OpenGate: A Privacy-Focused, Open-Source Smart Doorbell Camera

DANIIL KATULEVSKIY, University of California, San Diego, USA
ANDREW PEGG, University of California, San Diego, USA

Commercial smart home devices frequently compromise user privacy, with major brands like Ring and Nest facing lawsuits over their data practices. OpenGate is a fully open-source smart doorbell created to solve this, emphasizing on-device processing and complete user data control. Our system provides a robust, privacy-preserving alternative by integrating real-time bidirectional audio/video, local AI for person detection, and seamless automation with platforms like Home Assistant via MQTT. OpenGate empowers users to build a trustworthy and customizable smart security solution, demonstrating a viable model for secure, open-source IoT that is free from vendor lock-in and surveillance concerns.

CCS Concepts: • **Security and privacy** → **Embedded systems security**; • **Computer systems organization** → **Embedded systems**; • **Computing methodologies** → *Computer vision.*

Additional Key Words and Phrases: smart doorbell, privacy, embedded systems, Qualcomm RB3, computer vision, IoT, open source, GStreamer, YOLOv8, n8n, Home Assistant, MQTT

ACM Reference Format:
Daniil Katulevskiy and Andrew Pegg. 2025. OpenGate: A Privacy-Focused, Open-Source Smart Doorbell Camera. 1, 1 (June 2025), 11 pages. https://doi.org/unpublished

1 INTRODUCTION

Smart home devices have significantly transformed daily living, with smart doorbells emerging as a cornerstone of modern home security and convenience. These devices offer features such as remote video monitoring, two-way communication, and motion-triggered alerts, enhancing situational awareness and security for homeowners. However, the current market is dominated by commercial offerings that, while feature-rich, present several inherent limitations. Many popular smart doorbells, such as those from Ring and Google Nest, often operate within closed ecosystems, leading to vendor lock-in and restricting user customization. Furthermore, reliance on cloud-based services for video storage and advanced AI features frequently necessitates ongoing subscription fees, increasing the total cost of ownership. Perhaps most critically, these commercial systems have raised significant data privacy concerns, stemming from extensive data collection practices, the potential for unauthorized data access or sharing with third parties, and the opacity of their internal workings.

An explicit demonstration of the danger of using such products are lawsuits by the FTC against major industry players like Ring and Google Nest [13, 14]. The complaint against Ring charged the company with the following:

> ...compromising its customers' privacy by allowing any employee or contractor to access consumers' private videos and by failing to implement basic privacy and security protections, enabling hackers to take control of consumers' accounts, cameras, and videos. [13]

Authors' addresses: Daniil Katulevskiy, dkatulevskiy@ucsd.edu, University of California, San Diego, La Jolla, California, USA; Andrew Pegg, apegg@ucsd.edu, University of California, San Diego, La Jolla, California, USA.

ACM XXXX-XXXX/2025/6-ART
https://doi.org/unpublished

This case highlights the severe privacy and security lapses that can occur in closed-source, commercial IoT ecosystems. In response to these challenges, the OpenGate project was conceived to develop a powerful, customizable, and privacy-respecting open-source smart doorbell alternative. The project leverages the computational capabilities of the Qualcomm RB3 development platform, a robust embedded system well-suited for demanding tasks such as real-time multimedia processing and edge artificial intelligence [8, 9]. The primary goal of OpenGate is to provide users with full control over their hardware, software, and data, fostering transparency and enabling community-driven development. This approach not only addresses the shortcomings of proprietary systems but also serves as an educational platform for exploring advanced concepts in embedded systems, computer vision, and IoT integration. The decision to build an open-source solution is a direct countermeasure to the "black box" nature of many commercial devices, empowering users to understand, modify, and extend the system according to their specific needs.

The OpenGate system integrates a comprehensive suite of features expected from a modern smart doorbell. It provides high-quality, low-latency video streaming and bidirectional audio communication, allowing users to see and interact with visitors remotely. A key innovation lies in its on-device AI capabilities, including real-time object detection using the YOLOv8 model. This local processing minimizes reliance on cloud services for core AI functionalities, enhancing privacy and reducing latency. The system incorporates a physical doorbell button that triggers notifications and custom automation workflows. A significant aspect of OpenGate's design is its flexible integration with open-source smart home automation platforms. Initially, integration with proprietary systems like Google Home was considered, but due to licensing complexities and a desire to maintain a fully open ethos, the project pivoted to embrace n8n for visual workflow automation and Home Assistant via the MQTT protocol for robust event-driven control. This shift not only resolved practical issues but also reinforced the project's commitment to open standards and user empowerment.

