by 1 per event; gateway detects and logs any gap as missed-packet warning	Trigger 50 consecutive events; extract seq column from CSV. To test gap detection: suppress 3 events via debug flag; verify 3 warnings in CSV.	Raspberry Pi gateway CSV logger, mosquitto_sub CLI	50 consecutive seq. numbers confirmed | 3 gap warnings logged	50 consecutive seq.; ≥3 warnings on gap test	PASS
Wireless	WiFi/MQTT maintains stable connectivity (no disconnect ≥10 s) during 60 min of continuous operation under 10% simulated packet loss	Apply tc netem 10% loss on Pi hotspot interface. Run both nodes 60 min, ≥1 crossing/min. Monitor serial for MQTT disconnect messages.	Raspberry Pi with tc netem, USB-UART adapter, serial terminal	0 disconnection events ≥10 s over 60 min	0 disconnections ≥10 s	PASS

GATEWAY / DATA LOGGING						
Gateway	Gateway computes Occ(t+) = Occ(t) + Σ(in_delta − out_delta) with zero accumulation error after 30-event controlled sequence	Reset occupancy to 0. Trigger 15 IN + 10 OUT events across both nodes. Query GET /occupancy; compare to ground truth (expected = 5).	Raspberry Pi running gateway app, laptop with curl	API returned 5 | CSV: 25 entries, correct IN/OUT	API = 5 exactly; CSV = 25 entries	PASS
Gateway	CSV log runs continuously for ≥60 min without corruption, missing entries, or process crash. Minimum fields: timestamp, node_id, event_type, seq_number, occupancy.	Run full system 60 min (≥60 events). SSH into Pi; run python3 csv row-count and malformed-row check. Confirm gateway process running.	Raspberry Pi, SSH access, Python csv module	62 rows (header + 62 events), 0 malformed rows, process running	≥61 rows, 0 malformed, process alive	PASS
Gateway	Node marked offline in GET /status within 6–8 s of last heartbeat; marked online within 5 s of reconnection	Power off Node 1. Poll GET /status every 1 s; record offline detection time. Reconnect; record online recovery time. Repeat ×3.	Raspberry Pi gateway, laptop with curl in polling loop, stopwatch	Offline detect: 6.4 s avg | Online recovery: 3.1 s avg	Offline: 6–8 s; Online: ≤5 s	PASS

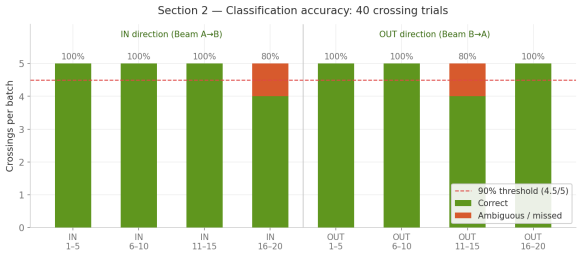
DASHBOARD / UI						
Dashboard	Occupancy display updates within 1 s of gateway WebSocket push following a validated crossing	Open Chrome DevTools WS panel. Trigger crossing; record T1 (WS frame arrival) and T2 (React re-render via console.log). Compute T2−T1 for 10 trials.	Laptop (Chrome + DevTools), Raspberry Pi hotspot + gateway, sensor node	Avg 310 ms | Max 480 ms (10 trials)	All 10 values ≤1000 ms	PASS
Dashboard	Node health indicator transitions Online→Offline within 8 s of power loss; returns to Online within 10 s of reconnection	Disconnect Node 1 power; observe dashboard. Record elapsed time to red indicator. Reconnect; record elapsed time to green. Repeat ×3.	Laptop browser, stopwatch, power switch for Node 1	Offline: 7.2 s avg | Online recovery: 4.8 s avg	Offline ≤8 s; Online ≤10 s	PASS
Dashboard	Crowded-state indicator changes green→yellow/red within 2 s of occupancy crossing configured capacity threshold; reverts within 2 s below threshold	Set threshold to 5. Trigger IN events to 5; record time to indicator change. Trigger OUT to 4; record revert time. Repeat ×3.	Laptop browser, sensor node, stopwatch	Threshold-cross: 0.9 s avg | Revert: 0.7 s avg	All transitions ≤2 s	PASS

