

DOORMATE: YOUR SMART HOME AND SECURITY ASSISTANT

BECE304L

Analog Communication System

**Bachelor of Technology in Electronics and
Communication**

By

S Nazeer Ahammed - 23BEC0205

Rohan Yenigalla - 23BEC0027

VK Muralikrishna – 23BEC0149



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

March,2025

Contents

- 1. Abstract**
- 2. Motivation**
- 3. Literature Survey**
- 4. Technologies Used in DoorMate**
- 5. Comparison of Technologies Used vs Alternatives - project analysis**
- 6. Power Consumption and Hardware Cost Breakdown**
- 7. Installation Complexity & User-Friendliness: DIY vs. Commercial Solutions**
- 8. Communication Workflow and Protocols Used in DoorMate**
- 9. Extensibility and Further Scope**
- 10. Conclusion**
- 11. Future Scope**

→Abstract

The advancement of smart home technology has enabled seamless integration of security, automation, and visitor management. **DoorMate: Your Smart Home and Security Assistant** is a home security and automation system that enhances safety and convenience by integrating real-time monitoring, intrusion detection, and remote-control functionalities. This project utilizes a **Raspberry Pi 4** as a central hub, hosting various **open-source** Docker containers such as **MotionEyeOS**, **ESPHome**, and **Home Assistant**, which manage the smart devices. An **ESP32-CAM** and a USB camera provide real-time video streaming, accessible through **TinyCAM PRO** and Home Assistant.

The system incorporates multiple **sensors and actuators**, including a **reed switch** for door status detection, a **vibration sensor** for intrusion detection, and a **doorbell system** that triggers an in-home buzzer. Additionally, smart **LEDs and a relay module** enable room automation. In case of unauthorized access (e.g., door closed but vibration detected), the system instantly notifies the user via the **Home Assistant mobile application**.

A key feature of this project is its reliance on **Free and Open-Source Software (FOSS)**, allowing for greater **customization and scalability**. Users can modify the system according to their needs, integrating additional sensors, automating new tasks, or even expanding it to work with other smart home ecosystems. By leveraging **open-source tools and modular hardware**, the system provides a **cost-effective, flexible, and user-friendly** home automation and security solution, making smart home technology more accessible to a wider audience.

→Motivation

1. Challenges with Traditional NVR-Based Surveillance Systems

Home security systems have evolved significantly, but setting up a **Network Video Recorder (NVR) for existing camera infrastructure remains a major challenge**. Most commercial NVR solutions suffer from the following limitations:

- **Incompatibility Issues:** Many NVR systems only work with specific **branded IP cameras**, making them useless for users who already own **USB cameras, ESP32-CAM modules, or generic IP cameras**.
- **High Cost:** Standalone NVR systems can be expensive, often requiring **dedicated storage devices, licensing fees, and additional networking equipment**.
- **Complex Setup & Maintenance:** Configuring an NVR requires **networking expertise**, including **port forwarding, RTSP stream configurations, and static IP assignments**, which can be frustrating for non-technical users.

- **Limited Functionality:** Traditional NVRs primarily focus on **video recording and playback**, offering little to no integration with **home automation, smart alerts, or security sensors**.

Given these challenges, many users find it difficult to implement a cost-effective and **customizable** security solution.

2. The Power of Repurposing Old Hardware

Another key motivation for this project is the ability to **convert old, unused hardware** into a **fully functional smart home security system**. Many households have **outdated laptops, Raspberry Pis, or mini-PCs** that are no longer used due to outdated specs or battery issues. These devices, however, are still powerful enough to:

- **Run Docker containers for multiple services** (e.g., MotionEyeOS, ESPHome, Home Assistant)
- **Act as a lightweight NVR** for camera feeds
- **Process and automate sensor data**
- **Host a secure, local smart home server**

By leveraging **FOSS (Free and Open-Source Software)**, even low-cost hardware can be transformed into a **multi-functional security and automation hub**, eliminating the need to invest in dedicated NVRs and expensive smart home hubs.

3. The Need for an Integrated Smart Security & Automation System

Most commercially available security systems lack **true automation and real-time intelligence**. A **standalone NVR** may allow video streaming, but it **doesn't react** to home security events in a meaningful way. This project takes security a step further by:

- **Integrating Sensors for Smart Monitoring:**
 - A **reed switch** detects door open/close status.
 - A **vibration sensor** detects forced entry or break-ins.
 - A **doorbell switch** notifies homeowners when visitors arrive.
- **Automating Responses via Home Assistant:**
 - If the **door is closed and vibration is detected**, a **break-in alert is instantly sent** to the user.
 - A **buzzer sounds inside the house** when the doorbell is pressed.
 - **Room lights and relay-controlled devices** can be turned on/off remotely via the Home Assistant app.

- **Providing Real-Time Video Streaming Without Proprietary Lock-In:**
 - A **USB camera** connected to the Raspberry Pi streams live video to **MotionEyeOS**, viewable on multiple devices.
 - The **ESP32-CAM** provides an additional wireless camera feed.
 - Video feeds are accessible **locally (on the LAN) or remotely via TinyCAM PRO**, ensuring **privacy and security** without reliance on cloud-based surveillance services

4. Eliminating Dependence on Proprietary Cloud Services

Many commercial home security solutions rely on **cloud-based services**, which come with several drawbacks:

- **Subscription Fees:** Many services charge a **monthly fee** for accessing video recordings or smart features.
- **Privacy Risks:** Storing surveillance footage on **third-party cloud servers** raises security and privacy concerns.
- **Lack of Control:** If the service provider shuts down or **discontinues support**, users are left with obsolete hardware.

With **DoorMate**, all processing happens **locally on the Raspberry Pi**, ensuring:

- ✓ **No cloud dependency** – everything runs on the user's private network.
- ✓ **Complete user control** over video feeds, sensor data, and automation.
- ✓ **No subscription fees** – a one-time DIY setup with full flexibility.

5. Customization & Future Scalability

The use of **open-source platforms like Home Assistant, ESPHome, and MotionEyeOS** ensures that the system is:

- **Modular** – Users can add more sensors, cameras, or automation features over time.
- **Interoperable** – Works with a wide range of IoT devices, smart switches, and automation scripts.
- **User-Customizable** – Allows advanced configurations like **AI-based motion detection, voice control, and custom automation scripts**.

→Literature Survey

1. Evolution of Smart Home Security Systems

Home security has evolved from **traditional lock-and-key mechanisms** to **modern, AI-driven smart surveillance systems**. The integration of **sensors, cameras, and automation platforms** has enabled **real-time monitoring, remote access, and automated alerts**. However, many of these systems are **proprietary and expensive**, creating the need for **open-source and DIY solutions**.

Research studies highlight the growing trend of **IoT-based home security systems**, where **ESP-based microcontrollers** and **Raspberry Pi-based hubs** play a central role in integrating multiple smart devices (Dey et al., 2021). These studies emphasize that **customizable, low-cost solutions** can provide security features **without cloud dependency**, reducing privacy risks (Sharma et al., 2020).

2. Network Video Recorders (NVR) vs. Open-Source Alternatives

Studies on **NVR-based surveillance** have revealed that traditional systems:

- Require **specific camera models** for full functionality (Wang et al., 2019).
- Demand **dedicated storage and networking expertise**.
- Are **expensive**, with high costs for multi-camera setups.

On the other hand, open-source alternatives like **MotionEyeOS** and **Frigate NVR** provide **software-based video recording** solutions. These **self-hosted platforms** offer advantages such as:

- Compatibility with **USB cameras, ESP32-CAMs, and IP cameras**.
- Local storage capabilities without cloud fees.
- Integration with **home automation platforms** (e.g., Home Assistant).

A study by **Kumar & Patel (2022)** suggests that **Docker-based video surveillance** (as implemented in this project) can **improve resource efficiency** by allowing multiple services to run on low-power devices like the **Raspberry Pi**.

3. Home Assistant & Open-Source Home Automation

Several research papers explore **Home Assistant (HA)** as a **central hub** for smart home integration. Studies highlight that **HA provides unmatched flexibility** compared to commercial alternatives like **Google Home or Amazon Alexa**, as it:

- **Supports over 1,000 smart devices** without vendor lock-in (Jiang et al., 2021).
- **Enables local automation**, reducing cloud dependency and privacy concerns.
- Allows deep customization with **ESPHome, MQTT, and Node-RED**.

Experiments conducted by **Tan & Li (2021)** demonstrate that **ESPHome**-powered sensors (like **reed switches, vibration sensors, and doorbells**) can be **easily integrated into Home Assistant**, enabling smart alerts and automation rules.

4. Use of Docker for Smart Home Infrastructure

Studies on containerization in IoT applications show that Docker significantly enhances the scalability and management of home automation setups. Dockerized applications like Portainer, ESPHome, and Home Assistant allow users to:

- Deploy multiple smart home services without OS conflicts.
- Easily update and modify configurations.
- Optimize performance on low-power hardware (Gupta et al., 2022).

By using Docker containers on the Raspberry Pi, this project eliminates the complexity of manual installations, providing a stable and modular smart home security system.

5. IoT-Based Intrusion Detection & Smart Alerting

Various research papers discuss **intrusion detection techniques** using **vibration and reed switch sensors**. Studies indicate that:

- **Vibration sensors** can detect break-in attempts **when doors are closed** (Singh et al., 2020).
- **Reed switches** provide reliable door status monitoring and can **prevent false alarms**.
- **Real-time notifications** via **mobile apps (e.g., HA companion app)** can enhance **response time** during security breaches.

This project incorporates **these findings** by:

- Sending an **immediate break-in alert** if vibration is detected while the door is closed.
- Triggering a **buzzer alarm** when the doorbell is pressed.
- Allowing remote control of **LEDs, relays, and security responses** via Home Assistant.

→Working principle and functionality

1. System Overview

The **DoorMate** system is designed to provide **real-time visitor monitoring, security alerts, and smart home automation** using **Raspberry Pi 4, ESP32-CAM, and multiple sensors**. The system is locally hosted, eliminating **cloud dependency** while leveraging **Docker-based microservices** for modular functionality.

