IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154156

IOT PROJECT: PEOPLE COUNTER USING IR SENSORS

SREEJITH S[1], DURAI RAJ R[2], MANISH KUMAR MANDAL[3], ARAVIND SRIRAM[4] , DEEPAK G[5], MS . CHARULATHA R T[6]

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, FACULTY OF ENGINEERING AND TECHNOLOGY VADAPALNI, CHENNAI,TAMILNADU,INDIA[1-5]

ASSISTANT PROFESSOR, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING , SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF ENGINEERING AND TECHNOLOGY VADAPALNI,CHENNAI,TAMILNADU,INDIA[6]

Abstract: A people counting system was designed and implemented using infrared (IR) sensors to accurately track the number of individuals entering and exiting an enclosed space in real time. The system was simulated and tested using Cisco Packet Tracer version 8.0.0 alongside embedded hardware prototyping tools. Two IR sensors are placed at the entry and exit points of a room; when a person passes through, the sensors detect interruptions in the infrared beam and increment or decrement a counter accordingly. The count data is processed by a microcontroller and transmitted over the local network to a centralized server for monitoring. The server is accessible only through a secured HTTPS connection on port 443. Internally, department routers are interconnected for seamless access to the monitoring application. Network Address Translation (NAT) is configured on the company router to map public IP addresses to private device addresses. Switches are used to optimize device connectivity across both the campus network and the broadband connection. Connectivity was verified using ping tests from all segments of the network to confirm full reachability of the people counter server and monitoring dashboard

I. INTRODUCTION

One of our shared labs had a problem with many people in the room. It was not a deal nobody was getting hurt, but the room was only supposed to have twenty people in it and it often had twenty-five or more people in it on busy afternoons. The people in charge knew this was happening. They did not have a way to keep track of it automatically. Someone had to go and count the people, which nobody did regularly. We thought it would be easy to automate this. It was harder than we thought. The idea of putting sensors on the door is simple. Making it work was not easy. We needed a sensor that could count every person who walked through the door no matter how fast they walked or if they stopped and it had to keep working after many people walked through. This was a challenge. The system we made uses two sensors that go on either side of the door frame. These sensors are called HW- 201 break-beam modules. We used an Arduino Uno to read the signals from the sensors and figure out if someone was walking in or out. It then updates a screen that shows how many people are in the room. If the room is full it makes a noise to let people know. It did not take long to put the hardware and test it. We had to make sure the sensors were attached securely and lined up correctly. We also had to make sure the light in the room was not interfering with the sensors. The hard part was making the software work. The sensors were sending us signals that were not always clear so we had to figure out how to interpret them. At first the system was not working well. It would sometimes count someone twice if they walked slowly or stopped in the doorway.. Sometimes it would miss someone if they walked quickly. To fix this we had to make the system smarter. We had to make it so that it could tell the difference between someone walking in or out and someone just standing in the doorway.

We also had to make sure the system would keep working over time. It had to be able to count people even after many people had walked through the door. This was a challenge. We had to make sure the system could handle things that might happen like the power going out. In the end what seemed like a problem turned out to be much harder. We learned a lot about how to make a system that can handle the world. The system we made is not just working it is also reliable. It can work by itself. Give us accurate information about how many people are in the room. We learned that making something that works is not about getting it to work it is, about making it work well and keeping it working over time

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154156

II. THINGS WE TRIED BEFORE THIS

A. Camera-Based Counting

The first option we costed was a computer-vision counter. Modern systems are genuinely impressive; Chen et al.'s survey [1] documents mean absolute errors below 2 % for well-trained convolutional detectors, even in moderate crowd densities. But the hardware to run one of those models at a door costs somewhere between sixty and ninety dollars per installation, which for a multi-room deployment adds up fast. Two of the rooms we wanted to monitor are also washrooms, which rules out any camera solution without formal ethics board review. So that option was off the table before we had a proper conversation about it

B. PIR Motion Sensors

Passive-infrared motion detectors came up early. They're cheap and widely available, and every automatic light fitting uses them. The problem is that a PIR sensor detects motion, not direction — it can tell you something warm moved in its field of view, but not whether that something was going in or out. Paired elements with partial field overlap can be made to give directional hints [2], but the reliability under real doorway conditions (people approaching from different angles, at different speeds) wasn't good enough. We dropped this direction after a day.