The contributions of the OpenGate project are summarized as follows:

- **Fully Open-Source Smart Doorbell System:** Development of a complete hardware and software solution where all components are open, promoting transparency, auditability, and community collaboration. The project provides a blueprint for users wishing to build their own privacy-centric smart security devices.
- **Advanced Embedded Multimedia Processing:** Successful implementation and optimization of real-time video and bidirectional audio streaming on the Qualcomm RB3 platform using the GStreamer multimedia framework and MediaMTX for stream distribution. This demonstrates the RB3's capability to handle complex media pipelines efficiently.
- **On-Device Artificial Intelligence:** Deployment of the YOLOv8 object detection model directly on the embedded platform. This showcases practical edge AI application for enhanced event understanding (e.g., person detection, familiar face identification) without mandatory cloud dependency.
- **Flexible Open-Source Automation Integration:** Novel and robust integration with leading open-source automation platforms, n8n and Home Assistant (via MQTT). This offers users unparalleled flexibility in creating custom alerts, responses, and interactions with other smart home devices, far exceeding the often-rigid ecosystems of commercial products.
- **Privacy-Focused and Cost-Effective Design:** By prioritizing local processing, open standards, and avoiding mandatory subscription fees, OpenGate presents a practical and economically viable approach to smart home security that respects user data privacy.

This report details the design, implementation, and successful completion of the OpenGate project. Section 2 reviews existing commercial and open-source smart doorbell solutions, positioning OpenGate within this landscape. Section 3 provides an in-depth description of the system's architecture, hardware, software modules, and key technical achievements. Section 4 outlines the project's development journey, from initial plans to the successful completion of all objectives. Finally, Section 5 summarizes the work, highlights its significance, and

discusses potential avenues for future development. The project's source code and detailed setup instructions are available in its GitHub repository.

2 RELATED WORKS

The smart home security landscape is a complex ecosystem encompassing a range of solutions from passive, large-scale surveillance systems to interactive, user-focused devices. To contextualize the contributions of the OpenGate project, it is essential to analyze the dominant product categories, including open-source Network Video Recorders (NVRs) and the feature-driven commercial smart doorbell market.

2.1 Open-Source Network Video Recorder (NVR) Solutions

Before the rise of integrated smart doorbells, open-source projects like ZoneMinder [15] and, more recently, Frigate [3] set the standard for self-hosted video surveillance. These platforms operate as software NVRs, designed to process video streams from multiple IP cameras. Their primary strength lies in providing robust, passive security. Frigate, for example, represents a modern approach, leveraging AI-based object detection on local hardware to minimize false positives from motion events.

While powerful for surveillance, these NVR solutions are not designed to be interactive doorbell cameras. They lack the built-in, real-time, two-way audio communication and immediate, actionable notifications for guest interactions that define the doorbell experience. They are fundamentally passive systems for observation, whereas a smart doorbell must be an active bridge for communication with visitors.

2.2 Commercial Smart Doorbells

The commercial market is dominated by tech giants who compete on features, ease of use, and ecosystem integration. Key features common across leading brands like Ring, Google Nest, and Eufy include HD video, two-way audio, and cloud-based event history. However, companies differentiate themselves with specialized technologies.

- **Ring**, an Amazon company, promotes its subscription plan which enables features like "Color Pre-Roll," a technology that captures video seconds before a motion event is triggered to provide more context [10].
- **Google Nest** leverages its deep expertise in artificial intelligence. Google has detailed how its Nest doorbells use on-device machine learning models to differentiate between people, packages, animals, and vehicles for smarter notifications [2].
- **Eufy** has built its brand identity around the promise of "No Monthly Fee" by emphasizing local storage on a device-specific base station or on the doorbell itself [4].

Despite these innovations, the commercial space is rife with challenges related to privacy and cost. The dependence on subscriptions is a significant drawback for many users, and the closed-source nature of these products creates opacity, as underscored by the FTC's legal actions [13].