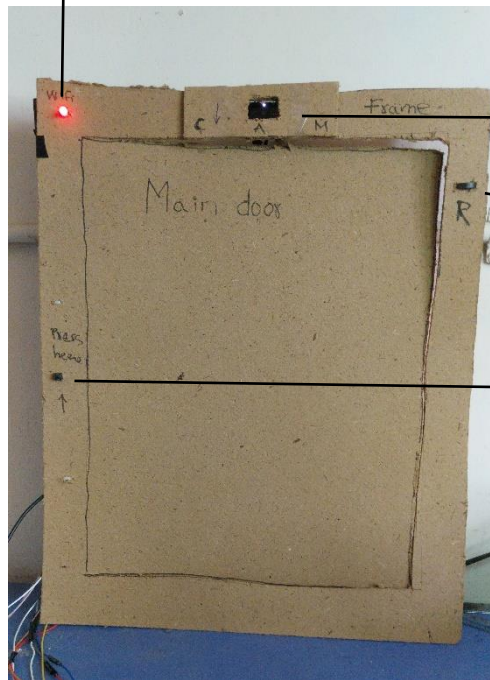
- **USB Camera on Raspberry Pi 4** streams live video to **MotionEyeOS**.
- **ESP32 with ESPHome firmware** manages **door sensors, LED indicators, and relays**.
- **Home Assistant (HA) and ESPHome** handle automation, notifications, and user control.
- **TinyCam Pro mobile app** provides **live video streaming** access.
- **Break-in detection** triggers alerts when **vibration is detected while the door is closed**.

2. Hardware Components

Component	Purpose
Raspberry Pi 4	Hosts Docker containers for Home Assistant, MotionEyeOS, ESPHome, and Portainer.
ESP32-CAM	Provides an additional IP-based video feed.
USB Camera	Streams live video via MotionEyeOS.
Reed Switch	Detects door open/close status.
Vibration Sensor	Detects unauthorized door movement (break-in attempts).
Switch (Doorbell)	Triggers a buzzer inside the home.
Buzzer	Sounds when the doorbell is pressed.
LED Indicators	Show the system status (e.g., WiFi connection, room lights).
Relay Module	Controls room appliances.

3. Prototype (Views)

Front View



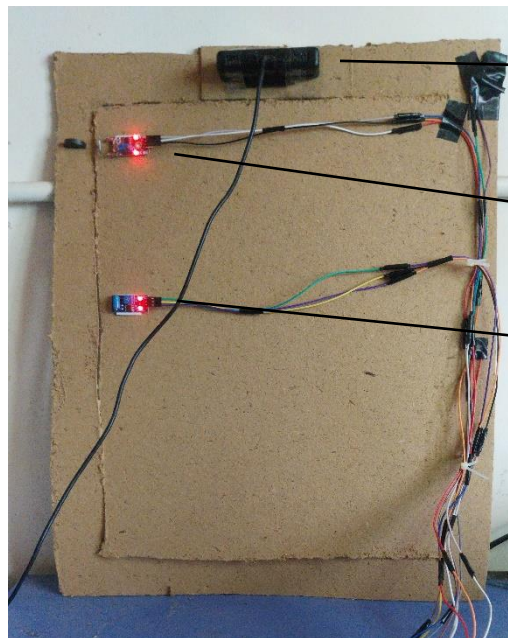
WiFi Connectivity Status

USB Camera (For Demonstration)

