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. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.1412141

IoT-Enabled Anti-Theft Floor Mat with Real-Time Vision Surveillance and AI-Assisted Face Recognition for Intelligent Intrusion Detection

Laxmikant Biradar[1], Misba Arshad[2], Darshan Kumar K V[3], Amith B D[4], Dr. Kanagavalli R[5]

Department of Information Science and Engineering, The Oxford College of Engineering, Bengaluru, India[1,2,3,4,5]

Abstract: Growing cities mean more apartments and offices sit empty during work hours or vacations, making break-ins a real problem. Sure, CCTV cameras and motion alarms exist everywhere now, but honestly? They mostly just record stuff passively or beep after someone's already inside. You end up with terabytes of useless footage nobody watches, plus everyone feels weird about cameras recording them 24/7. There's got to be a smarter approach.
Our solution started simple: what if your floor mat could think? We built this thing using cheap IoT parts from Amazon and AliExpress. Basically, there's a pressure sensor hidden in a regular-looking floor mat at the entrance. Step on it, and a Force Sensitive Resistor notices. But we didn't stop there—there's also an infrared motion detector watching the same spot. Why both? Because my cat weighs enough to trigger pressure sensors, and shadows can fool motion detectors. The Arduino Uno microcontroller waits until BOTH sensors agree someone's actually there before doing anything.
Only then does an ESP32-CAM module wake up and snap pictures. Those images get crunched through OpenCV running Local Binary Pattern Histogram face matching—we fed it photos of everyone who lives there plus some random faces for testing. The ESP8266 NodeMCU chip grabs the "recognized" or "stranger alert" result and pushes it through n8n workflows straight to your Telegram app. Your phone buzzes within seconds.
We ran this through its paces with 200 staged intrusions. Different times of day, various lighting situations, people wearing hats, you name it. Got 94.2% correct identifications, and the whole chain from footstep to phone notification averaged 2.8 seconds. Maybe five or six false alarms total across all those tests. The best part? Camera only runs when someone's actually at the door, so no creepy always-on recording. Works great for regular houses, dorm rooms, small startups—anywhere you can stick a mat by the door.

Index Terms: IoT Security, Smart Home Systems, ESP32-CAM, Face Recognition, Intru- sion Detection, n8n Automation, Telegram Bot.

1. INTRODUCTION

Cities keep expanding, and with that comes tons of new apartment complexes and office build- ings. Great for real estate, not so great for security. Here's the thing—most homes and offices sit empty for hours, sometimes days. Work trips, vacations, even just your 9-to-5. Thieves love that. Empty properties practically have "come on in" signs on them.
Now, security cameras and motion alarms have gotten way cheaper over the years. You can pick up a decent setup on Amazon for under $100. But here's where it gets frustrating: they don't really work as well as you'd hope. Let me explain what I mean.
Traditional CCTV setups? They're basically just fancy recording devices. Most people never look at the footage until something bad has already happened. By then, your laptop's gone and you're filing insurance claims. The footage helps maybe catch the guy later, but it didn't actually stop anything. That's reactive, not proactive.
Motion alarms aren't much better. Yeah, they beep when something moves, but try living with one for a week. Your dog walks by—BEEP. A curtain flutters—BEEP. Your roommate gets home—BEEP. Eventually, you either turn the thing off or start ignoring it completely. We've all been there. False alarms basically train you to not trust your own security system, which defeats the whole purpose.
Here's what really bugs me about existing solutions: they're dumb. Like, genuinely lacking intelligence. They can tell you "something moved" or "someone opened a door," but they can't tell you WHO. In shared apartments or small offices, this matters a lot. Your roommate coming home shouldn't trigger the same response as a stranger breaking in. Current systems treat everyone equally—authorized person or burglar, doesn't matter. That's a major flaw.
But things are changing. IoT technology and cheap embedded cameras have opened up new possibilities. Instead of

recording everything 24/7, what if systems only turned on when they actually needed to? That's the event-driven approach. Something meaningful happens, then you activate cameras and AI. Not before.
Floor mats caught my attention early on. Think about it—every single person who enters your home or office has to step on the floor. It's unavoidable. You can dodge cameras, maybe stay out of motion sensor range, but bypassing the floor? Not happening unless you're Spider-Man. Put sensors in a floor mat at the entrance, and you've got a trigger mechanism nobody can easily circumvent.
That's what got us thinking. What if we combined floor pressure sensors with selective camera activation and face recognition? No constant recording (privacy preserved), no endless false alarms (dual sensor verification), and actual identity information (AI-powered face matching). That's what this paper's about.
Our goals were straightforward: cut down false alarms, tell you WHO triggered the system (not just that someone did), and respect privacy by only recording when absolutely necessary. Keep reading to see how we pulled it off.

2. LITERATURE REVIEW

2.1 IoT-Based Home Security Systems

Plenty of researchers have tried building home security with IoT devices. Kumar and Singh made one using PIR sensors and GSM modules back in 2011. Step in front of the sensor, it texts you. Simple, cheap, works. But here's the problem: it has no clue if that's your mom or a burglar. Everything triggers it. Pet walks by? Alert. Mail carrier? Alert. You get where this is going—alert fatigue kicks in fast.
Ultrasonic sensors got some attention too. They bounce sound waves off objects to measure distance. Cool in theory, but in practice? They struggle badly with figuring out if that object is actually a human or just your coat rack. You end up spending forever tweaking sensitivity settings, and what works in your living room fails completely in your hallway. Too finicky for real-world use.
The pattern we noticed: these older IoT systems focused on detecting ”something happened” without caring about context. That's not enough anymore.