Further compounding these issues are the inherent privacy vulnerabilities in how these devices handle encrypted traffic and cloud connections. Apthorpe et al. [1] demonstrated that even when smart home devices like the Nest Cam Indoor use TLS encryption, metadata such as traffic rates and DNS queries can leak sensitive behavioral information. Their analysis showed that a passive network observer—such as an Internet Service Provider (ISP)—can infer when a user is asleep, when they're interacting with their security camera, or when motion is detected in their home, all without decrypting a single packet. These findings underscore the inadequacy of encryption alone and reveal a critical blind spot in commercial device design: the lack of transparency and auditability in proprietary systems prevents users and researchers from fully understanding and mitigating privacy risks. Without access to open code or formal guarantees, consumers are left to trust that companies are not only competent in their implementations, but also motivated to protect against these metadata-based attacks.

2.3 Positioning of the OpenGate Project

The OpenGate project synthesizes concepts from both the NVR and commercial doorbell domains. It aims to provide the advanced, AI-driven event detection found in modern NVRs like Frigate, but packaged into an interactive, real-time communication device like a commercial doorbell. By building on a powerful embedded platform (the Qualcomm RB3), OpenGate can perform sophisticated, low-latency AI processing directly on-device, mirroring the efficiency of Google's on-device models but within a fully open and transparent framework.

Table 1 provides a summary of how OpenGate compares to the established categories in the smart security field. OpenGate's primary contribution is its creation of a system that is not only technically capable but also fundamentally trustworthy and adaptable, directly addressing the privacy, cost, and interoperability limitations of existing solutions.

Table 1. Comparative Analysis of Smart Security Solutions.

Category	Key Characteristics	Limitations
Open-Source NVRs (e.g., Frigate, ZoneMinder)	High-accuracy AI object detection and continuous recording. Self-hosted and fully user-controlled. Excellent for passive, multi-camera surveillance.	Not designed for real-time guest interaction. Lacks integrated two-way audio and doorbell-specific logic. Requires separate camera hardware.
Commercial Doorbells (e.g., Ring, Google Nest)	User-friendly with polished apps. Advanced, proprietary features (e.g., Pre-Roll, on-device AI alerts). Strong integration within their own smart home ecosystems.	Cloud-dependent, often requiring subscriptions for core features. Significant privacy concerns and vendor lock-in. Limited customizability.
OpenGate Project	Combines NVR-grade on-device AI with doorbell interactivity. Fully open-source and self-hosted, ensuring privacy and control. Integrates with open standards (MQTT) for maximum flexibility.	Requires more technical expertise for setup compared to commercial products. Hardware cost may be higher upfront than some subsidized commercial devices.

3 TECHNICAL MATERIAL

This section details the architecture, hardware platform, software implementation, and performance characteristics of the OpenGate smart doorbell system. The design philosophy emphasizes local processing, modularity, and the use of open standards to ensure flexibility and user control.

3.1 System Architecture Overview

The OpenGate system is architected as a set of interconnected modules running on the Qualcomm RB3 platform. Software modules responsible for video and audio streaming, AI processing (object detection), doorbell event handling, notification dispatch, and smart home integration orchestrate the system's functionality. Data flows originate from the camera and microphone, are processed by the GStreamer pipelines, and can be optionally fed into the AI inference engines. Events, such as a doorbell press or AI-detected activity, trigger notifications and can be propagated to external automation platforms (n8n, Home Assistant) via MQTT. The system prioritizes

local processing for core functions like AI detection and streaming management to enhance privacy and reduce latency.

3.2 Hardware Platform: Qualcomm RB3

The project is built upon the Qualcomm Robotics RB3 Platform, chosen for its potent processing capabilities suitable for demanding embedded applications, including real-time multimedia and AI. The RB3 features a Qualcomm QCS6490 system-on-chip (SoC) with a multi-core CPU, Adreno GPU, and a Neural Processing Unit (NPU). The platform runs a customized version of Linux, providing the necessary utilities and drivers to interact with native hardware accelerators on the RB3 platform. Hardware setup involved flashing the RB3 board with the appropriate operating system image (QCom-Linux IMSDK 1.4), connecting the camera module, speakers for audio output, a microphone for audio input, and wiring a physical push-button to the RB3's General Purpose Input/Output (GPIO) pins to serve as the doorbell trigger. This setup provides all the necessary physical interfaces for a fully functional smart doorbell.