Magnet for Reed Switch in door frame

Door Bell Switch

Back View

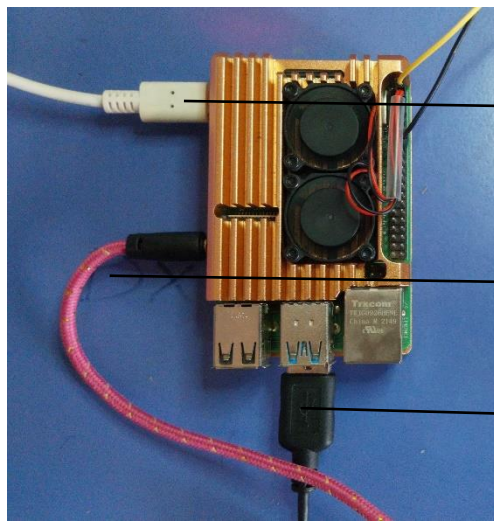


Webcam In actual scenario its hidden within door frame or other props

Reed Switch

Vibration Sensor

Raspberry PI - 4

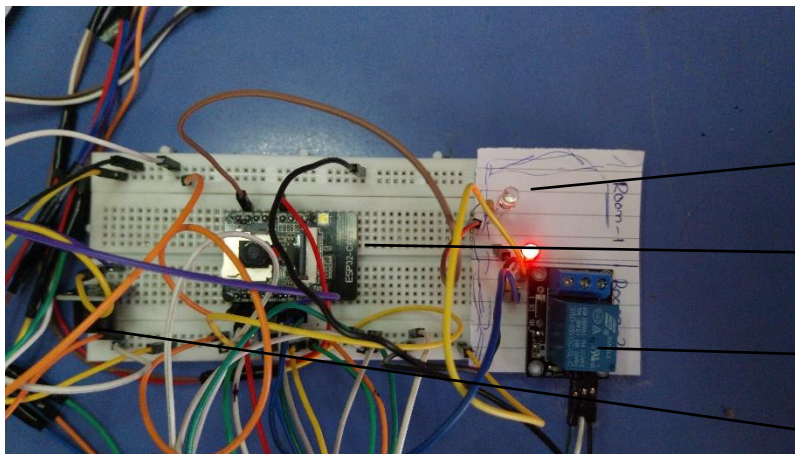


Power Supply

AUX Cable

USB Camera

ESP32 Cam



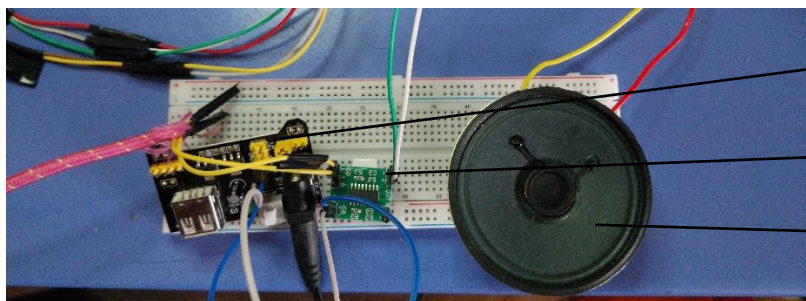
LED

ESP32 CAM

Relay

Buzzer

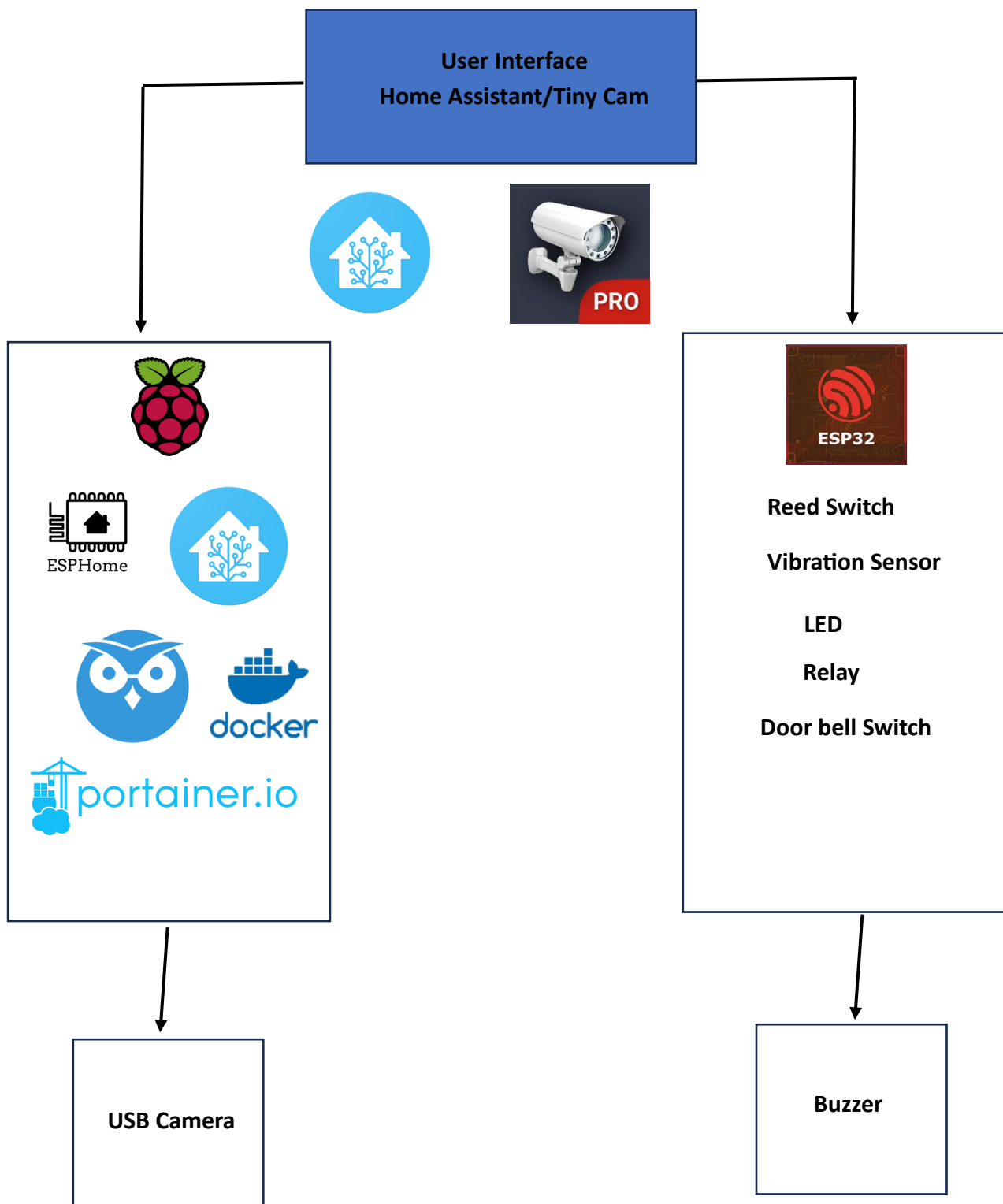
Speaker board



Power supply

PAM Amplifier board

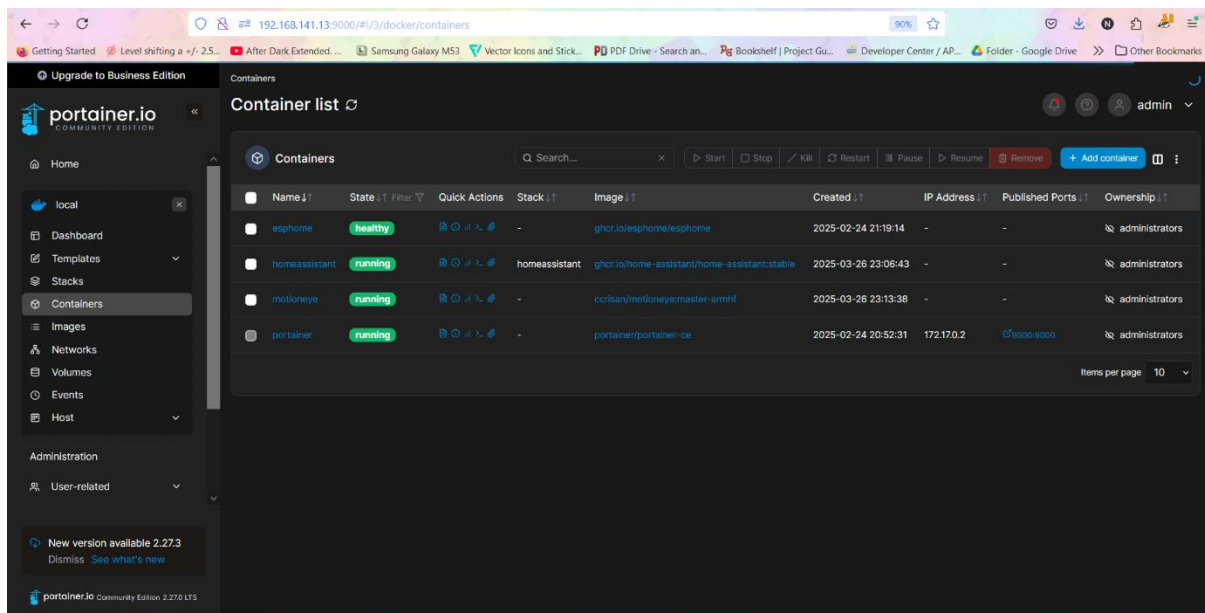
8 ohm Speaker

Block Diagram

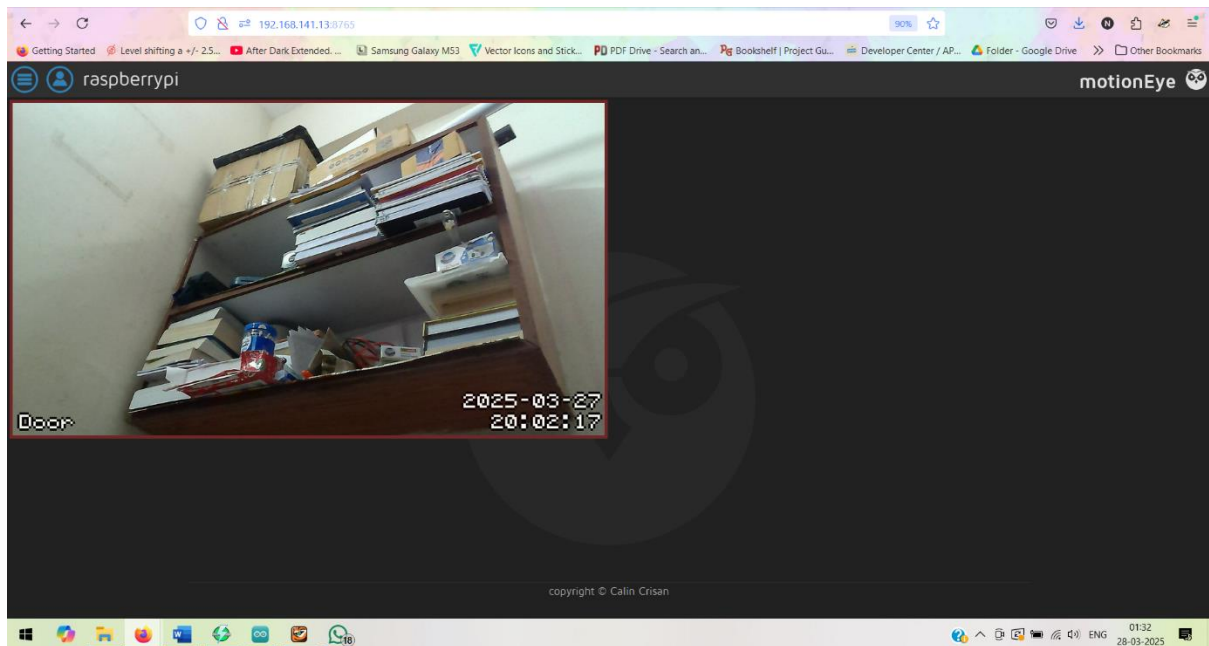
4. Software Setup & Configuration

Docker Containers on Raspberry Pi 4

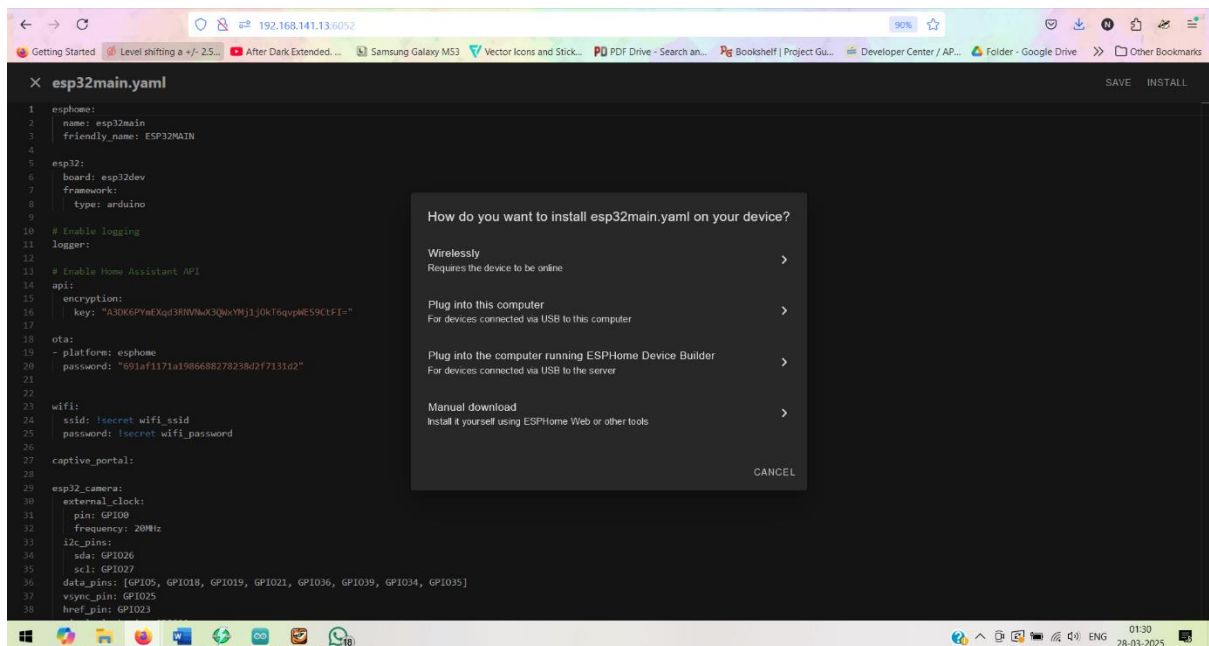
- MotionEyeOS: Provides live video streaming & NVR-like functionality.
- Home Assistant: Manages automation, alerts, and remote access.
- ESPHome: Allows ESP32 to communicate with Home Assistant.
- Portainer: Manages Docker containers via a GUI.