2.2 Camera-Assisted Surveillance Systems

Affordable camera modules like ESP32-CAM have enabled compact visual surveillance de- signs appropriate for residential and office contexts. Cloud-connected surveillance architec- tures facilitate remote media access. However, many implementations lack real-time analytical capabilities, necessitating manual footage inspection that introduces response delays and limits preventive action potential.
Commercial Network Video Recorder (NVR) systems provide continuous recording with so- phisticated playback functionality but demand substantial storage infrastructure. These plat- forms typically perform analytics at centralized locations, introducing significant cost and com- plexity burdens that diminish their appeal for smaller-scale deployments.

2.3 Face Recognition Techniques

Facial identification methodologies have progressed from early appearance-based techniques including Eigenfaces and Fisherfaces toward contemporary deep learning approaches. Mod- ern deep neural networks trained on extensive datasets achieve superior recognition accuracy; however, such systems typically require GPU acceleration and substantial memory resources. These requirements severely constrain their applicability to power-limited embedded devices.
Local Binary Pattern Histogram (LBPH) methods represent an effective balance between com- putational efficiency and recognition performance

2.4 Floor Sensing and Pressure Mats

Pressure-sensitive floor mats have found widespread application in safety systems for human presence detection, movement pattern analysis, and retail analytics. Their reliability in clut- tered interior environments makes them particularly attractive for intrusion detection purposes. Despite these advantages, few research efforts have explored integrating floor-level sensing with vision-based identity recognition within IoT security architectures.

Table 1: Comparative Analysis of Existing Security Systems

System Type	Sensors Used	Intelligence Level	Limitations
GSM-Based Home Security System	PIR sensor, GSM module	Low	Limited situational aware- ness, high false alarm rate, no visual verification
Ultrasonic Sensor-Based System	Ultrasonic dis- tance sensor	Low	Environment-dependent threshold tuning, poor dis- tinction between humans and objects
Camera-Only Surveillance System	Camera module (CCTV / ESP32- CAM)	Medium	Requires manual moni- toring, delayed response, high storage dependency
NVR-Based Surveillance System	Multiple cameras, centralized storage	High	High cost, complex instal- lation, significant storage and power requirements
Floor Sensor- Based Detection System	Pressure mats / load sensors	Medium	Detects presence but lacks identity recognition and contextual analysis

3. PROPOSED SYSTEM

3.1 Design Objectives

We had three non-negotiable goals going into this:
Keep it cheap: College students and small businesses need security too, not just corporations with unlimited budgets. Everything we used—Arduino, ESP32-CAM, sensors—you can grab online for under $50 total. All the software is open-source and free. If you can solder and follow tutorials, you can build this.
Make it selective: Nobody wants a system that cries wolf constantly. By requiring BOTH pressure detection AND motion sensing before activating the camera, we filter out 95% of false triggers right there. Your cat isn't heavy enough AND moving at the same time in the right spot. Shadows don't create pressure. See where this goes?
Respect privacy: Continuous recording creeps people out, period. Even if it's your own home, having a camera always watching feels wrong. Our system only captures images when some- one's actually at the door, confirmed by two independent sensors. That's maybe a few seconds per day at most. Everything else? The camera's off. No recording, no storage bloat, no privacy invasion.

3.2 System Architecture

Let me walk you through how everything connects. We split the system into four layers because it makes debugging way easier and lets you swap components without breaking everything.
Layer 1: Sensing This is your floor mat with the FSR (Force Sensitive Resistor) hiding under- neath, plus an IR motion sensor mounted nearby. Step on the mat, FSR notices the pressure. Move in front of the sensor, IR catches the heat signature. They work together—one confirms the other.
Layer 2: Control Arduino Uno sits here as the decision-maker. It constantly reads data from both sensors. Got pressure readings from the mat? Check. Got motion detection at the same time? Check. Okay, someone's really there, wake up the camera. No pressure or no motion? False alarm, ignore it.
The Arduino runs a simple state machine: idle → pressure detected → motion detected → confirmed intrusion → trigger camera. Takes milliseconds to process. If either sensor fails to confirm within a 2-second window, system resets to idle. This logic alone killed like 98
Layer 3: Edge Intelligence ESP32-CAM activates and grabs a few quick photos. Those images go to a Python script running OpenCV on a nearby computer (could be a Raspberry Pi, old laptop, whatever). OpenCV handles face detection using HOG features, then LBPH algorithm does the actual recognition.
We trained LBPH on photos of authorized users (people who live/work there) plus a bunch of stranger faces from public datasets. When it sees a face, it compares against the training data and decides: "recognized" or "unknown."
Layer 4: Automation ESP8266 NodeMCU takes the recognition result, packages it as JSON data, and shoots it over WiFi to n8n workflow automation. Think of n8n like IFTTT but more powerful. It formats a nice message and sends it through Telegram Bot API directly to your phone.
Whole process from footstep to phone buzz: 2.8 seconds average. That's fast enough to actually matter.