3.3 Software Implementation & Core Modules

OpenGate Software is comprised of multiple key components: audio/video streaming pipelines, a dynamic re-encoding turn server, GPIO handler, communication modules, miscellaneous software built for this project.

3.3.1 Video and Audio Streaming. Real-time, low-latency video and bidirectional audio streaming are critical functionalities of OpenGate. This was achieved using the GStreamer multimedia framework and MediaMTX. **GStreamer Framework:** GStreamer is a powerful, pipeline-based multimedia framework that allows developers to construct complex media processing workflows by linking together various processing elements [11]. Elements are modular plugins that perform specific tasks (e.g., capturing from a source, encoding, decoding, rendering). OpenGate uses multiple plugins, some provided by Qualcomm, this is described below. **Video Pipeline:** A GStreamer pipeline was designed to capture video from the RB3's camera module. We utilized Qualcomm's 'qtiqmmfsrc' plugin, an encoder 'v4l2h264enc' for hardware-accelerated H.264 encoding, 'h264parse' for parsing the frames from H264 encoding, and 'mpegtsmux' to package the frames in a format that MediaMTX will understand.

Fig. 1. Video Streaming Pipeline

Audio Pipeline (Bidirectional): Separate GStreamer pipelines were implemented for audio. One pipeline captures audio from the microphone using 'pulsesrc', reencodes and converts it using 'audioconvert', 'audioresample', 'lamemp3enc', 'mpegaudioparse', and streams it out in correct format using 'mpegtsmux' to MediaMTX. Another pipeline receives an audio stream from the network, decodes it, and plays it back through the RB3's speakers, enabling two-way communication, using 'rtspsrc' to get the audio stream over rtsp, 'rtpmpadepay', 'mpg123audiodec' and 'mpegaudioparse' to reformat the audio stream in appropriate format, and 'pulsesink' to output the audio on the device speakers.

MediaMTX Integration: MediaMTX serves as a ready-to-use, zero-dependency real-time media server. It takes the raw GStreamer audio and video streams (published via UDP) and repackages them into various widely supported protocols like RTSP, RTMP, HLS, and WebRTC. This allows diverse clients (web browsers, mobile apps,

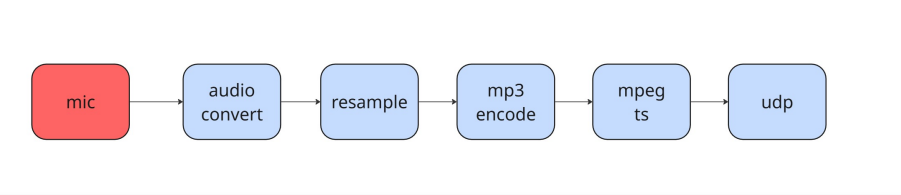

Fig. 2. Audio Streaming Pipeline

media players) to easily consume the streams. MediaMTX is not pre-installed on device, so a binary needs to be downloaded onto the system. Supported protocols and definitions for MediaMTX operations are defined in a global mediamtx.yaml file. Our approach with MediaMTX was to export our video streams into the widest amount of formats possible to allow for wider range of integrations with other devices and viewers. Our MediaMTX setup is designed to require as little processing power as possible. We do that by not repackaging streams by default, unless a stream is requested in that format. Moreover, we leverage MediaMTX's power of run-on-demand to handle certain streaming pipelines, such as audio receiving pipeline, only running them when requested.

3.3.2 Doorbell Event Handling and Notification System. **Physical Button Integration:** A physical push-button was connected to the RB3's GPIO pins 2 (ground) and 21 (generic GPIO) to act as the doorbell trigger. A Python script, encapsulated within a Docker container for consistency, monitors the GPIO pin state. GPIO was exported from udev on the host to the Docker image to access GPIO states. Docker also allowed to install extra python packages to handle data, presses, and notification sending, since installing python packages is not supported directly on the system. The script includes logic for debouncing the button input to ensure a single notification event is generated per press. **Notification Delivery:** Upon a confirmed button press, the system sends a push notification to the user's designated devices using ntfy.sh, a simple, open-source, publish-subscribe notification service that can be self-hosted.