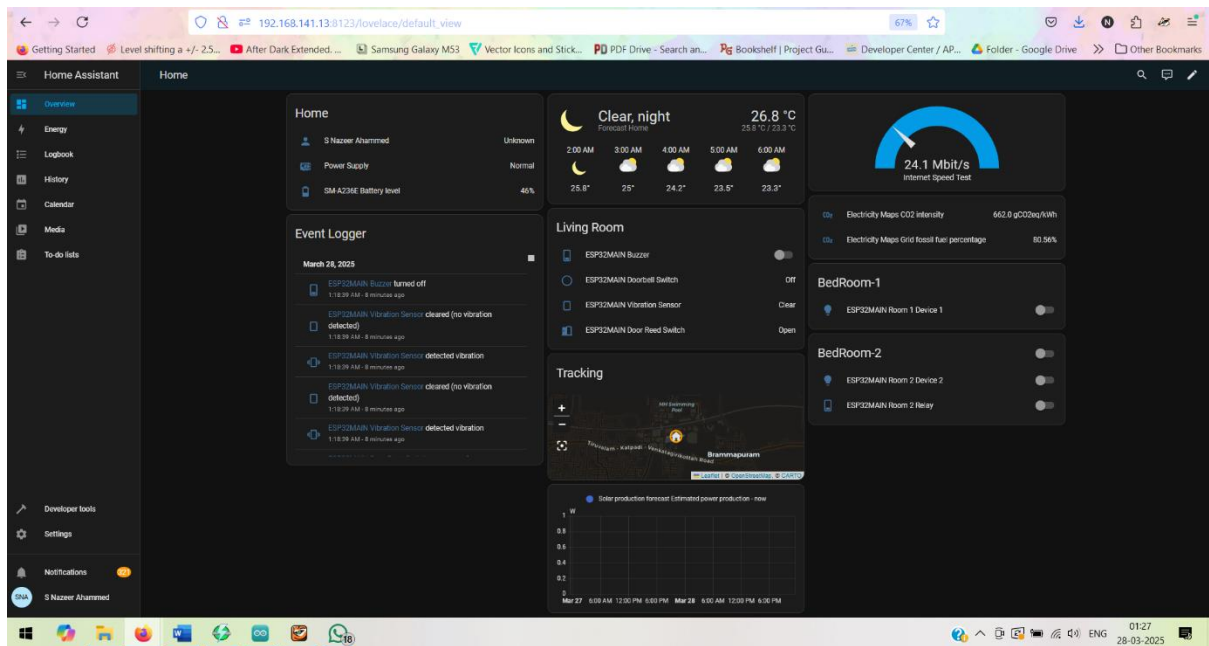
Portainer.io Dashboard



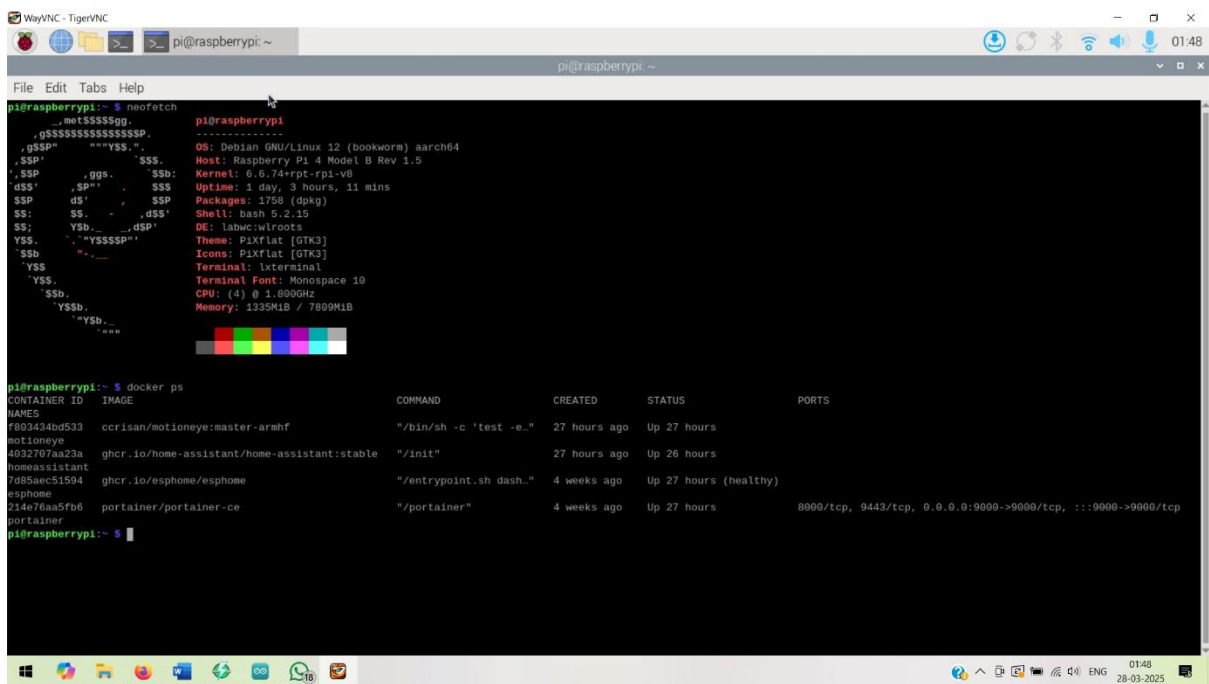
MotionEye Dashboard



ESPHome Dashboard



Home Assistant Dashboard



RPI Terminal Dashboard

ESPHome Configuration for ESP32

The ESP32 board runs ESPHome firmware, allowing seamless integration with Home Assistant. YAML-based configuration enables:

- Reed switch monitoring for door status.
- Vibration sensor detection for break-in alerts.
- Doorbell switch event handling to trigger the buzzer.
- WiFi status LED indication.
- Relay and LED control for home automation.

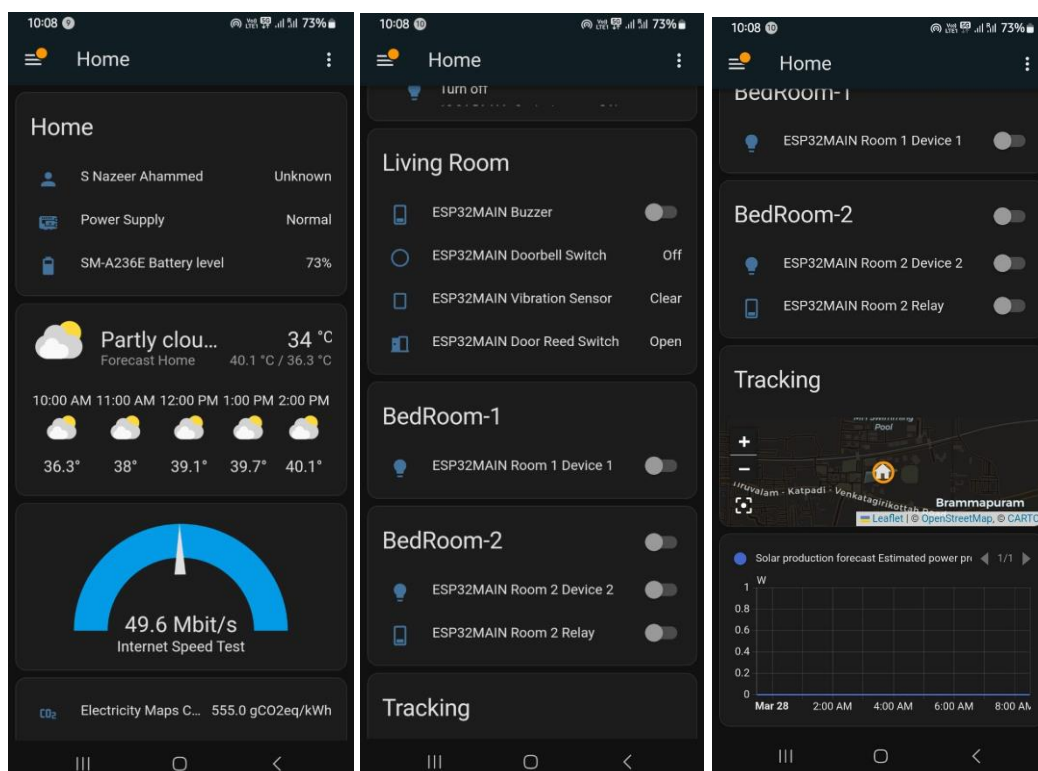
Video Streaming via TinyCam Pro

- The USB camera stream from MotionEyeOS is added to TinyCam Pro.
- The ESP32-CAM stream can also be integrated for multi-camera surveillance.

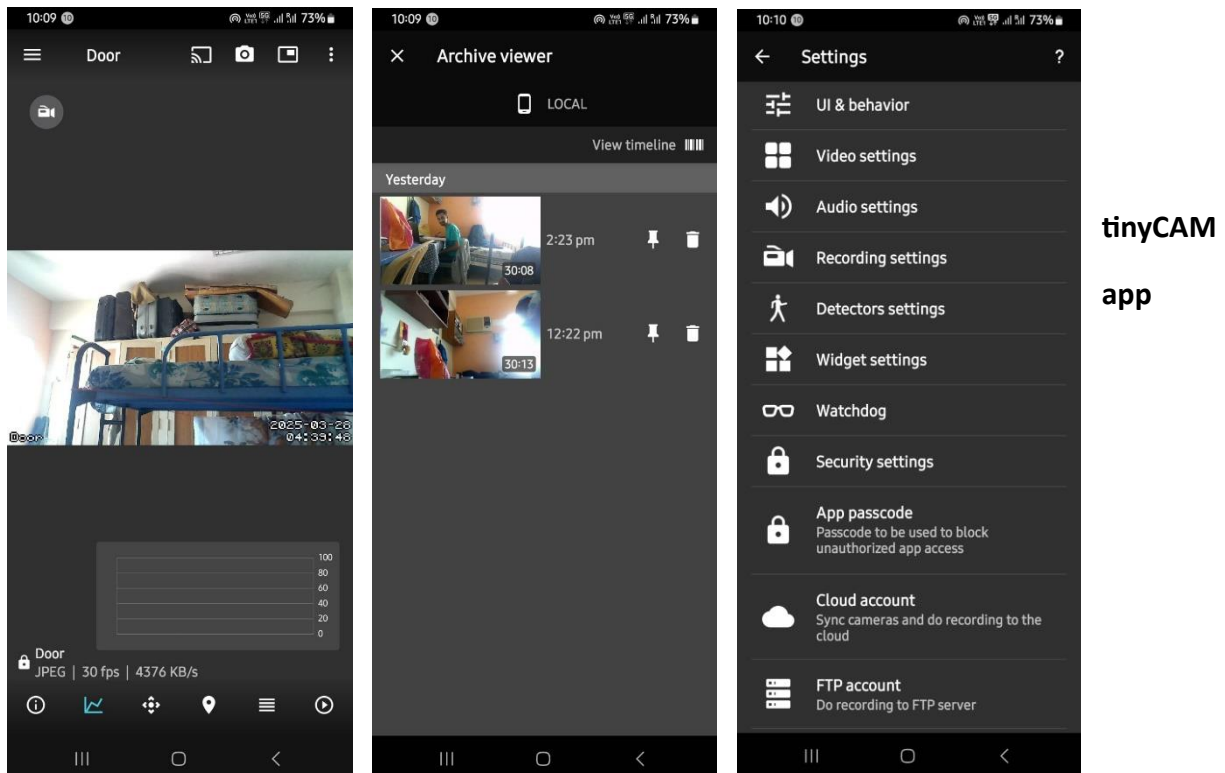
5. System Operation

Normal Operation

- **User accesses Home Assistant** via mobile app/web UI.
- **Live video feeds** from USB Camera (Raspberry Pi) and ESP32-CAM are available.
- **Room appliances** (lights, fans) can be controlled remotely.