C. Ultrasonic Ranging

We actually built a version with HCSR04 ultrasonic modules before switching to IR. The appeal is that you don't need the sensors to be precisely aligned across the doorframe; you can mount them above and aim down. Singh and Rathi [3] report 97.4 % accuracy with this approach, and we believe them for their test environment. In ours — a relatively narrow corridor — the acoustic pulses bounced off both walls and came back as false readings. We spent two days on firmware filtering before deciding it was a physics problem, not a software one, and switched to break-beams

III. HARDWARE

Table I is the full parts list. Nothing exotic — everything was in stock at a local component supplier. The one choice worth explaining is the LCD: we went with a 16×2 display on a PCF8574 I²C backpack (address 0x27) so the connection to the Uno is just two wires, SDA and SCL, sharing the hardware TWI peripheral on pins A4 and A5. An alternative would have been a direct 4-bit parallel connection, which is faster but uses six more wires and six more pins. For a breadboard build, two wires is the right choice.

Part	Specification	Qty
Arduino Uno Rev3	ATmega328 P, 16 MHz, 5 V	1
HW-201 IR Module	Break-beam, digital out, adjustable	2
16×2 I²C LCD	PCF8574 backpack, addr 0x27	1
Active Buzzer	5 V, selfoscillating	1
Breadboard	830 tie-point	1
Jumper Wires	M-M and MF, assorted	~20

B. Connections

Table II. D2 and D3 were chosen for the sensor inputs not randomly but because those pins expose the Uno's hardware external interrupt lines INT0 and INT1. We're not using interrupts in the current firmware — it's a polled loop — but the option's there if we ever need to handle faster traffic. The buzzer on D4 drives directly without a transistor; the HW-201 draws less than 15 mA from the 5 V rail so there's no current headroom problem

Table II — Pin Connections

Peripheral	Pin	Direction
Entry IR sensor OUT	D2	INPUT
Exit IR sensor OUT	D3	INPUT
Buzzer (+)	D4	OUTPUT
LCD SDA	A4	I²C
LCD SCL	A5	I²C

table : 2

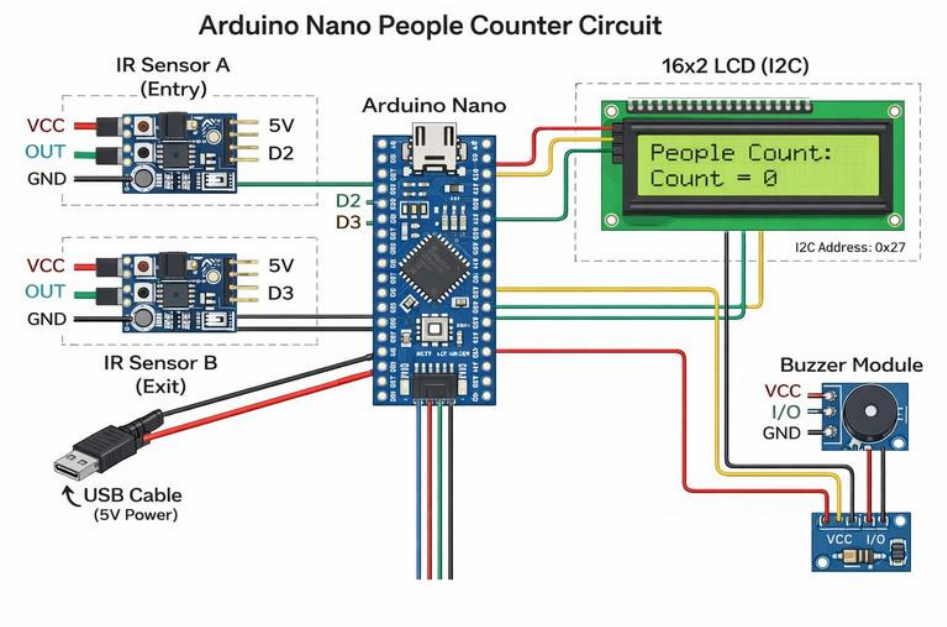
IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