3.3.3 Artificial Intelligence Capabilities. A significant feature of OpenGate is its on-device AI processing for object and face detection, enabling more intelligent event interpretation.
Object Detection with YOLOv8: The You Only Look Once (YOLO) family of models are renowned for their speed and accuracy in real-time object detection. YOLOv8, developed by Ultralytics [12].
Running detections on NPU: Since we wanted to utilize the processing power of the on-board NPU, we had to optimize the model for real time inference and make it suitable to work with the NPU. First, we downloaded a tflite version of YOLO provided by Qualcomm. Then the model was quantized using QNNSDK by Qualcomm, from FP16 to Q8. Since it is optimized and quantized, it can run real time inference on the NPU, 2.5-3 ms average inference time, with accurate detections for people and other needed classes.
Face Recognition: To complement object detection, a face recognition capability was added externally, not integrated with onboard AI system, using a face_recognition github repository [5]. For it to work, a user has to put images of people to recognize into a specified directory on host machine, which is very simple to do for any user.

3.3.4 Smart Home Integration and Automation. A core design principle of OpenGate is flexible and powerful integration with smart home ecosystems, achieved through open-source platforms.
n8n Automation Platform: n8n is an open-source, extendable workflow automation tool that allows users to connect various apps and services using a visual, node-based interface [7]. For OpenGate, n8n was deployed to create sophisticated automation sequences. Example workflows developed for the project include: upon a person being detected, capture a snapshot, send this snapshot to a Vision Language Model (LLM) to generate a textual

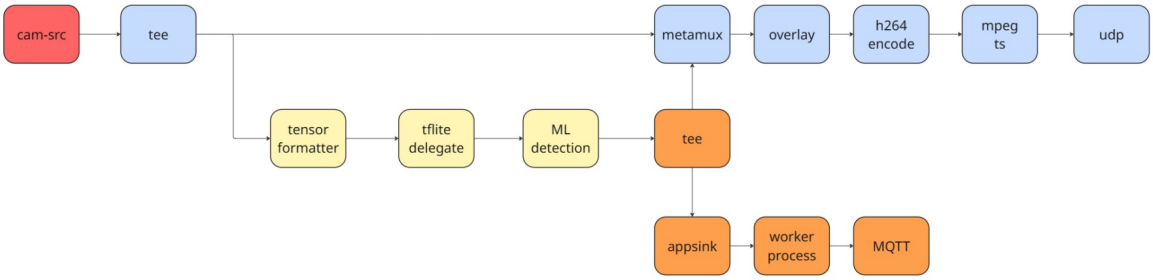

Fig. 3. Streaming pipeline with NPU detections & MQTT

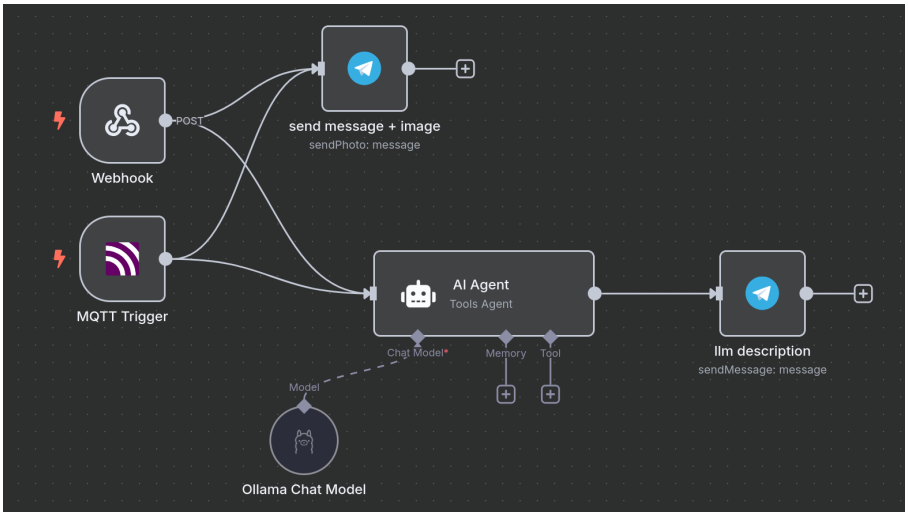

Fig. 4. Example n8n automation

description, and forward this description to notification services like Telegram.