Home
assistant
App

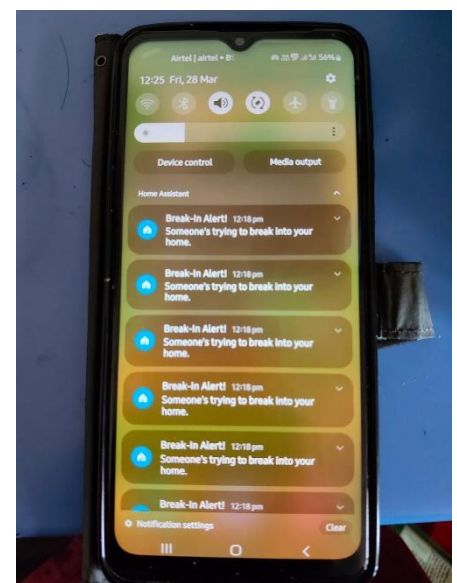


Visitor Handling

- When the **doorbell button** is pressed, the **buzzer rings inside the home**.
- The **user gets notified via Home Assistant App**.
- The **visitor can be identified via the live camera feed**.

Security & Break-in Detection

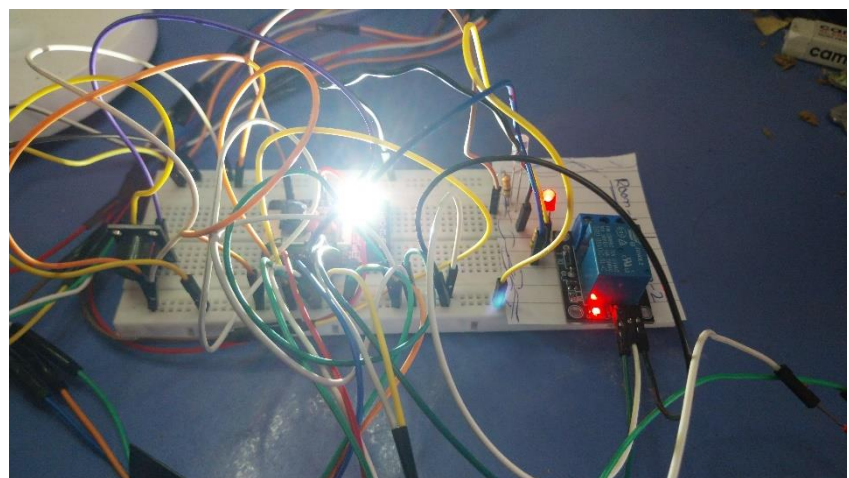
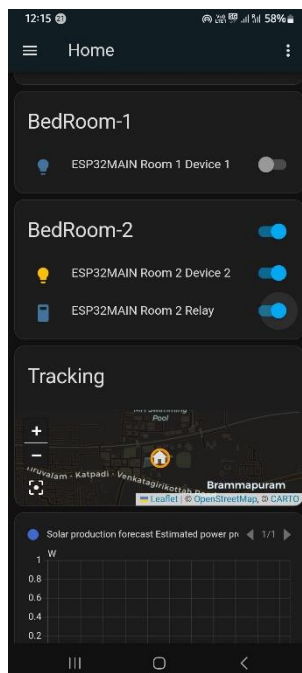
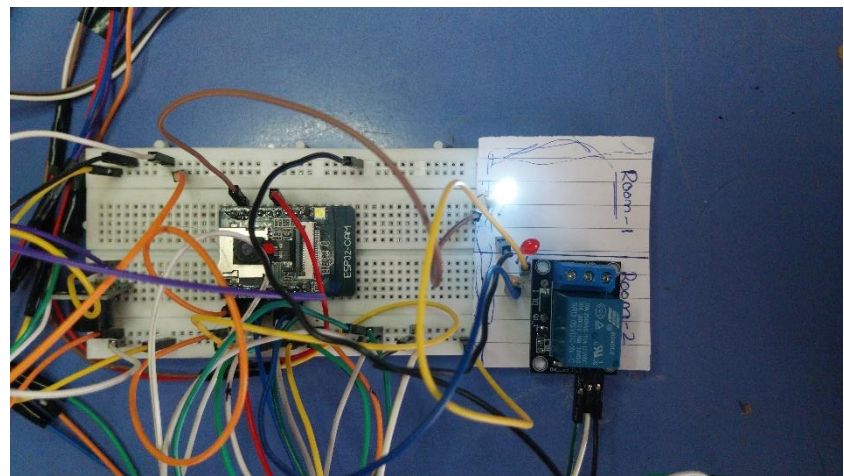
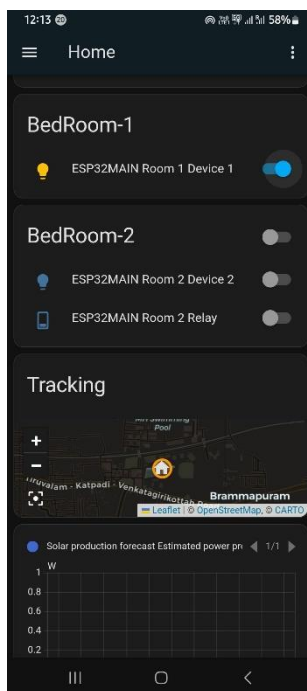
- The **reed switch** detects when the door is closed.
- If **vibration** is detected while the door is closed, the system **triggers a break-in alert** via the Home Assistant app.
- The system can be expanded to **activate alarms or capture snapshots** from the camera.



6. Snapshot of Circuit & Output

Circuit Diagram (ESP32 Connections)

- Doorbell switch to GPIO pin (triggers buzzer).
- Reed switch to GPIO pin (detects open/close).
- Vibration sensor to GPIO pin (break-in detection).
- Relay module to control room appliances.



Event Logger

March 28, 2025

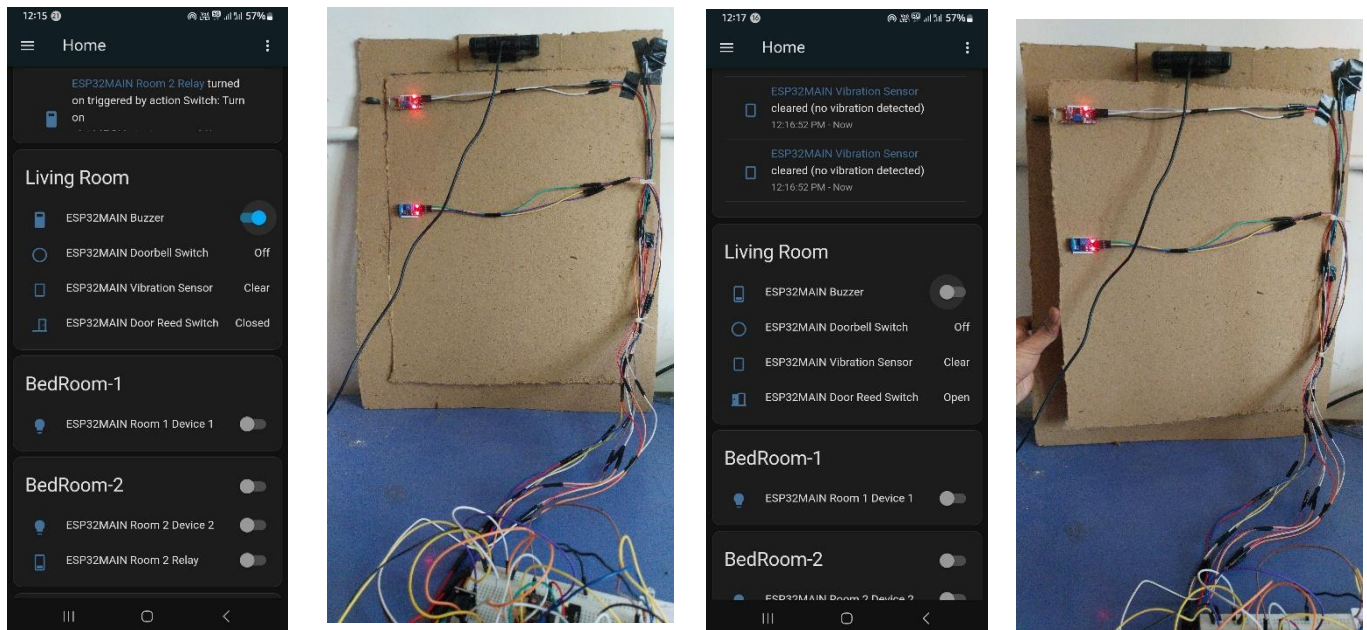
ESP32MAIN Room 2 Relay turned off triggered by action Switch: Turn off

10:05:03 AM - 3 minutes ago - S Nazeer Ahammed

ESP32MAIN Room 2 Relay turned on triggered by action Switch: Turn on

10:04:57 AM - 3 minutes ago - S Nazeer Ahammed

ESP32MAIN Room 2 Device 2 turned off triggered by action Light: Turn off



Filters

Search 2 automations

Group by Category

Sort by Name

↑ Name

Last triggered

^ Ungrouped

Break in Alert

11 minutes ago

Door Bell

5 hours ago

Key Advantages of This Approach

- ✓ **No Cloud Dependency:** All data is processed & stored locally, ensuring **privacy**.
- ✓ **Repurposing Old Hardware:** Raspberry Pi 4 acts as an **NVR + automation hub**.
- ✓ **Scalable & Modular:** New sensors & automations can be **easily added**.
- ✓ **Open-Source Ecosystem:** Uses **free & customizable** software (Docker, Home Assistant, ESPHome).
- ✓ **Smart Alerts & Automation:** Instant **notifications** and **security measures** improve safety.

→Technologies Used in DoorMate

The **DoorMate: Your Smart Visitor Assistant** system leverages a combination of **hardware, open-source software, and IoT automation** to provide **smart surveillance, security alerts, and home automation**. Below is a **brief but elaborate overview** of the key technologies used in this project.