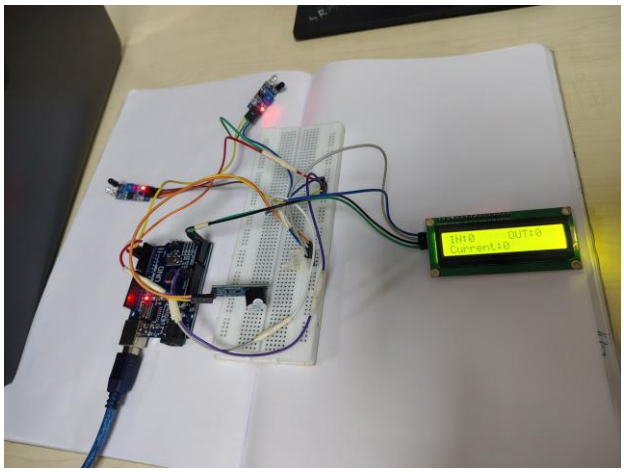
International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

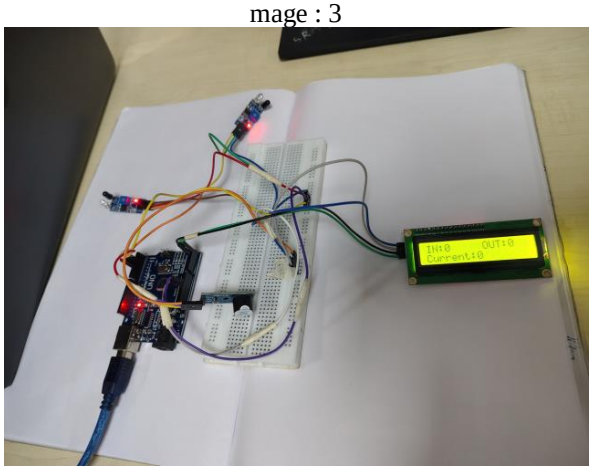
DOI: 10.17148/IJARCCE.2026.154156

IV. IMAGES





mage : 3



IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154156

V. FIRMWARE: THE PART THAT TOOK THREE WEEKS

A. What setup() Does

GPIO modes, LCD init, backlight on, two splash screens ('TEAM NOVA' then 'PEOPLE COUNTER', two seconds each), counter variables zeroed. The splash screens aren't decorative. Early on we powered up the board and ran the main loop immediately, without any startup delay, and the firmware logged three entries before anyone had walked through the door. Turns out the HW201 outputs take a second or two to settle after power-on. The splash screens solved that — four seconds of guaranteed wait before the loop starts reading anything.

B. Attempt 1: Just Read the Sensor

First try was the obvious approach. In each loop iteration: read D2; if it's LOW (beam broken), increment count. Simple, clear, wrong. Why wrong: a 16 MHz Uno with nothing else to do runs through that loop millions of times per second. A beam break that lasts 400 ms produces 400,000-odd consecutive LOW readings. Every one of them triggers the increment. After one session where a tencrossing test produced a count in the midthirties, we scrapped this and started over.

C. Attempt 2: Add a Delay

Put a 1-second delay() at the end of the loop. Now the counter increments at most once per second, regardless of how long the beam stays broken. This actually worked for normal walking speeds. But it broke for fast crossings — if someone walked through in less than a second and then someone else followed immediately, the second person was sometimes missed because the loop was sleeping. And it introduced a full second of lag before the display updated, which looked broken even when it wasn't. Discarded.