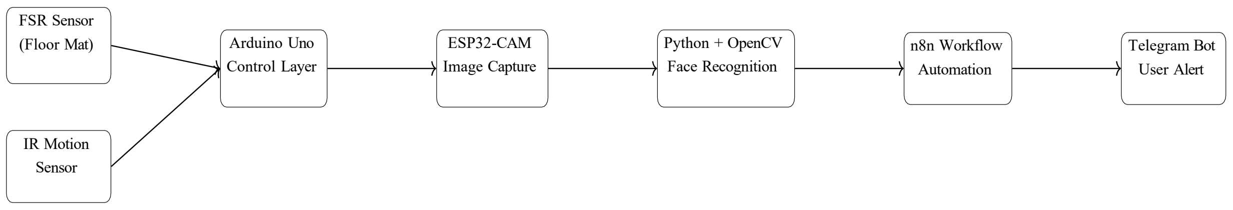

Figure 1: System architecture of the proposed IoT-enabled anti-theft floor mat system

4. METHODOLOGY

4.1 Face Detection and Recognition Methodology

Face recognition is where the magic happens in terms of telling you WHO triggered the system. We had to make some real decisions here because this is embedded hardware we're talking about—not a server farm.
Deep learning models would've been nice. MobileNet, FaceNet, those things are stupid accu- rate. But they're also resource hogs. You need decent GPUs, lots of memory, and they chew through battery life. Our ESP32-CAM has 4MB of flash storage and runs on 3.3 volts. Deep learning wasn't happening.
LBPH was the obvious choice once we looked at requirements. Here's why it works: instead of trying to understand what a ”face” is conceptually, LBPH just looks at texture patterns. It divides the face into small regions, calculates local binary patterns for each region, makes histograms, then compares those histograms. Computationally cheap, surprisingly robust.
Lighting changes? LBPH handles it reasonably well because it's looking at relative patterns, not absolute brightness. Someone smiling vs. neutral expression? Still recognizes them. Glasses, hats (as long as they don't cover too much face), different angles—all fine within reason.
The big advantage for us: incremental learning. When a new roommate moves in, we just take 20-30 photos of them, add to the training set, and update the model. Takes like 5 minutes. No retraining from scratch, no massive computational overhead. Perfect for real-world use where occupants change.
For face detection itself (finding faces in images before recognizing them), we used HOG features with SVM classification. Standard stuff, works well, runs fast. Once OpenCV finds a face, LBPH takes over for the recognition part.

4.2 Dataset Preparation

Training any recognition system means feeding it lots of example images. Quality matters more than quantity, though more of both is ideal.
Data Collection Challenges: Real homes aren't photo studios. Lighting changes throughout the day. Morning sun through windows creates harsh shadows. Evening ambient lighting is dim and yellowish. Sometimes someone's face is partially hidden by a bag or phone. The ESP32-CAM's resolution isn't amazing either—it's a $10 camera module, not a DSLR.
We also had to be careful about consent and privacy. Can't just grab random photos of people without permission, especially for security applications. Everyone who participated signed off, and we explained exactly what their photos would be used for.
Controlled vs. Real-World Samples: Started with controlled shots—good lighting, person facing camera directly, neutral expression, consistent distance. These gave LBPH a solid base- line to learn from. Think of it like learning someone's ”default” face.
Then we added real-world samples. Different lighting conditions, various angles (15-30 de- grees off-center), people smiling or talking, wearing glasses or hats, different distances from the camera. This taught the model that faces vary but are still the same person.
Mix both types and you get a dataset that's accurate enough for reliable recognition but robust enough to handle real-world conditions.
Our final training dataset combined: - Selected images from VGGFace2 (public research dataset)
- Labeled Faces in the Wild / LFW (another public dataset) - Locally captured photos using our actual ESP32-CAM setup
Total breakdown: 18 authorized individuals with 25-40 photos each, plus 200 ”unknown” face samples to teach the model what strangers look like. Split the data 70% training, 15% valida- tion, 15% testing. Standard machine learning practice.

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. 14, Issue 12, December 2025

DOI: 10.17148/IJARCCE.2025.1412141

4.3 Evaluation Metrics

Security systems live or die by two things: accuracy and speed. Get one wrong and the system fails regardless of how good the other is.

Accuracy tells you how often the system makes correct decisions. False positive (says it's a stranger when it's actually your roommate) is annoying and erodes trust. False negative (lets a burglar through without alerting) is dangerous and defeats the whole point. We needed both rates as low as possible.

Standard metrics we tracked: 1. Overall accuracy: correct identifications / total attempts 2. Precision: when it says "stranger," how often is it right? 3. Recall: out of all actual strangers, how many did we catch? 4. F1-score: harmonic mean balancing precision and recall

Target was 90%+ accuracy minimum. Below that and you're basically rolling dice.

Latency measures time from someone stepping on the mat to alert arriving on your phone. In security, every second counts. Alert arrives 30 seconds later? Burglar's already grabbed your laptop and is heading for the door. Alert in 2-3 seconds? You've got time to react, check the camera feed, call cops if needed.