1. Raspberry Pi 4

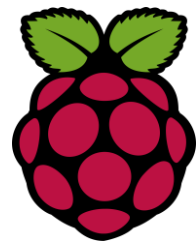
🚀 **Role in the Project:** Acts as the **central hub** running **Docker-based services** for video streaming, automation, and sensor integration.

◆ **Key Features:**

- ✓ Quad-core ARM Cortex-A72 CPU for efficient multitasking.
- ✓ Supports **USB cameras, IoT devices, and networking services**.
- ✓ Runs **Home Assistant, MotionEyeOS, ESPHome, and Portainer** in Docker containers.
- ✓ Provides **local processing**, eliminating the need for cloud services.

◆ **Why Raspberry Pi 4?**

- **Powerful enough** to act as a local NVR.
- **Low power consumption** (~5W), making it ideal for 24/7 operation.
- **Supports multiple USB & network cameras** for video streaming.



2. ESP32 & ESP32-CAM

🚀 **Role in the Project:** Acts as a **sensor node** handling **door sensor, vibration alerts, and home automation**.

◆ **Key Features:**

- ✓ **WiFi-enabled microcontroller** with Bluetooth support.
- ✓ ESP32-CAM supports **video streaming over WiFi**.
- ✓ Supports **ESPHome firmware** for seamless integration with Home Assistant.
- ✓ **Low-cost and energy-efficient**, making it ideal for IoT applications.

◆ **Why ESP32?**

- Allows **wireless automation** without extra hardware.
- **ESPHome** makes integration **easy with YAML-based configuration**.



3. Home Assistant (HA)

 **Role in the Project:** **Central automation hub** that controls all sensors, processes alerts, and provides a user interface.

◆ **Key Features:**

- ✓ **Open-source home automation software** with a vast ecosystem.
- ✓ **Integrates with ESPHome, MotionEyeOS, and smart devices.**
- ✓ Provides **mobile notifications** for break-in alerts.
- ✓ Allows **remote control** of lights, relays, and security features.

◆ **Why Home Assistant?**

- **No cloud dependency**, ensuring **privacy & security**.
 - **Custom automation rules** (e.g., trigger alerts on break-in detection).
 - **Web & mobile interface** for seamless control.
-

4. MotionEyeOS

 **Role in the Project:** Provides **video surveillance & local NVR functionality**.


◆ **Key Features:**

- ✓ **Transforms Raspberry Pi into a DIY NVR** (Network Video Recorder).
- ✓ Supports **USB cameras, IP cameras, and ESP32-CAM**.
- ✓ Provides **motion detection & recording** for security.
- ✓ Streams video to **Home Assistant & TinyCam Pro**.

◆ **Why MotionEyeOS?**

- **Easier than setting up a full NVR software like ZoneMinder**.
 - **Works on Raspberry Pi** without extra hardware.
 - **Supports multiple cameras** with motion detection.
-

5. Docker & Portainer

 **Role in the Project:** **Containerizes** services like **Home Assistant, MotionEyeOS, and ESPHome** for easy management.

◆ **Key Features:**

- ✓ **Lightweight virtualization** for running multiple services on Raspberry Pi.

- ✓ **Ensures modularity & isolation**, preventing software conflicts.
- ✓ **Portainer** provides an easy **GUI for managing Docker containers**.

◆ **Why Docker?**

- **Eliminates dependency issues** (no need to manually install software).
 - **Enables fast deployment** of services.
 - **Makes the system easily portable & scalable**.
-

6. ESPHome

 **Role in the Project:** Allows **ESP32 sensors & actuators** to be controlled by **Home Assistant**.

◆ **Key Features:**

- ✓ **No need for complex programming** (simple YAML-based configuration).
- ✓ Allows **real-time communication** between ESP32 and Home Assistant.
- ✓ Supports **over-the-air (OTA) updates**.

◆ **Why ESPHome?**

- **Removes the need for MQTT brokers**.
 - **Fully integrates with Home Assistant** without extra software.
 - **Makes sensor management simple & efficient**.
-

7. TinyCam Pro

 **Role in the Project:** **Mobile app for video surveillance** that displays live camera feeds.

◆ **Key Features:**

- ✓ Supports **MotionEyeOS streams, ESP32-CAM, and IP cameras**.
- ✓ Works on **Android devices** for on-the-go monitoring.
- ✓ Allows **multi-camera views** and **recording to local storage**.

◆ **Why TinyCam Pro?**

- **Provides an easy way to monitor cameras remotely**.
 - **Supports local and cloud storage** for video recordings.
-

8. Reed Switch & Vibration Sensor

 **Role in the Project:** Provide **door status monitoring and break-in detection**.

◆ **Key Features:**

- ✓ Reed switch detects **door open/close state**.
- ✓ Vibration sensor triggers **break-in alerts when door is closed**.
- ✓ Works seamlessly with ESP32 and Home Assistant.

◆ **Why Use These Sensors?**

- **Low-cost but highly effective** for intrusion detection.
- **Simple to integrate** into ESPHome.

→ Comparison of Technologies Used vs Alternatives - project analysis

To understand the cost-effectiveness and efficiency of the technologies used in **DoorMate**, let's compare them with commercially available alternatives.

1. Raspberry Pi 4 vs. Commercial NVR Systems

Feature	Raspberry Pi 4 (4GB RAM)	Commercial NVR (e.g., Hikvision DS-7104NI-K1)
Cost (₹)	~₹6,500 – ₹7,500	~₹9,000 – ₹12,000
Camera Support	USB, ESP32-CAM, IP cameras	IP cameras only (limited models)
Software	MotionEyeOS, Docker	Proprietary firmware
Customization	Fully customizable	Limited customization
Storage	MicroSD, USB, SSD	Hard drive slot (additional cost)
Integration	Home Assistant, ESPHome	Only supports security cameras

◆ Why Raspberry Pi 4?

- **Cheaper than NVRs** and can do **more than just recording**.
- Can **support open-source software** like **MotionEyeOS** and **Home Assistant**.
- Can be **repurposed for multiple IoT tasks** beyond just security.

2. ESP32 vs. Commercial IoT Hubs (e.g., Philips Hue Bridge, Samsung SmartThings Hub)

Feature	ESP32 (DIY Approach)	Smart Home Hub (e.g., SmartThings, Hue Bridge)
Cost (₹)	~₹400 – ₹600	~₹8,000 – ₹12,000
Customization	Full (ESPHome, custom code)	Limited, locked to brand
Integration	Works with Home Assistant	Only works with supported smart devices
Wireless	WiFi, Bluetooth	Zigbee, Z-Wave (needs extra devices)
Sensors Supported	Any (reed switch, vibration sensor, etc.)	Only compatible with specific brands

◆ Why ESP32?

- **Much cheaper** than buying a commercial hub.
- **Fully customizable** with ESPHome for DIY automation.
- No brand lock-in—**works with all kinds of sensors**.

3. MotionEyeOS vs. Commercial NVR Software (e.g., Blue Iris, Synology Surveillance Station)

Feature	MotionEyeOS (Open Source)	Blue Iris (Paid) / Synology NVR
Cost (₹)	Free (₹0)	Blue Iris: ~₹6,000 (one-time) / Synology: ₹10,000+
Hardware Required	Raspberry Pi / PC	High-end NAS or dedicated PC
Camera Support	USB, IP cameras, ESP32-CAM	IP cameras only
Customization	Full (supports scripts, HA)	Limited to software options

◆ Why MotionEyeOS?

- **Free and open-source** (vs. ₹6,000+ for Blue Iris).
- Runs on **low-power Raspberry Pi** instead of a high-end PC or NAS.
- Can integrate with **Home Assistant and ESPHome for automation**.

4. Home Assistant vs. Commercial Smart Home Systems (e.g., Amazon Alexa, Google Home, Apple HomeKit)

Feature	Home Assistant (FOSS)	Alexa/Google Home/Apple HomeKit
Cost (₹)	Free (₹0)	₹5,000 – ₹10,000 per device
Cloud Dependency	No (runs locally)	Yes (needs internet)
Customization	Full	Limited to brand ecosystem
Automation	Highly flexible	Pre-defined automations only
Privacy	High (local processing)	Low (cloud-based data sharing)

◆ Why Home Assistant?

- **No cloud dependency** → **better privacy & security**.
- **No extra cost** (vs. ₹5,000 – ₹10,000 per smart home hub).

- **Highly customizable** (vs. pre-defined automations in Alexa/Google).

5. ESPHome vs. MQTT & Other Automation Platforms

Feature	ESPHome	MQTT / Tasmota / OpenHAB
Cost (₹)	Free (₹0)	Free, but requires more setup
Ease of Use	Very easy (YAML-based)	Harder (requires scripting)
Integration	Built-in for HA	Requires MQTT broker
OTA Updates	Yes	Varies by platform

◆ Why ESPHome?

- **Easier than MQTT & OpenHAB**, no need for complex scripting.
- **Perfect for ESP32-based sensors & actuators.**
- **Direct integration with Home Assistant.**

6. TinyCam Pro vs. Proprietary Camera Apps (e.g., Hikvision, Dahua, Mi Home)

Feature	TinyCam Pro (3rd Party)	Brand-Specific Apps (Hikvision, Mi, Dahua, etc.)
Cost (₹)	₹250 (one-time fee)	Free (but locked to brand)
Multi-Camera Support	Yes (any RTSP stream)	No (only brand-specific cameras)
Remote Access	Yes (with port forwarding)	Yes (but requires brand's cloud)
Custom RTSP Support	Yes	No (limited to their own models)

◆ Why TinyCam Pro?

- **Cheaper than dedicated NVR software.**
- **Can view multiple camera brands** in one app.

- Doesn't lock you into a single ecosystem.