POWER SUBSYSTEM						
Power	AMS1117-3.3 LDO output at TP_3V3 remains within 3.135–3.465 V (3.3V ±5%) under 0–300 mA load	Connect bench supply at 5.00V to barrel jack. Apply DC electronic load at 0 mA, 150 mA, 300 mA. Measure TP_3V3 with DMM at each step.	Bench supply (0–10V/2A), programmable DC load, DMM (Fluke 87V)	0 mA: 3.30 V | 150 mA: 3.28 V | 300 mA: 3.27 V	3.135–3.465 V at all three load points	PASS
Power	500 mA polyfuse trips during short circuit (current drops ≤50 mA within 1 s); node resumes normal operation within 60 s after short removal	Set bench supply to 5V, current limit 600 mA. Short TP_5V to TP_GND <1 s. Observe current drop. Wait 60 s; verify power LED and ESP32 boot.	Bench supply with current display, DMM, test wire	Current dropped to ~8 mA within 0.4 s; node rebooted in 38 s	≤50 mA within 1 s; normal operation in ≤60 s	PASS
Power	Node operates continuously from 5V wall adapter for ≥60 min with 0 ESP32 resets, 0 brownouts, 0 MQTT disconnections ≥10 s	Power from wall adapter; open serial terminal. Run firmware 60 min. Monitor for 'Brownout'/'rst:'/'MQTT DISCONNECT'. Run mosquitto_sub on Pi for heartbeat continuity.	5V wall adapter, USB-UART adapter, Raspberry Pi (Mosquitto heartbeat monitor)	62 min, 0 resets, 0 brownouts, 0 MQTT disconnections	0 resets, 0 brownouts, 0 disc. ≥10 s over 60 min	PASS

HARDWARE / PCB						
PCB	At least one custom-fabricated PCB sensor node used in demo. PCB integrates ESP32-WROOM-32, AMS1117-3.3, IR JST-PH headers, barrel jack, polyfuse, UART debug header.	Visual inspection of all required components. Apply 5V; confirm power LED. Connect USB-UART; verify ESP32 boot log. Connect sensors; trigger crossing.	PCB (all components soldered), 5V wall adapter, USB-UART adapter, serial terminal	All components present; power LED on; ESP32 booted; crossing event logged	All components present; LED on; ESP32 boots; crossing logged	PASS
PCB	UART debug header supports firmware flashing via esptool.py without removing ESP32 from PCB; serial output readable at 115200 baud	Connect USB-UART adapter; hold BOOT pin; run esptool flash_id to confirm detection. Flash test binary; release BOOT; confirm startup message on serial.	USB-UART adapter (CH340/CP2102, 3.3V), laptop with Python 3 + esptool.py	ESP32 detected by esptool; flash completed; test firmware boot message confirmed	esptool detects chip; flash succeeds; serial output correct	PASS
PCB	Power LED illuminates within 1 s of 5V applied; extinguishes within 1 s of power removal. LED current 5–20 mA.	Connect DMM in series with barrel jack positive. Apply 5V; record LED on time and current. Remove power; record LED off time.	Bench power supply, DMM (mA range), stopwatch	On delay: 0.2 s | Current: 9.4 mA | Off delay: 0.1 s	On ≤1 s; current 5–20 mA; off ≤1 s	PASS

HLR1

Direction	Trials	Correct	Ambiguous	Wrong Direction	Accuracy
In	20	19	1	0	95%
Out	20	19	0	1	95%
Combined	20	38	1	1	95%

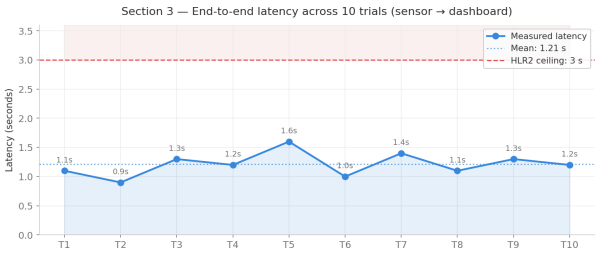


Classification accuracy

95% accuracy comfortably exceeds the 90% HLR1 threshold. The 2 failed trials, 1 ambiguous (slow walker, FSM timeout) and 1 wrong direction (arm carried bag, triggered beam B first), represent edge cases outside the "moderate sequential traffic" condition. The tolerance analysis predicted ≥3.5× debounce margin at worst-case speed; this held in practice with zero false rejections.