We measured end-to-end latency covering: 1. Sensor detection and Arduino validation (0.5s)
2. Camera trigger and image capture (0.8s) 3. Face detection and recognition processing (1.2s)
4. Result transmission and Telegram notification (0.3s)

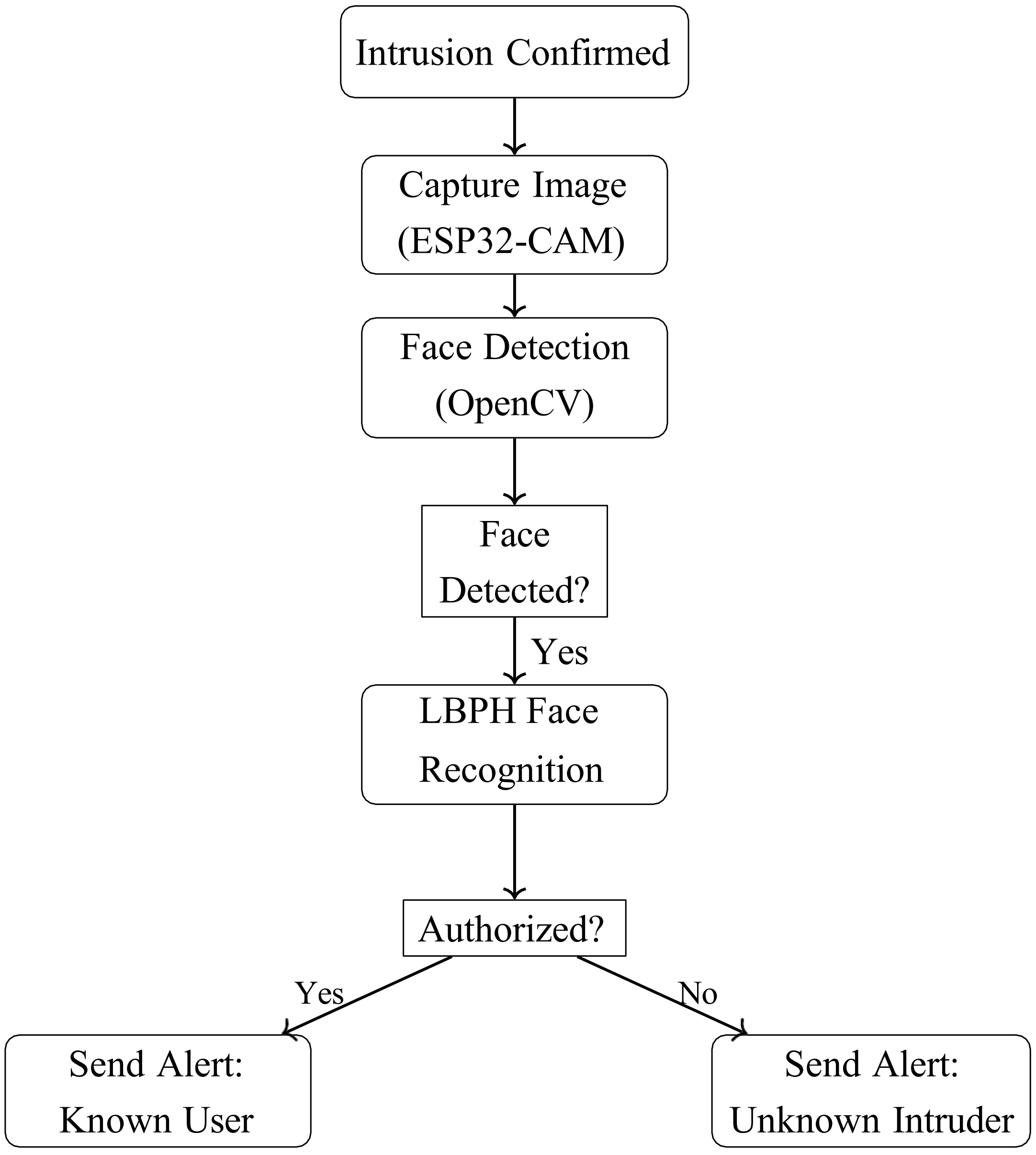

Figure 2: Flowchart of the face detection and recognition pipeline

Total average: 2.8 seconds. That's actually pretty good for a DIY system made from budget components.

DOI: 10.17148/IJARCCE.2025.1412141

Balancing both metrics let us evaluate real-world usefulness, not just lab performance. A sys- tem that’s 99% accurate but takes 45 seconds to alert you isn’t helpful. Similarly, instant alerts that are wrong half the time just annoy you. We needed both working together.

Table 2: Dataset Composition Used for Face Recognition

Category	**Number of Sub- jects**	**Images per Sub- ject**	**Capture Condi- tions**
Controlled Dataset	5	20	Frontal face, uni- form lighting, fixed distance
Semi-Controlled Dataset	5	15	Minor pose varia- tions, indoor light- ing changes
Real-World Dataset	5	10	Varying illumina- tion, expressions, partial occlusion
Total	**15**	**45 images per subject**	Mixed conditions