Home Assistant Integration via MQTT: Home Assistant is a leading open-source home automation platform that prioritizes local control [6]. OpenGate integrates with it by publishing events (e.g., "doorbell pressed," "person detected") to specific MQTT topics. Home Assistant, subscribed to these topics, can then trigger automations, update dashboards, or control other connected devices (e.g., turn on porch lights when a person is detected at night). This MQTT-based integration ensures robust, real-time communication and allows OpenGate to become a seamless part of a larger, user-controlled smart home environment.

Setup Process: To make automation systems modular and possible to setup on any machine, we made a configurable docker-compose file that launches and sets up all the required components for the automations with a single command. The only requirement is Docker installed on the machine. So the process of adding to and building on top of existing automations is also straightforward and goes in line with the project's holistic extensibility approach.

3.4 MQTT

To enable communication between OpenGate and external platforms such as n8n and HomeAssistant, we integrated MQTT onto the RB3 board. Given that MQTT is widely adopted as a standard protocol for IoT devices and smart home integrations, its presence was essential for seamless interoperability. To achieve this, we developed a Python wrapper around the Paho MQTT C++ library, cross-compiled it for the RB3 using an ARM toolchain along with pybind11, and transferred the resulting shared object file onto the board. This wrapper allows OpenGate to publish messages directly to any MQTT topic, enabling functionalities like sending object detections and triggering automations within n8n and HomeAssistant environments.

3.5 Frontend UI Client

We used Go and Wails framework to create a cross-platform, system agnostic frontend UI client. It allows any user on any platform to interact with OpenGate smart doorbell. User is able to watch the stream from their camera, receive audio from the device microphone, and send audio to the doorbell speakers to easily communicate with the guests. Works in real time, with minimal delays. We made the custom UI client to get around missing encodings preventing us from using webRTC, so having a web app in browser was not possible.

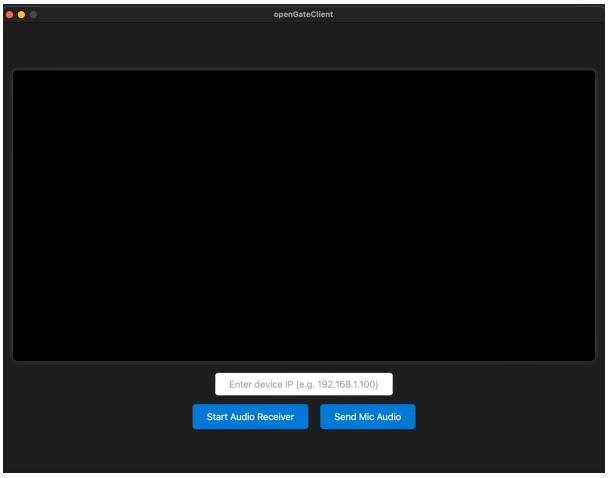

Fig. 5. Frontend UI Client

3.6 Difficulties and Pain Points

During system development, we encountered a series of difficulties and pain points when interacting with RB3.
Package Management: One significant challenge we faced while developing OpenGate was the absence of a package manager. For instance, when creating the GPIO interface, we found that the board lacked native software support for handling raw GPIO operations. Consequently, we were forced to rely on Docker containers to access necessary packages instead of running the interface directly on the board. Introducing a package manager would streamline this process by eliminating the dependency on Docker for resolving missing packages, significantly accelerating development and enabling faster iterations.
Interacting with NPU: Interacting with the NPU presented unexpected challenges primarily due to incomplete documentation about the necessity of a display connection when using specific GStreamer plugins. Initially, Qualcomm's documentation suggested that a physical display might only be required when using display-oriented

sinks, such as those based on Wayland. However, when we integrated plugins like qtimlvconverter and qtimltflite, we discovered they would fail to start due to null EGL instances unless a physical HDMI connection was present. This forced us into workarounds involving manually connecting an HDMI device solely to initiate Wayland sessions. Introducing a clearly documented headless mode for running AI inference pipelines would significantly streamline development and reduce unnecessary complexity.