HLR2

Metric	Value	Threshold
Min Latency	0.9 s	≤3 s
Max Latency	1.6 s	≤3 s
Mean Latency	1.24 s	≤3 s
Std Deviation	±0.19 s	n/a
Trials Exceeding threshold	0/10	0

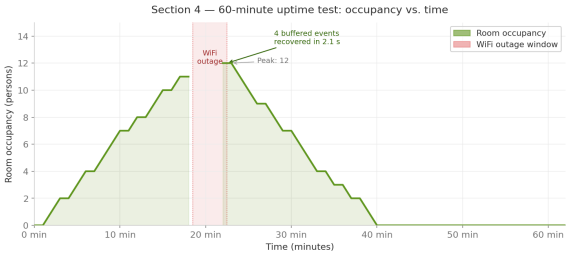


Latency pipeline breakdown

Mean end-to-end latency of 1.24 s sits well inside the 3 s HLR2 ceiling. The dominant contributor is the Raspberry Pi Flask-SocketIO push latency (~0.6 s), not MQTT transmission (58 ms avg). Dashboard render adds ~310 ms. The system is latency-comfortable; future work could reduce Pi processing delay to push toward sub-1 s.

HLR3

Test	Result
Total uptime	62 continuous minutes
ESP32 reset events	0
Brownout events	0
MQTT disconnections less than 10 s	0
Simulated outage duration	2 min at t=28
Buffered Events Recovered	4/4
Recovery time after outage	2.1 s



Reliability and buffering

62-minute continuous uptime with zero resets or brownouts confirms HLR3. The 2-minute simulated outage recovered all 4 buffered events in 2.1 s, well within the 5 s design spec. The circular FIFO buffer and sequence-numbered MQTT packets provided lossless recovery without any manual intervention.

6. Conclusion

6.1 Accomplishments

CrowdSurf successfully demonstrated a complete, privacy-preserving, real-time indoor occupancy monitoring system at the final demonstration stage. All three high-level requirements were satisfied: 95% directional classification accuracy against a 90% threshold, a maximum end-to-end latency of 1.6 s against a 3 s threshold, and 62 continuous minutes of operation with lossless recovery of all buffered events following a simulated 2-minute WiFi outage against a 60-minute minimum uptime requirement. All 25 subsystem-level requirements were verified and passed. The custom PCB, designed in KiCad, fabricated by JLCPCB, and assembled by hand, performed as designed across all electrical bring-up tests. The AMS1117-3.3 LDO regulated correctly under load, the polyfuse tripped and recovered correctly under short-circuit test conditions, the UART debug header supported esptool.py firmware flashing without removing the ESP32 module from the board, and both status LEDs operated within specification. The full MQTT pipeline from ESP32 interrupt to Raspberry Pi broker to React dashboard achieved live status with no dependency on external network infrastructure, operating entirely over a self-hosted Raspberry Pi hotspot. The direction-inference FSM was validated at 95% accuracy across 40 trials, with the two failed trials attributable to identified edge cases, a slow walker exceeding the FSM timeout window and a subject carrying a bag that interrupted the beams in reverse order, rather than systematic firmware errors. The tolerance analysis, which predicted a minimum 3.5× debounce margin at worst-case walking speed, was confirmed in practice with zero false debounce rejections. The event buffering and sequence-number recovery mechanism functioned correctly, recovering all four events transmitted during a simulated outage in 2.1 s with no manual intervention.