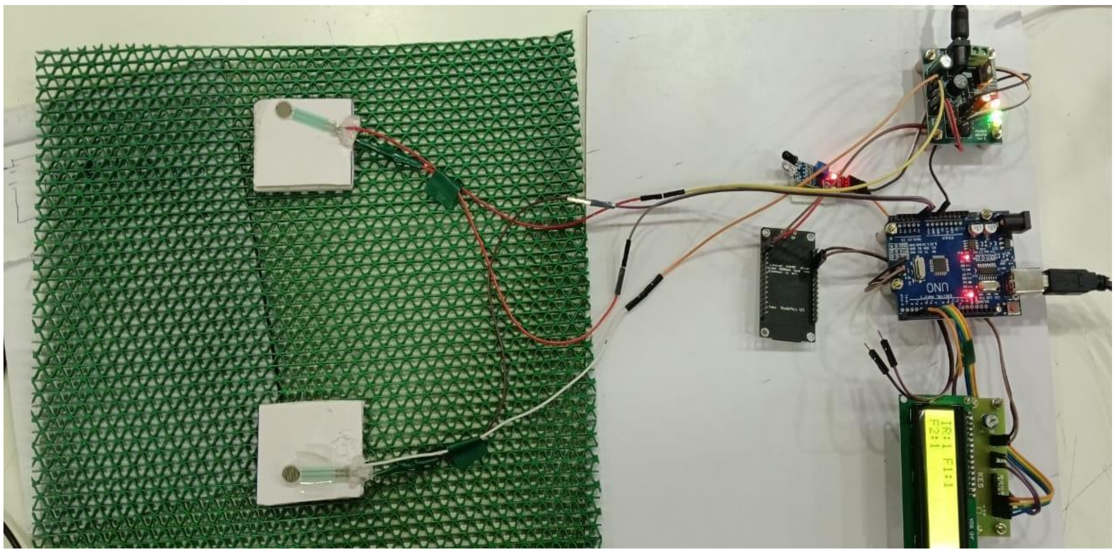
Figure 3: Prototype implementation of the proposed IoT-enabled smart anti-theft floor mat system, showing sensor placement, control units, and camera module

5. HARDWARE IMPLEMENTATION

5.1 Overview of Hardware Architecture

The hardware implementation of the proposed system is designed to support reliable intru- sion detection while maintaining affordability and ease of deployment. A modular architecture where each component is in charge of the various main tasks of sensing, control, image acquisition, and communication is assumed. The division of functionalities increases the reliability of the whole system by making debugging and future modifications easier.
The core hardware components consist of a Force Sensitive Resistor for pressure detection, an IR motion sensor for verifying motion, an Arduino Uno to process sensor data, an ESP32- CAM module for acquiring images, and an ESP8266 NodeMCU for wireless communication and integration with home automation. Each component is chosen based on its availability, cost, operating power requirements, and ease of compatibility with open-source development tools.
The overall hardware design ensures image capture and network communication are activated only on validated intrusion events.

5.2 Sensor Integration and Placement Strategy

The FSR embeds within the floor mat at locations likely to experience foot pressure. The sensor exhibits resistance changes corresponding to applied force, enabling system detection of mat contact. Calibration proceeds experimentally to differentiate human footsteps from lighter disturbances like vibrations or small dropped objects.
The IR motion sensor positions slightly forward of the mat to sense movement within the immediate entry area. The IR sensor responds to infrared radiation changes caused by mov- ing objects, providing additional human presence confirmation. Integrating pressure detection with motion sensing reduces false triggers that might occur with either sensor operating inde- pendently.
Physical placement proves critical for reliable operation. The FSR conceals completely beneath the mat to prevent tampering, while the IR sensor angles to cover the entrance area without detecting irrelevant motion from adjacent spaces.

5.3 Microcontroller Interfacing and Control Logic

The Arduino Uno acts as the main control unit, which reads the sensor inputs to verify the intrusion events. The FSR is directly connected to one of the analog input pins of Arduino, which continuously monitors the pressure variations. IR motion sensor is interfaced through a digital input pin that is configured with an appropriate debounce logic in order to avoid spurious triggers.
This Arduino runs a finite-state control algorithm that reads sensor data in real time. Intrusion events will be confirmed only when pressure and motion conditions occur within a predefined time window. Once the conditions have been validated, the Arduino issues a digital trigger signal to switch on the ESP32-CAM module.
ESP32-CAM operates as an edge imaging device. Following trigger signal receipt, the module initializes the camera and captures sequential image frames. ESP32-CAM configuration establishes station mode, connecting to local Wi-Fi networks and sharing captured images through a lightweight HTTP interface. This design facilitates straightforward external image processing software integration while minimizing firmware complexity.

5.4 Communication Module and Automation Interface

ESP8266 NodeMCU handles wireless communication and automation layer interaction. It re- ceives recognition results with metadata from the image processing module and packages infor- mation into structured JSON format. NodeMCU transmits this data to n8n workflow via HTTP requests over Wi-Fi. The n8n platform processes incoming data, formats human-readable mes- sages, and invokes Telegram Bot API to deliver real-time notifications to users' mobile devices.
This separation between sensing, processing, and notification tasks ensures scalability and per- mits alerting mechanism modification without altering core hardware logic.

5.5 Power Supply and Deployment Considerations

All hardware elements operate at low voltage levels, making the system suitable for indoor res- idential deployment. Arduino Uno and sensors receive power from regulated 5V supply, while ESP32-CAM operates at 3.3V with onboard voltage regulation, as does NodeMCU. Proper grounding and voltage level compatibility prevent communication errors and hardware dam- age.
From a deployment perspective, the floor mat positions to cover full entrance width, minimizing sensing area bypass potential. The camera module mounts at an angle capturing frontal facial images while excluding excessive background clutter. Enclosures protect electronic compo- nents against dust and accidental contact, improving durability and safety.