Missing codecs: A significant frustration during the audio streaming pipeline development was the absence of essential audio codecs, specifically AAC and Opus. The lack of AAC encoding and decoding was particularly problematic, as AAC is widely used and essential for audio transmission, notably in protocols like HLS. Additionally, the absence of the Opus codec prevented native WebRTC audio connections, limiting us to external clients relying on protocols such as RTSP. Including these codecs would greatly enhance compatibility, simplify development, and enable a more seamless, browser-based streaming experience.

4 MILESTONES

This section documents the project's evolution, from its initial conception through various development stages to its successful completion. The project adapted to challenges and feedback, ultimately delivering a fully functional system that met all its objectives.

4.1 Initial Project Plan

The project was initially structured with a 10-week plan, with distinct milestones and ownership to ensure systematic progress.

Table 2. Original Project Milestones.

Week	Milestone	Description	Owner
1	Hardware Setup	RB3 hardware flashing and OS upgrade + speaker connection	Andrew
2	Stream Test	Validate audio/video functionality	Andrew
3	Doorbell Trigger	Physical input detection logic	Daniil
4	Notification System	Send local network alerts	Daniil
5	Face/Object Detection	Implement model on RB3	Andrew
6	Optimization	Improve latency/performance	Andrew
7	Smart API	Google Home integration	Daniil
8	Event Logging	Store local ring/log data	Daniil
9	Power Saving	Implement low-power standby	Andrew
10	Final Demo	Full system + recording	Both

4.2 Project Feedback and Adaptations

Early in the project, feedback was received emphasizing the need to "think carefully about how to test and demonstrate various deliverables." This constructive input was addressed by enhancing team communication through more frequent meetings and maintaining a detailed, shared action plan. For demonstrating deliverables, the team created both component-specific and holistic video demonstrations. This feedback helped the project become more organized and product-oriented.

4.3 Adjusted Project Plan and Final Execution

Based on initial progress and encountered challenges, the milestones were adjusted.

Table 3. Adjusted Project Milestones.

Week	Milestone	Description	Owner
1	Hardware Setup	RB3 hardware flashing and OS upgrade + speaker connection	Both
2	Stream Test	Validate audio/video functionality	Andrew
3	Doorbell Trigger	Physical input detection logic	Daniil
4	Notification System	Send local network alerts	Daniil
5	Object Detection on NPU	Implement detection model	Andrew
6	Optimization	Improve latency/performance	Andrew
7	Smart API	n8n automations integration	Daniil
8	Events and Smart Home	Send events through and allow for home integrations	Andrew
9	Face Recognition, Vision LLM	Add face recognition and Vision LLM to n8n automations	Daniil
10	Final Demo	Full system + recording	Both

All milestones from this adjusted plan have now been fully completed according to schedule. The project successfully transitioned from development to a feature-complete state. The most significant adjustment was the shift away from the Google Home API due to licensing complexities. Consequently, the project embraced n8n and Home Assistant via MQTT. The Event Logging milestone was merged into this smart home integration work, as platforms like Home Assistant provide robust logging mechanisms. The Power Saving milestone was also redefined to focus on software optimizations like on-demand streaming, which were successfully implemented.

4.4 MVP Completion and Overall Status

The original Minimum Viable Product (MVP) goal was to create a smart doorbell capable of notifying a user of a button press, providing a live video feed, and enabling two-way audio communication. All aspects of this MVP were successfully achieved early in the project. Beyond the MVP, the project has delivered on all its extended goals. The OpenGate system is now a fully functional smart doorbell camera with advanced features, and all previously incomplete or in-progress milestones are now fully completed.

5 CONCLUSION

The OpenGate project has successfully culminated in the development of a feature-rich, open-source smart doorbell camera system. Built upon the capable Qualcomm RB3 embedded platform, the project systematically addressed its core objectives: delivering real-time audio/video streaming, implementing on-device artificial intelligence, and enabling flexible integration with leading open-source smart home automation platforms. This work demonstrates a viable and powerful alternative to commercial smart doorbell solutions, prioritizing user control, data privacy, and system transparency.