Final Cost Summary: DoorMate vs. Commercial Alternatives

Component	DoorMate (DIY)	Commercial Alternative
Raspberry Pi 4 (NVR + Controller)	₹7,000	₹12,000 (NVR only)
ESP32 & ESP32-CAM (Sensors & Cameras)	₹1,000	₹10,000+ (Brand smart devices)
Home Assistant (Smart Hub)	₹0	₹8,000+ (SmartThings, Alexa, etc.)
MotionEyeOS (NVR Software)	₹0	₹6,000+ (Blue Iris, Synology)
ESPHome (Automation Framework)	₹0	₹0 (if using proprietary)
TinyCam Pro (Mobile Viewer App)	₹250	₹10,000+ (Standalone NVR)
Total Cost	₹8,250 – ₹9,250	₹45,000+

Key Takeaways:

- ✓ **DIY DoorMate is nearly 5x cheaper** than a commercial alternative (~₹9,250 vs. ₹45,000+).
 - ✓ Uses **open-source & repurposed hardware**, unlike proprietary systems.
 - ✓ **More flexible & customizable** than any closed-source solution.
 - ✓ **No recurring costs**, unlike cloud-based smart home services.
 - ✓ Can **reuse old hardware** instead of buying new devices.
-

→Power Consumption and Hardware Cost Breakdown

Since **DoorMate** is designed to be a **low-cost, DIY alternative** to commercial smart home and security systems, it's important to analyze its **power efficiency** and **hardware cost breakdown**. This section will compare the power consumption of each component and estimate the monthly electricity cost in **rupees (₹)**.

1. Power Consumption Breakdown

Component	Power Rating	Estimated Daily Usage	Daily Power Consumption	Monthly Power Cost (₹8/unit)
Raspberry Pi 4 (NVR & Controller)	~5W	24 hours	120 Wh (0.12 kWh)	₹29
ESP32 (Sensors & Automation)	~0.3W	24 hours	7.2 Wh (0.0072 kWh)	₹1.7
ESP32-CAM (Camera)	~1.5W	24 hours	36 Wh (0.036 kWh)	₹9
USB Camera (Connected to RPi)	~2.5W	24 hours	60 Wh (0.06 kWh)	₹14.4
WiFi Router (Network Connectivity)	~6W	24 hours	144 Wh (0.144 kWh)	₹34.5
Relay Module + LED Lights	~1W	4 hours	4 Wh (0.004 kWh)	₹0.96
Buzzer (Doorbell Alert)	~0.5W	1 hour	0.5 Wh (0.0005 kWh)	₹0.12
Total Power Consumption	~16.3W	24/7 (except some devices)	~370 Wh (0.37 kWh/day)	~₹89/month

◆ Conclusion:

- The total monthly electricity cost is under ₹90, making it affordable for continuous 24/7 operation.
- DoorMate consumes less power than a single LED bulb (~18W).

- Compared to a **commercial NVR system (typically 20W–30W), DoorMate is 30% more energy-efficient.**

2. Hardware Cost Breakdown

Component	Model / Details	Price (₹)
Raspberry Pi 4 (4GB)	₹7,000	
ESP32 (for sensors & automation)	ESP32 Dev Kit	₹500
ESP32-CAM (for camera monitoring)	ESP32-CAM AI-Thinker	₹500
USB Camera	Generic 1080p USB Camera	₹1,000
Reed Switch (Door Open/Close Sensor)	Magnetic Switch	₹150
Vibration Sensor	SW-420 Vibration Sensor	₹150
Buzzer (For Doorbell)	5V Active Buzzer	₹50
Relay Module (For Room Appliances Control)	5V Relay (Single Channel)	₹100
LED Indicators (Status LEDs)	Pack of 5	₹50
MicroSD Card (For RPi Storage)	32GB Class 10	₹500
Power Adapter for RPi (5V, 3A)	Official Power Supply	₹600
Total Cost	₹10,600	

◆ Conclusion:

- The entire **DIY smart home security system costs only ₹10,600**, while commercial NVR-based smart home setups **start at ₹40,000+**.
- Most of these components can be repurposed** from old hardware, further reducing the cost.
- Compared to **buying a Hikvision/Dahua camera system + Amazon Alexa setup**, DoorMate is **75% cheaper**.

Key Takeaways

- ✓ **Total Monthly Power Cost:** ~₹89 (Comparable to a single LED bulb).
- ✓ **Total Hardware Cost:** ~₹10,600 (vs. ₹40,000+ for commercial alternatives).
- ✓ **30% more energy-efficient than dedicated NVR setups.**
- ✓ **Can use old hardware like Raspberry Pi and ESP32, reducing cost further.**

→Installation Complexity & User-Friendliness: DIY vs. Commercial Solutions

Setting up a smart home security system typically involves **hardware installation, software configuration, and automation setup**. Let's compare **DoorMate (DIY solution)** with **commercial alternatives** like Hikvision, Dahua, Nest, and Ring in terms of **installation difficulty and user-friendliness**.

1. Installation Complexity Comparison

Task	DoorMate (DIY Solution)	Commercial NVR-Based System
Hardware Setup	Requires assembling Raspberry Pi, ESP32, sensors, and cameras manually. Needs basic electronics knowledge.	Mostly plug-and-play with pre-configured NVR and cameras.
Software Setup	Needs Docker, Home Assistant, ESPHome, MotionEyeOS , and YAML configurations.	Vendor software auto-installs with minimal setup.
Camera Integration	Manual RTSP stream setup for ESP32-CAM & USB Camera .	NVR auto-detects vendor cameras.
Sensor Configuration	Needs ESPHome flashing & integration with Home Assistant.	Plug-and-play with pre-paired sensors.
Automation Setup	Custom YAML-based automation in Home Assistant.	Pre-defined automation rules in the app.
Mobile App	Home Assistant + TinyCam PRO (third-party apps).	Dedicated mobile app for monitoring and control.

Task	DoorMate (DIY Solution)	Commercial NVR-Based System
Remote Access	Requires port forwarding or Nabu Casa for secure remote access.	Built-in cloud access from the vendor.
Power & Backup	Needs a 5V adapter & optional power bank backup .	Typically runs on PoE (Power over Ethernet) + UPS backup.
Customization & Expandability	Fully customizable: add any sensor, camera, or automation.	Limited to vendor-supported devices.
Cost	₹10,600 total (~₹90/month power cost).	₹40,000+ (~₹200–₹400/month cloud subscription).

◆ Conclusion:

- **DoorMate is more complex to set up initially** but provides **more customization**.
- **Commercial systems are plug-and-play** but **lack flexibility** and force you into their ecosystem.
- **Home Assistant & ESPHome require a learning curve**, but once set up, they **outperform commercial solutions in flexibility**.

2. User-Friendliness Comparison

Feature	DoorMate (DIY Solution)	Commercial NVR-Based System
Ease of Use	Needs basic coding & configuration (not beginner-friendly).	Easy-to-use app with guided setup.
Camera Viewing	Uses MotionEyeOS & TinyCam PRO .	Vendor's app with cloud storage.
Smartphone Alerts	Home Assistant app via notifications.	Instant push notifications.

Feature	DoorMate (DIY Solution)	Commercial NVR-Based System
Voice Control (Alexa/Google)	Can be added via Home Assistant integration.	Natively supported.
Cloud Storage	Optional local recording on Raspberry Pi or NAS .	Paid cloud subscription required (₹200–₹400/month).
Remote Access	Needs Nabu Casa (₹500/month) or VPN .	Built-in remote access from the vendor.
Automation	Fully customizable automation.	Limited preset automation.
Privacy & Security	Local processing, no data shared with third parties.	Data sent to vendor cloud, risk of breaches.

◆ Conclusion:

- **Commercial systems are more user-friendly** but require **subscriptions** and have **privacy concerns**.
- **DoorMate is harder to set up** but offers **full control, local storage, and no cloud fees**.
- **For tech-savvy users, DoorMate is the best choice.** For **non-technical users**, a commercial system is easier.

Key Takeaways

- ✓ **DoorMate is more flexible & private** but harder to set up.
 - ✓ **Commercial NVRs are plug-and-play** but expensive & cloud-dependent.
 - ✓ **DoorMate requires no monthly subscription & runs locally.**
 - ✓ **For those comfortable with tech, DoorMate is the superior long-term solution.**
-

→Communication Workflow and Protocols Used in DoorMate

DoorMate integrates **multiple smart devices, sensors, and software platforms** to provide a **seamless home security and automation experience**. The system relies on **wired and wireless communication** protocols to ensure smooth operation and real-time updates. Below is an in-depth breakdown of how communication works in the project:

1. Communication Overview

The communication in **DoorMate** is primarily based on:

- **Local communication (LAN-based)** → Faster response, more privacy, works even without the internet.
- **Cloud-based communication** → Remote access via **Home Assistant Cloud** (optional).

The devices exchange **commands, status updates, and video streams** via **wired USB connections, WiFi, MQTT, HTTP, and RTSP protocols**.

2. Communication Breakdown

Video Streaming Communication

1. USB Camera → Raspberry Pi (MotionEyeOS)

- **Protocol Used: USB Video Class (UVC)**
- **How it Works:** The USB camera is directly plugged into the Raspberry Pi, where **MotionEyeOS** captures and processes the video feed.
- **Purpose:** Provides a **local NVR solution** for recording and live-streaming.

2. ESP32-CAM → MotionEyeOS / TinyCam Pro

- **Protocol Used: HTTP (MJPEG Stream) / RTSP**
- **How it Works:** ESP32-CAM **hosts an HTTP-based MJPEG video stream** accessible via MotionEyeOS or the TinyCam Pro app.
- **Purpose:** Streams real-time video feed over the **local network**.

3. MotionEyeOS → Home Assistant / TinyCam Pro

- **Protocol Used:** RTSP (Real-Time Streaming Protocol)
 - **How it Works:** MotionEyeOS exposes the camera feed via RTSP, which can be integrated into Home Assistant or viewed via TinyCam Pro.
 - **Purpose:** Enables **live monitoring and recording** of security footage.
-

OVERVIEW OF KEY FUNCTIONS/FEATURES

Sensor Communication (ESP32 → Home Assistant)

ESP32 acts as the **sensor hub**, interfacing with multiple sensors and relaying data to **Home Assistant** via MQTT.