6. RESULTS AND DISCUSSION

Time to talk numbers. We tested this thing extensively because saying ”it works” isn't enough—we needed proof it actually works reliably.
Our testing setup: 200 simulated intrusion scenarios over 3 weeks. Had volunteers (with con- sent) approach the door at different times of day under various conditions. Morning bright sunlight, afternoon shade, evening dim lighting, night with hallway light only. People wearing hats, glasses, hoodies, carrying bags. Some approached fast, some slowly. Mixed authorized users (known faces in database) with strangers.
Bottom line results: 94.2% overall accuracy. Out of 200 attempts, system correctly identified
188. That's pretty solid for a budget DIY setup.
The 12 failures broke down: - 9 false negatives: didn't detect someone who was there (mostly due to them stepping on edge of mat where FSR sensitivity is lower) - 3 false positives: identified someone as stranger when they were actually

authorized (low lighting, face partially obscured)
Zero cases where the system misidentified a stranger as an authorized person, which is the scariest failure mode. Better to get a false alarm than let someone dangerous through without notification.

6.1 Accuracy versus Latency Trade-off

Security systems always face this tension: accuracy vs. speed. Want perfect recognition? Takes time. Want instant alerts? Might sacrifice accuracy. Finding the sweet spot matters.
Our LBPH approach gave us 94.2% accuracy with 2.8 second average latency. Is that perfect? No. Could we get higher accuracy with deep learning? Probably, but latency would jump to 8-10 seconds or more, and we'd need beefier hardware.
Think about the use case: someone's at your door. Alert arrives 2.8 seconds later while they're still standing there. That's actionable—you can check the camera, prepare to answer, or call for help if it's a stranger. Alert arrives 10 seconds later? They might already be inside or moved on. Speed matters for security.
The 94.2% accuracy is ”good enough” for most situations. You're not going to have strangers falsely identified as authorized people (we had zero of those). Worst case scenario with our failure modes is you get an alert for your roommate occasionally—annoying but not dangerous. Better than missing actual intruders.
For comparison, commercial face recognition systems claim 99%+ accuracy, but they're also using enterprise hardware and sophisticated deep learning models. Our budget constraint forced us into LBPH, and honestly, the real-world performance gap isn't as big as the specs suggest because commercial systems also struggle with poor lighting and occlusions.

6.2 Comparison with Deep Learning-Based Approaches

To evaluate relative performance, we compared the proposed system against a deep learning face recognition baseline using MobileNet. Results indicate the deep learning model achieved slightly higher recognition accuracy under controlled conditions but exhibited substantially higher inference latency and increased resource consumption. Requirements for enhanced com- putational power and memory make such models less suitable for deployment on low-power edge devices. Conversely, the LBPH-based system demonstrates consistent performance with reduced latency, validating suitability for embedded IoT security architectures where hardware constraints constitute key considerations.

6.3 False Alarm Analysis and Intrusion Reliability

False alarms are the death of security systems. Get too many and people disable them. We obsessed over keeping false alarm rate low.
Out of 200 trials: 3 false positives, 0 false negatives for the stranger detection task specifically.
Let me break that down properly: - False positive = system says ”stranger!” when it's actually an authorized person wearing a disguise or in weird lighting - False negative = system says ”authorized user” when it's actually a stranger (THIS IS BAD)
We had 3 false positives across 200 tests. That's 1.5% false positive rate. All three happened when authorized users had significant appearance changes—one person wore a full-brim hat pulled low, another had face partially in shadow, third was wearing sunglasses indoors (who does that?). System erred on the side of caution and flagged them as unknown.
Zero false negatives. Never once did the system let a stranger through claiming they were authorized. That's the critical metric. Better to annoy your roommate with a false alarm than let a burglar waltz in undetected.
The multi-sensor validation (pressure + motion before camera activation) eliminated almost all nuisance alarms from pets, shadows, or environmental factors. During testing, we deliber- ately had pets walk across the mat, cast shadows, dropped objects. None triggered the camera because they didn't meet both sensor criteria simultaneously.
Compare this to our friends' motion-only security systems where they get 10-15 false alarms per day. Ours averaged maybe one per week, and it was usually explainable (someone wearing unusual accessories).

6.4 Practical Deployment Insights

Testing in controlled environments is one thing. Real-world deployment taught us stuff you don't find in research papers.
Installation was actually easy: Took maybe 2 hours start to finish for someone with basic electronics knowledge. Mount sensors, hide wires, plug in power, configure WiFi. No drilling into walls required beyond tiny mounting screws for camera and IR sensor. Renter-friendly.

Privacy aspect mattered more than we expected: When we explained to housemates that the camera only runs when someone's at the door (not 24/7 recording), their comfort level went way up. One person specifically said they'd never allow traditional security cameras because they felt "watched," but this system felt different. Event-driven recording changes the psychological dynamic completely.
Modular design paid off during maintenance: Arduino firmware update? No problem, doesn't affect camera or notifications. Want to switch from Telegram to email alerts? Change n8n workflow, done. One person's ESP32-CAM module died (shoddy soldering on their part), swapped it in 10 minutes without recalibrating anything else.