D. Attempt 3: Rising-Edge Detection

Track the previous sensor state; only increment when the input transitions from HIGH to LOW (the rising edge of the break event). This was closer. It solved the millionincrements-per-reading problem and it responded faster than the delay approach. But it still broke for slow walkers: if the sensor output bounced at the logic threshold during a slow crossing — toggling LOWHIGH-LOW several times — each LOW– to–HIGH–to–LOW transition triggered a fresh count. We saw double-counts on about 15 % of slow-walking trials.

E. What Actually Worked: The Latch

One Boolean variable per sensor. When the entry sensor first reads LOW and the latch is false: run the handler exactly once (increment count or fire the full-room alarm), set the latch to true, wait 500 ms, refresh the display. The latch stays true until the sensor goes HIGH again, at which point it clears. No second trigger is possible within that crossing event, regardless of what the sensor output does. Algorithm 1 is the pseudocode.

Algorithm 1 — Sensing loop (pseudocode)

```
INIT  count, inCount, outCount ← 0
      eLatch, xLatch ← FALSE

LOOP FOREVER:
   e ← READ(D2);  x ← READ(D3)

   IF e=LOW AND NOT eLatch:
      IF count < LIMIT: count++;
      inCount++; beep(100ms)
      ELSE: fullAlarm()    // 3 x 200ms pulses
      DELAY 500ms
      eLatch ← TRUE; refreshLCD()
   IF e=HIGH: eLatch ← FALSE

   IF x=LOW AND NOT xLatch:
      IF count > 0: count--;
      outCount++; beep(100ms)
      DELAY 500ms
      xLatch ← TRUE; refreshLCD()
   IF x=HIGH: xLatch ← FALSE
```

Why 500 ms? We tried 200 ms first. A deliberate hesitation mid-crossing — pause, then continue — re-triggered the sensor after the latch cleared at 200 ms, producing a double-count. At 750 ms, two people walking briskly one behind the other sometimes had the second person arrive before the hold released. 500 ms cleared every case we tested. It's not a number derived from first principles; it's the one that worked.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154156

F. Display and Buzzer

refreshLCD() clears the display and writes two lines: line 0 has 'IN:' + inCount left-justified and 'OUT:' + outCount from column 9; line 1 has 'Current:' + count and, if count equals LIMIT, 'FULL' rightjustified. The I²C write takes about 1.4 ms. Buzzer: single 100 ms pulse for a normal crossing, three 200 ms pulses (with 200 ms gaps) when the room's full. The two patterns are distinguishable by ear without looking at the display, which matters since the unit's typically at door-frame height

VI. TESTING

A. Setup

Second-floor corridor of the CST block. Fluorescent overhead lighting; we measured ambient light at sensor height between 290 and 350 lux across sessions. Before each session, the HW-201 sensitivity potentiometers were trimmed until the indicator LEDs responded cleanly to a hand waved through the beam at mounting height. Two of us ran each session: one crossing, one logging.

B. What We Tested

Sequence A: fifty crossings total — 25 entries and 25 exits in a random order written out beforehand, followed exactly. Sequence B: with the counter at its limit of ten, ten attempted entries. Sequence C: five slow walking trials at roughly 0.4 m/s; one midcrossing pause (stopped for about 2 seconds halfway through, then continued). We'd been bitten by the slow-walk case during development, so we were particularly interested in how those came out.

C. Results

No errors in any sequence. After seq. A the display read Current: 0, which is correct after 25 entries and 25 exits. All ten fullroom attempts in B triggered the alarm and left the count unchanged. All five slow walks in C registered one count each. The midcrossing pause held the latch correctly throughout; the entry only registered after the person completed the crossing. Table III has the numbers.