The significance of OpenGate lies in its holistic approach to open design. By combining a powerful embedded Linux environment with modern AI models like YOLOv8, and by leveraging open standards such as MQTT for interoperability with Home Assistant and n8n, the project offers a degree of customization and user empowerment not commonly found in off-the-shelf products. The successful reduction of streaming latency, the implementation of on-demand streaming, and the effective deployment of AI functionalities directly on the RB3 platform are key technical achievements. All project milestones, including advanced optimizations and final integrations, were met, resulting in a fully functional and robust system. The project's journey, including its adaptation to challenges such as proprietary API limitations by pivoting to superior open-source alternatives, underscores the resilience and benefits of an open development philosophy.

While OpenGate stands as a complete and functional system, several avenues for future work could further enhance its capabilities and impact:

- **Expanded Peripheral and Protocol Support:** Investigating the integration of additional hardware, such as Zigbee or Z-Wave controllers via USB dongles, could allow OpenGate to act as a more comprehensive smart home hub.
- **Enhanced Power Management for Battery Operation:** Future hardware revisions (e.g. RUBIK Pi 3), power states management could explore the feasibility of a battery-operated version.
- **User Interface and Experience Refinements:** Development of a dedicated, user-friendly web interface for configuration.
- **User Interface and Experience Refinements:** Working with Qualcomm to add codecs support such as Opus and AAC to support fully web based streaming and audio client.
- **Community Building and Feature Expansion:** As an open-source project, fostering a community of users and developers could lead to the contribution of new features, expanded device support, and ongoing maintenance, ensuring the project's long-term viability.

In summary, the OpenGate project makes a valuable contribution to the open-source smart home landscape. It provides a well-documented, high-performance, and privacy-conscious smart doorbell solution that empowers users with control over their security and data.

REFERENCES

[1] Noah Apthorpe, Dillon Reisman, and Nick Feamster. 2017. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. arXiv:1705.06805 [cs.CR] https://arxiv.org/abs/1705.06805

[2] Ashish Arora and Vinay Shet. 2019. Smarter alerts for your Nest cameras. Google AI Blog. https://ai.googleblog.com/2019/08/smarter-alerts-for-your-nest-cameras.html

[3] Blake Blackshear. 2024. Frigate: NVR with Realtime Local Object Detection. GitHub Repository. https://github.com/blakeblackshear/frigate

[4] Eufy. 2024. eufy Security - Local Storage, No Monthly Fee. Product Marketing. https://eufy.com/security

[5] Github User Ageitgey. 2020. Face Recognition. Website. https://github.com/ageitgey/face_recognition

[6] Home Assistant. 2024. MQTT Integration. Home Assistant Documentation. https://www.home-assistant.io/integrations/mqtt/

[7] n8n. 2024. Flexible AI workflow automation for technical teams. n8n.io. https://n8n.io/

[8] Qualcomm. 2024. Gst-AI face detection. Developer Documentation. https://docs.qualcomm.com/bundle/publicresource/topics/80-70018-50/gst-ai-face-detection.html

[9] Qualcomm. 2024. Gst-AI face recognition. Developer Documentation. https://docs.qualcomm.com/bundle/publicresource/topics/80-70018-50/gst-ai-face-recognition.html

[10] Ring. 2024. Advanced Pre-Roll and Color Pre-Roll. Support Article. https://support.ring.com/hc/en-us/articles/360049109372-Advanced-Pre-Roll-and-Color-Pre-Roll

[11] The GStreamer Project. 2024. GStreamer: Open Source Multimedia Framework. Website. https://gstreamer.freedesktop.org/

[12] Ultralytics. 2024. Ultralytics YOLOv8 Docs. Documentation. https://docs.ultralytics.com/

[13] U.S. Federal Trade Commission. 2023. FTC Says Ring Illegally Surveilled Customers, Failed to Stop Hackers from Taking Control of Users' Cameras. Press Release. https://www.ftc.gov/news-events/news/press-releases/2023/05/ftc-says-ring-illegally-surveilled-customers-failed-stop-hackers-taking-control-users-cameras

[14] Davey Winder. 2021. Google Nest Camera Hacking Lawsuit Given Green Light By U.S. Judge. Forbes. https://www.forbes.com/sites/daveywinder/2021/04/18/google-nest-camera-hacking-lawsuit-given-green-light-by-us-judge/

[15] ZoneMinder. 2024. ZoneMinder - The top Linux video camera security and surveillance solution. https://zoneminder.com/