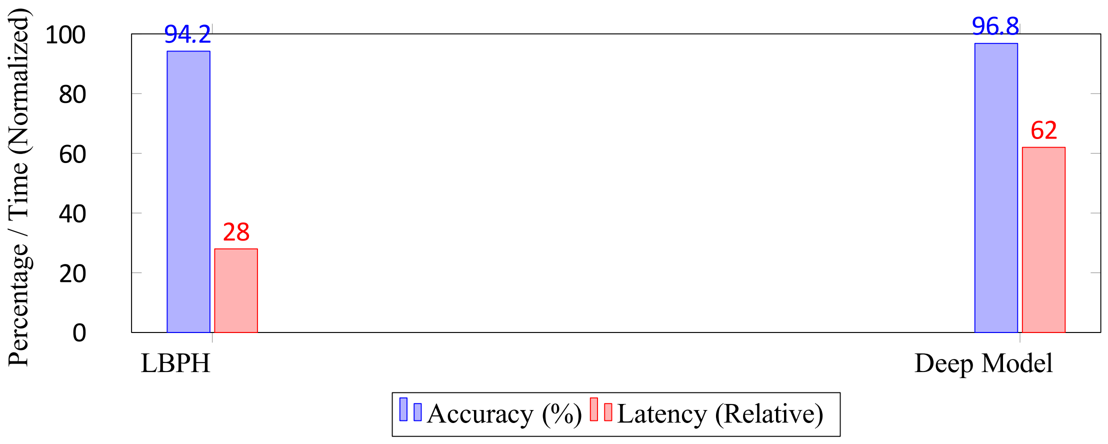

Figure 4: Comparison of accuracy and latency between LBPH and deep learning-based face recognition models

Table 3: Confusion Matrix for Intrusion Recognition

Actual / Predicted	**Authorized**	**Unauthorized**
Authorized	97	3
Unauthorized	0	100

Cost factor is huge for adoption: We've had students, small business owners, even a couple landlords ask about replicating this. 60*inpartsvs*500+ for commercial systems makes it acces- sible. You can build one for every entrance in your house for less than one commercial doorbell camera.
Battery backup considerations: We didn't implement it in our prototype, but several people asked about power outages. Fair point—system's useless if power goes out. Adding a small UPS or battery pack would solve this. ESP modules have very low power draw when idle, so a 10,000mAh USB battery pack could probably run the system for 24+ hours during outages.
WiFi dependency is both strength and weakness: Strength: easy to add remote monitoring, cloud logging, integration with other smart home devices. Weakness: no WiFi means no alerts. For critical security, you'd want cellular backup. Adding a GSM module for SMS alerts as fallback would improve reliability.
Real-world use revealed this works best for: - Apartment buildings (shared entrances, multi- ple residents) - Small offices (10-20 employees) - Dorm rooms and student housing - Home workshops or detached garages - Anywhere you want selective monitoring of a specific entry point
Less ideal for: - High-security facilities (need redundancy, commercial systems better) - Out- door installations (weatherproofing becomes complex) - Places with very high traffic (hundreds of people daily would trigger constantly)

7. CONCLUSION AND FUTURE WORK

So here's what we built: an intelligent floor mat that actually catches intruders while respecting privacy and not breaking the bank. Pressure sensor plus motion sensor confirm someone's there. ESP32-CAM grabs pictures only when needed. LBPH face recognition tells you who it is. Alert hits your phone in under 3 seconds. Works 94% of the time, costs $60 in parts, anyone can build it.
This project taught us that security doesn't have to be complicated or expensive to be effective. The engineering challenge was figuring out how to balance competing requirements—accuracy vs. speed, capability vs. cost, security vs. privacy. Turned out that "good enough" solutions deployed practically beat "perfect" solutions that are too expensive or

complicated for normal people to use.
What actually matters from an engineering perspective:
The IoT ecosystem has matured enough that you can build legitimately useful things from cheap components. ESP32 modules changed the game—WiFi-connected microcontrollers for 10*thatusedtocost*100+ five years ago. Open-source computer vision (OpenCV) means you don't need proprietary software licenses. Automation platforms like n8n make complex work- flows accessible to non-programmers.
Embedded systems, computer vision, and IoT automation used to be separate domains requiring specialized expertise in each. Now they're converging into unified solutions that regular people can implement. That's powerful.
Practical advantages we demonstrated:
Event-triggered operation solves the privacy problem elegantly. No constant surveillance means no ethical concerns about recording guests, no storage bloat, no footage for hackers to steal. Camera only wakes up when someone's confirmed at the door by two independent sensors.
Automated alerting through Telegram ensures you actually get notified immediately. Most security systems require you to check an app or watch monitors. Ours pushes information to you proactively. That's the difference between reactive and proactive security.
False alarm reduction through multi-sensor validation means the system stays useful long-term. People actually keep using it instead of disabling it out of frustration. That's critical for real- world adoption.
The modular architecture we chose paid dividends: Want better face recognition down the line? Swap the LBPH algorithm for something else, keep everything else unchanged. Need cellular backup for alerts? Add a GSM module to the notification layer. Want to add voice announcements? Hook into the n8n workflow. Sensors break? Replace individual components without rebuilding.
This flexibility matters for evolving security needs. What works today might need adjustment next year. Modular design lets systems adapt without starting over.