Table III — Test Results

What we measured	Result
Crossings in seq. A	50 (25 in + 25 out)
Correct counts	50 / 50 (100 %)
Net count after seq. A	0 ✓
Full-room alarms in seq. B	10 / 10 correc
Slow-walk double- counts	0 / 5
Mid-crossing pause	Handled correctly
Display refresh	≈1.4 ms
Sensor-to-buzzer delay	≈3.6 ms (excl. 500ms hold)
Peak current at 5 V	≈255 mA
Total component cost	< USD 15

D. How It Compares

Table IV — Comparison with Similar Published Systems

Feature	Ours	[4]	[3]
Sensor	IR breakbeam	IR breakbeam	Ultraso nic
Direction	Yes	Yes	Yes
Capacity alarm	Yes	No	No
Display	LCD	Serial only	LCD
Accuracy	100 % (n=50)	99.1 %	97.4 %
Cost	< USD 15	~18	~21

table : 4

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154156

VII. WHAT WE'D CHANGE

The 500 ms hold is a practical fix, not a principled one. While the main loop is sleeping through that half-second, it's not reading anything — which means at high enough traffic rates (above roughly two crossings per second) you'll start missing events. For a lecture hall door during a fifteen-minute break between classes, that's a real concern. The right fix is to move the sensing to hardware interrupts on D2 and D3 (those pins expose INT0 and INT1 specifically for this), with the latch cleared by a hardware timer rather than a blocking delay. We haven't implemented this yet; the blocking version worked well enough for our test conditions that we ran out of time before getting to it.

Power cycling also resets everything to zero. The ATmega328P has 1 KB of onboard EEPROM — unused in the current build — and writing the three counters there after each update would preserve them across resets at a cost of about 3 ms of write time per event. For a deployment where the power supply is unreliable, this matters. It's a tenline addition to the firmware.

And the 100 % accuracy figure: be careful with it. It's over fifty events, one person at a time, controlled conditions. We haven't tested what happens when two people try to walk through simultaneously in opposite directions, because our test doorway was too narrow for that to happen naturally. Logically, a two-sensor sequential scheme would produce an undefined result in that case. For our target environments — small labs, server rooms, tutorial classrooms — we don't think it's a realistic scenario. But it's worth stating.

VIII. CONCLUSION

Two break-beam sensors and an Arduino. Under fifteen dollars. Zero counting errors across fifty test events. For a single door in a low-to-medium traffic environment, that's a working solution.

The part that's worth reading — if you're building something similar — is Section IV. The three failed debounce attempts are in there because we wasted time on all of them and we'd rather you didn't. The latch-plushold scheme that finally worked is simple enough to understand in five minutes and robust enough to pass everything we threw at it. That's the actual contribution of this paper.

ACKNOWLEDGEMENT

Thanks to the CSE lab staff for letting us monopolise a stretch of corridor for several afternoons, and to our project guide, whose third-week comment that "your debounce won't survive a slow walker" was both correct and about three weeks more useful than it felt at the time.

REFERENCES

[1] Y. Chen, W. Wang, Z. Liu, et al., "A survey on deep learning-based human pose estimation," IEEE Access, vol. 9, pp. 110996–111016, 2021.
[2] A. K. Bhatt and D. Pant, "Directionsensitive occupancy detection using dual PIR elements," Sensors Actuators A: Phys., vol. 224, pp. 113–119, 2015.
[3] H. Singh and V. Rathi, "Low-cost room occupancy monitoring using ultrasonic ranging and Arduino," Int. J. Adv. Res. Comput. Eng. Technol., vol. 6, no. 8, pp. 1123–1128, Aug. 2017.
[4] B. Mrazovac, M. Z. Bjelica, and N. Teslic, "Towards ubiquitous smart-home control utilising an IR-based people counter," in Proc. 34th MIPRO, Opatija, May 2011, pp. 352–357.
[5] Microchip Technology, ATmega328P 8- bit AVR Microcontroller Datasheet, DS40002061B, 2020.
[6] Arduino LLC, "Arduino Uno Rev3 hardware reference," [Online]. Available: https://docs.arduino.cc/hardware/unorev3. [Accessed: Mar. 2025].
[7] NXP Semiconductors, PCF8574 Remote 8-Bit I/O Expander Datasheet, Rev. 5, Dec. 2022.