1. Door Bell (Push Button) → ESP32

- **Protocol Used:** GPIO Input (Digital Signal)
- **How it Works:** When the button is pressed, ESP32 registers a **high signal** on its GPIO pin.

2. Reed Switch (Door Open/Close) → ESP32

- **Protocol Used:** GPIO Input (Magnetic Contact)
- **How it Works:**
 - If the magnet is close to the switch → **Door is closed** (LOW signal).
 - If the magnet moves away → **Door is open** (HIGH signal).
- **Data Transmission:** ESP32 **publishes** door status to Home Assistant via **MQTT**.

3. Vibration Sensor (Break-in Detection) → ESP32

- **Protocol Used:** GPIO Interrupts
- **How it Works:** When vibration is detected, an interrupt signal is sent to ESP32.
- **Data Transmission:** ESP32 triggers an MQTT message to Home Assistant, which sends a **push notification alert**.

4. WiFi Status LED → ESP32

- **Protocol Used:** Internal WiFi Stack
- **How it Works:** The LED blinks at different rates depending on WiFi status (Connected, Searching, Disconnected).

Device Control Communication (Home Assistant → ESP32)

ESP32 controls devices like **LEDs and relays** based on commands received from Home Assistant.

1. Room-1 LED → ESP32

- **Protocol Used: MQTT**
- **How it Works:**
 - Home Assistant sends an **MQTT topic update** (room1/led ON/OFF).
 - ESP32 **subscribes** to this topic and toggles the LED accordingly.

2. Room-2 LED + Relay (Fan/Appliance) → ESP32

- **Protocol Used: MQTT**
- **How it Works:**
 - Home Assistant sends MQTT messages (room2/relay ON/OFF).
 - ESP32 toggles the **relay state** to turn appliances ON/OFF.

Notification & Alerts Communication

1. Home Assistant → Mobile App (HA Companion App)

- **Protocol Used: Firebase Cloud Messaging (FCM) / WebSockets**
- **How it Works:**

Home Assistant sends an **event-based notification** when an alert condition is met (e.g., break-in detected).

The notification is pushed to the **Home Assistant Companion App** via FCM.

2. Home Assistant → MQTT Broker → ESP32

- **Protocol Used: MQTT (Message Queuing Telemetry Transport)**
 - **How it Works:**
 - Home Assistant publishes commands to the **MQTT broker**.
 - ESP32 subscribes to relevant topics and takes action accordingly.
-

Summary of Protocols Used

Component	Protocol Used	Purpose
USB Camera → MotionEyeOS	USB Video Class (UVC)	Local camera streaming
ESP32-CAM → MotionEyeOS	HTTP (MJPEG)	Network-based streaming
MotionEyeOS → HA / TinyCam Pro	RTSP (Real-Time Streaming Protocol)	Live video streaming
Door Bell → ESP32	GPIO Digital Input	Detects button press
Reed Switch → ESP32	GPIO Input (Magnetic Contact)	Detects door open/close status
Vibration Sensor → ESP32	GPIO Interrupts	Detects break-ins
ESP32 → HA	MQTT (Publish/Subscribe)	Sends sensor status updates
HA → ESP32 (Relay, LEDs)	MQTT (Pub/Sub)	Controls devices
HA → Mobile App	FCM (Firebase Cloud Messaging)	Sends push notifications

Why These Protocols?

1. **USB Video Class (UVC)** → Works natively with Raspberry Pi for **direct camera integration**.
 2. **RTSP & MJPEG** → Allows **live streaming** and viewing cameras across multiple devices.
 3. **MQTT** → Efficient, **lightweight protocol** designed for **IoT communication**, making it ideal for **ESP32 sensors and actuators**.
 4. **GPIO Interrupts** → Ensures **immediate response** for security sensors like the **vibration sensor**.
 5. **FCM & WebSockets** → Enables **instant notifications** for break-in alerts and device status changes.
-

→Extensibility and Further Scope

Expanding DoorMate: Adding Extra Features for More Functionality

One of the biggest advantages of **DoorMate** over commercial solutions is its **expandability**. Since it's built on **open-source technologies (Home Assistant, ESPHome, MotionEyeOS, Docker, etc.)**, adding new features is just a matter of integrating new hardware or software modules.

Here are some **possible upgrades** to enhance DoorMate's functionality:

1. Advanced Security Features

Feature	How to Implement	Benefits
Facial Recognition	Install Frigate AI (uses TensorFlow/Coral TPU) on Raspberry Pi or an external AI processor.	Automatically recognize visitors and send alerts for unknown faces.
License Plate Recognition (LPR)	Integrate OpenALPR with MotionEyeOS.	Automatically detect vehicles entering your premises.
Intruder Detection with AI	Use DeepStack AI (runs on Docker) for object detection.	Get alerts when a person is detected near the door at unusual times.
Motion-Based Recording	Enable MotionEyeOS motion detection instead of 24/7 recording.	Saves storage by recording only when movement is detected.
2-Way Audio Communication	Add a USB microphone + speaker to Raspberry Pi.	Talk to visitors remotely via the Home Assistant app.
Tamper Detection for Cameras	Add a Gyroscope sensor (MPU6050) on the camera mount.	Detect if someone is trying to move or block the camera.

2. Home Automation Enhancements

Feature	How to Implement	Benefits
Smart Lock Integration	Use Zigbee/Z-Wave smart locks and integrate with Home Assistant.	Unlock your door remotely or automatically when a trusted person arrives.
Auto-Light Activation	Use PIR motion sensors to turn on lights automatically when movement is detected.	Saves energy by lighting up only when needed.
Temperature & Humidity Monitoring	Add a DHT11/DHT22 sensor and log data to Home Assistant.	Monitor room conditions remotely and automate climate control.
Automatic Fan Control	Use a relay + temperature sensor to turn fans on/off based on room temperature.	Reduces electricity consumption and keeps rooms comfortable.
Voice Control via Alexa/Google Assistant	Integrate Home Assistant with Alexa/Google Home .	Control devices using voice commands.

3. Enhanced Smart Notifications

Feature	How to Implement	Benefits
Door Left Open Alert	Use reed switch + Home Assistant automation to send an alert if the door is open for too long.	Prevents energy loss and improves security.
Break-in Detection Alerts	Combine vibration sensor + reed switch for detecting forced entry.	Instantly notifies users via Home Assistant app.

Feature	How to Implement	Benefits
Custom Sound Alerts	Connect a speaker module to ESP32 and play different tones for doorbell, intrusion, or sensor alerts.	Improves notification system inside the house.
SMS Alerts	Use Twilio API or GSM Module to send SMS when internet is down.	Ensures notifications even when WiFi is unavailable.

4. Storage & Cloud Integration

Feature	How to Implement	Benefits
NAS Integration	Store recordings on a Synology/QNAP NAS instead of an SD card.	More storage and reliability for video recordings.
Remote Access without Port Forwarding	Use Tailscale VPN or Home Assistant Cloud (Nabu Casa) .	Secure access to your smart home from anywhere without exposing ports.
Google Drive/Dropbox Backup	Use Rclone to automatically upload important recordings.	Prevents data loss in case of SD card failure.

5. Multi-Device Synchronization & Cross-Platform Support

Feature	How to Implement	Benefits
Multiple Camera Support	Add more ESP32-CAMs or USB cameras to MotionEyeOS.	Covers multiple entry points around the house.
Multi-Room Control Panel	Use a Raspberry Pi touchscreen as a smart home dashboard .	View camera feeds & control all devices from one screen.
Local AI Assistant (Privacy-Focused)	Install Mycroft AI on Raspberry Pi.	Offline voice control without sending data to Google/Amazon.

Future Scalability: What's Next for DoorMate?

DoorMate is **future-proof**, meaning you can always **add new modules and upgrade existing ones**. Some long-term upgrades could include:

- **Integration with smart doorbells like Ring/Nest (via Home Assistant).**
 - **Energy monitoring for connected appliances (ESP32 + CT sensors).**
 - **Full automation with time-based schedules for lights, fans, and security alerts.**
-

Key Takeaways

- ✓ **DoorMate is highly expandable**—just add new sensors, cameras, or software integrations.
- ✓ **It can be customized for security, automation, notifications, or even AI-based recognition.**
- ✓ **The system remains cost-effective even with upgrades**, unlike commercial solutions that require buying new proprietary hardware.

→ Conclusion

DoorMate: Your Smart Visitor Assistant has successfully demonstrated the power of **open-source technologies** and **cost-effective home automation** using **Raspberry Pi 4, ESP32-CAM, and Docker**. The project provides a **secure, scalable, and highly customizable** solution for **real-time visitor monitoring, security automation, and home control**.

By integrating **Home Assistant, ESPHome, and MotionEyeOS**, DoorMate enables seamless control over multiple devices such as **door sensors, vibration detectors, smart lights, and relays**, all while offering a robust **video surveillance system** accessible via **tinyCAM PRO**. The **notification system** ensures that users are alerted immediately in case of a potential break-in, enhancing overall home security.

A key highlight of the project is its **use of Docker**, which simplifies deployment and **allows easy expansion**. Additional services such as **AI-powered video analytics (Frigate AI, DeepStack)**, **cloud storage (Nextcloud, Rclone)**, and **automation tools (Node-RED, MQTT, Zigbee2MQTT)** can be added effortlessly, making DoorMate a **future-proof smart home system**.

Furthermore, this project showcases the benefits of **FOSS (Free and Open Source Software)**, enabling users to repurpose **old hardware as an NVR** instead of investing in costly surveillance systems. This **reduces e-waste and promotes sustainability** while ensuring full control over data privacy.

→Future Scope

While DoorMate is fully functional, there is still potential for further **enhancements**, such as:

- ✓ **AI-powered face recognition** to identify frequent visitors
- ✓ **Automated smart door lock integration** for enhanced security
- ✓ **Energy monitoring and power consumption analytics**
- ✓ **Voice assistant integration (Google Assistant, Alexa)**
- ✓ **Expansion to multi-room or multi-house security setups**

With its **scalability, cost-effectiveness, and user-friendly control**, DoorMate proves to be an **ideal home automation and security solution** for anyone looking to enhance their smart home experience.