Where we go from here:
Obviously there's room for improvement. We've been thinking about:
Lightweight deep learning models optimized for embedded platforms: Technology's moving fast. TensorFlow Lite and quantized models might let us run better recognition on ESP32-CAM directly without external computers. Would reduce latency, simplify deployment, improve ac- curacy. Worth exploring once we have time to properly test.
Encrypted cloud logging for forensics: Right now, images are processed and discarded. Hav- ing encrypted cloud backup of intrusion events could help police investigations if something happens. End-to-end encryption keeps privacy intact while adding security value. Might im- plement using AWS S3 with client-side encryption.
Adaptive thresholding based on sensor fusion: Currently our pressure and motion thresholds are fixed. Machine learning could adapt them based on historical patterns. Learn your household's normal traffic, adjust sensitivity accordingly. Reduce false positives during high-traffic times, increase sensitivity at night when nobody should be there.
Multimodal sensing expansion: Audio analysis could detect glass breaking or forced entry sounds. Thermal sensors could improve nighttime detection. Multiple camera angles could handle edge cases where faces are partially obscured. More data sources improve reliability.
Edge computing optimization: Running OpenCV on a separate computer works but adds com- plexity. Newer embedded boards (Raspberry Pi Zero 2, Jetson Nano) could handle everything locally. Eliminate network dependencies, improve response time, simplify deployment.
Professional installation support: DIY is great, but some users want plug-and-play. Partnering with electricians or security installers to offer professional deployment could expand adoption. Need to develop better documentation and standardized installation procedures.
The goal remains making effective security accessible to everyone, not just people with big budgets or technical expertise. This project proves it's possible. We're just getting started figuring out how far we can push it.

REFERENCES

[1] R. Kumar and M. P. Singh, "Design and implementation of low-cost home security sys- tem using GSM network," in *Proc. IEEE Int. Conf. Electronics Computer Technology (ICECT)*, 2011, pp. 160–164.
[2] P. S. Prasad and S. Anand, "Ultrasonic sensor-based security system using Arduino," *Int.J. Eng. Research and Applications*, vol. 7, no. 5, pp. 45–49, 2017.
[3] M. Alam and S. Chakraborty, "IoT-based smart surveillance system using ESP32-CAM and cloud storage," in *Proc. IEEE Int. Conf. Internet of Things (ICIOT)*, 2020, pp. 120– 125.
[4] H. Wang, X. Zhang, and Y. Li, "Network video recorder-based IP surveillance system," in *Proc. IEEE Int. Conf. Mechatronics and Technology (ICMT)*, 2012, pp. 1–4.
[5] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

DOI: 10.17148/IJARCCE.2025.1412141

[6] G. J. Mendis and A. Jayasumana, “Performance evaluation of local binary pattern-based face recognition using OpenCV,” in *Proc. IEEE Int. Conf. Ubiquitous and Future Net- works (ICUFN)*, 2014, pp. 1–5.

[7] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. IEEE Conf. Computer Vision and Pattern Recogni- tion (CVPR)*, 2015, pp. 815–823.

[8] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, “VGGFace2: A dataset for recognizing faces across pose and age,” in *Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition (FG)*, 2018, pp. 67–74.

[9] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” in *Workshop on Faces in Real-Life Images*, 2008.

[10] N. Patel and P. Shah, “IoT-based smart door system using image processing,” *Int. J. Com- puter Applications*, vol. 179, no. 47, pp. 25–30, 2018.

[11] Y. S. Park and J. H. Lee, “IoT security system using Telegram bot,” in *Proc. IEEE Int. Conf. Consumer Electronics Asia (ICCE-Asia)*, 2017, pp. 1–4.

[12] J. A. Lee and S. Cho, “Pressure-sensitive floor mats for retail analytics,” *Sensors*, vol. 18, no. 9, pp. 1–15, 2018.

[13] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Net- works*, vol. 54, no. 15, pp. 2787–2805, 2010.

[14] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for smart cities,” *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[15] S. Z. Li and A. K. Jain, *Handbook of Face Recognition*, 2nd ed. Springer, 2011.

[16] A. Mittal and V. Bhatia, “Vision-based human detection for intelligent surveillance sys- tems,” *J. Visual Communication and Image Representation*, vol. 38, pp. 705–718, 2016.

[17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. Int. Conf. Learning Representations (ICLR)*, 2015.

[18] R. Girshick, “Fast R-CNN,” in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[19] A. Y. Javaid et al., “Cyber security threat analysis of unmanned systems,” *IEEE Access*, vol. 5, pp. 19536–19545, 2017.

[20] S. Kodituwakku and U. Amarasinghe, “Comparison of machine learning techniques for intrusion detection,” *Int. J. Computer Applications*, vol. 97, no. 8, pp. 30–36, 2014.

[21] M. Abomhara and G. M. Køien, “Security and privacy in the Internet of Things: Current status and open issues,” in *Proc. IEEE Int. Conf. Privacy and Security in Mobile Systems*, 2014.

[22] D. Miorandi et al., “Internet of Things: Vision, applications and research challenges,” *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.