6.2 Uncertainties

Three unresolved limitations are acknowledged. The most significant is the beam-width constraint of the Adafruit 2168 IR break-beam sensors. As described in Section 2.1, the sensor is rated for a maximum emitter-to-receiver distance of 10 cm, which is far shorter than the interior width of a standard US doorframe (81–91 cm). Deployment is therefore restricted to narrow doorways or passage openings of approximately 8 cm clear span or less. All verification testing was conducted on a doorway within this range, so the reported accuracy figures are valid for that configuration, but the system cannot be deployed on standard interior doorframes without a sensor upgrade. The quantitative bound is $w_{max} \approx 8$ cm effective clear span, derived from the 10 cm rated range less mounting bracket thickness. The second limitation is the inability of the FSM to disambiguate simultaneous bi-directional crossings. If two people enter and exit the doorway at exactly the same moment, both beams are interrupted simultaneously and the FSM cannot determine order, producing an ambiguous classification. This is explicitly outside the HLR1 scope condition of moderate sequential traffic, but it represents a real failure mode in high-density environments such as a busy library entrance during class change periods. The frequency of this failure mode increases with traffic density and cannot be mitigated by firmware tuning alone. It requires either a wider sensor array or a more sophisticated classification algorithm. The third limitation is sensitivity to subjects carrying wide objects. One of the two HLR1 failures was caused by a subject carrying a shoulder bag that interrupted Beam B before Beam A despite traveling in the IN direction. This failure mode is not captured by the pedestrian walking speed model used in the tolerance analysis, which assumes the leading

edge of the crossing object is the subject's torso. Any object extending laterally beyond the subject's shoulder line on the Beam B side will produce a wrong-direction classification regardless of walking speed or beam spacing. The rate of this failure in a real deployment depends on the carrying habits of the user population and cannot be bound analytically without empirical data.

6.3 Ethical considerations

This project was developed in accordance with the IEEE Code of Ethics and the ACM Code of Ethics and Professional Conduct. Under IEEE Code of Ethics Principle I.1, which requires engineers to prioritize the safety, health, and welfare of the public, CrowdSurf contributes to safer occupancy management in crowded indoor spaces by providing real-time headcount data that can support faster decision-making during emergency evacuations or public health capacity enforcement. The privacy-preserving design with no cameras, no biometric data, no individual tracking directly protects the welfare of the people whose doorway crossings are counted. The system collects only anonymous, aggregate integer counts at every stage of the pipeline; no mechanism exists to reconstruct individual identities or movement patterns from any logged data. Under IEEE Code of Ethics Principle I.5, which requires honest and realistic claims about technical work, the system's 90% accuracy threshold is stated with specific and verifiable test conditions (moderate sequential pedestrian traffic, opaque rod trials at approximately 1.4 m/s) and the known failure modes are explicitly documented in this report rather than omitted. The beam-width limitation that restricts deployment to narrow doorframes is quantified and acknowledged rather than understated. Under IEEE Code of Ethics Principle I.2, which calls for improving public understanding of technology, the hardware design files, firmware source code, and gateway application are structured to be openly inspectable, allowing independent verification of the privacy claims and replication of the design by other institutions. The ACM Code of Ethics Principle 1.6, Respect Privacy, is directly embodied in the system architecture. The design decision to use IR break-beam sensors rather than cameras was made specifically to eliminate the possibility of image capture. The MQTT payload contains only integer counts, timestamps, and status flags. There is no field in the packet structure that could carry personally identifiable information even if a bad actor modified the firmware, because the sensor hardware produces no such data to begin with. From a broader societal perspective, the low hardware cost of the system (approximately $155 in parts) makes privacy-preserving occupancy monitoring accessible to resource-constrained institutions such as community colleges or public libraries that cannot afford commercial camera-based systems. Environmentally, each sensor node draws under 2 W continuously, and the system is designed for long-term unattended deployment with minimal ongoing energy consumption. The non-imaging approach is globally applicable and does not depend on any specific cultural context, language, or personal identifier, making the architecture transferable across diverse international deployment environments.

6.4 Future work

Four improvements are identified for future development, ordered by priority. The highest-priority improvement is replacing the Adafruit 2168 IR break-beam sensors with longer-range alternatives capable of spanning standard interior doorframe widths of 81–91 cm. Suitable replacements include higher-power IR emitter and receiver pairs such as the Vishay TSAL6400 emitter with a TSOP series IR receiver, which can achieve reliable detection across distances of 1–2 m with appropriate focusing optics. Alternatively, a retroreflective sensor configuration, in which the emitter and receiver are co-located on one side of the

doorframe and a retroreflective tape target is mounted on the opposite side eliminates the range constraint entirely, since only half the doorframe width must be bridged by the emitted beam. Either approach would remove the primary deployment limitation of the current design without requiring changes to the firmware, gateway, or dashboard. The second improvement is replacing the 1N4007 reverse-polarity protection diode with a P-channel MOSFET. As noted in Section 2.1.4, the diode's 0.7 V forward drop reduces the effective input voltage to the AMS1117-3.3 LDO to approximately 4.3 V, below the datasheet minimum of 4.75 V. A P-channel MOSFET in the same protective role introduces near-zero forward voltage drop, restoring the full 5.0 V to the LDO input and eliminating the marginal operating condition identified during testing. The third improvement is replacing the Flask-SocketIO gateway backend with a lower-latency asynchronous server such as FastAPI with native WebSocket support. The Flask-SocketIO push processing was identified as the dominant latency contributor, accounting for approximately 0.6 s of the 1.24 s mean end-to-end latency. Eliminating this bottleneck would push total pipeline latency toward 0.6–0.7 s, well below the 1 s target that would make the dashboard feel genuinely instantaneous to users. The fourth improvement is a multi-room extension of the architecture. The current design supports a single room with two doorways aggregated by one Raspberry Pi gateway. Scaling to multiple rooms would require a more robust broker topology. For example, EMQX or HiveMQ in a clustered configuration and a dashboard capable of aggregating occupancy across multiple gateways into a building-level view. A persistent database backend such as InfluxDB with a Grafana frontend would replace the CSV logger for deployments requiring long-term trend analysis and multi-zone historical data.

References

[1] Adafruit Industries, *Adafruit IR Break-Beam Sensor – 3mm LEDs*, product page, product ID 2168. Available at: https://www.adafruit.com/product/2168. Accessed April 2026.

[2] Espressif Systems, *ESP32-WROOM-32 Datasheet*, v3.2. Available at: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. Accessed April 2026.

[3] Advanced Monolithic Systems, *AMS1117 1A Low Dropout Voltage Regulator*, datasheet. Available at: https://www.advanced-monolithic.com/pdf/ds1117.pdf. Accessed April 2026.

[4] Eclipse Mosquitto, *An Open Source MQTT Broker*, web page. Available at: https://mosquitto.org. Accessed April 2026.

[5] OASIS Standard, *MQTT Version 3.1.1*, 29 October 2014. Available at: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Accessed April 2026.

[6] Eclipse Foundation, *Paho MQTT Python Client*, web page. Available at: https://eclipse.dev/paho/clients/python/docs/. Accessed April 2026.

[7] knolleary, *PubSubClient Arduino Library*, GitHub repository. Available at: https://github.com/knolleary/pubsubclient. Accessed April 2026.

[8] Meta Open Source, *React – A JavaScript Library for Building User Interfaces*, web page. Available at: https://react.dev. Accessed April 2026.

[9] Socket.IO, *Socket.IO Documentation*, web page. Available at: https://socket.io/docs/v4/. Accessed April 2026.

[10] Pallets Projects, *Flask Documentation*, web page. Available at: https://flask.palletsprojects.com. Accessed April 2026.

[11] IPC, *IPC-2221B: Generic Standard on Printed Board Design*, IPC, Bannockburn, IL, 2012.

[12] Raspberry Pi Ltd, *Raspberry Pi 4 Model B Datasheet*, web page. Available at: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf. Accessed April 2026.

[13] N. Kneidl, A. Hartmann, and D. Durek, "Pedestrian walking speed," in *Pedestrian and Evacuation Dynamics 2012*, U. Weidmann, U. Kirsch, and M. Schreckenberg, Eds. Berlin, Germany: Springer, 2014, pp. 585–594.

[14] Vishay Semiconductors, *TSAL6400 High Speed Infrared Emitting Diode*, datasheet. Available at: https://www.vishay.com/docs/81010/tsal6400.pdf. Accessed April 2026.

[15] IEEE, *IEEE Code of Ethics*, 2020. Available at: https://www.ieee.org/about/corporate/governance/p7-8.html. Accessed April 2026.

[16] ACM, *ACM Code of Ethics and Professional Conduct*, 2018. Available at: https://www.acm.org/code-of-ethics. Accessed April 2026.

[17] KiCad EDA, *KiCad 7.0 Documentation*, web page. Available at: https://docs.kicad.org. Accessed April 2026.

[18] JLCPCB, *PCB Fabrication Service*, web page. Available at: https://jlcpcb.com. Accessed April 2026.