Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

Submitted to
Alan Pisano
8 St. Mary's Street Boston
Room 522
(617) 353-6264
apisano@bu.edu

Smart Home Control System

Team 27
Domus

Venkata Sai Sreeram Andra vssa8989@bu.edu
Anirudh Singh ansingh@bu.edu
Vansh Bhatia vansh@bu.edu
Akhil Bongu akbongu@bu.edu

Submitted On: 18th April 2025

Domus User Manual

Table of Contents

Executive Summary

Domus is a smart home and security system built on the Raspberry Pi platform. It integrates home automation with robust security features by combining industrial IoT components and a hardware-software co-design framework. With an edge-deployed Large Language Model processing user commands locally, Domus delivers rapid response times, enhanced privacy, and reliable control over smart devices and security systems without cloud dependency.

1 Introduction

Domus is a unified smart home control system engineered to simplify and elevate everyday living. Designed in response to the fragmented landscape of modern home automation solutions, Domus integrates environmental monitoring, intelligent surveillance, and power management into one cohesive, privacy-focused platform. Unlike conventional systems that rely heavily on cloud infrastructure, Domus operates on an edge-computing model, offering immediate responsiveness, enhanced reliability, and complete data ownership for the user.

The system revolves around a robust hardware-software architecture: a Raspberry Pi 5 acts as the central processor, supported by a Raspberry Pi 4 for video processing, and a network of ESP32 microcontrollers distributed throughout the home. These microcontrollers collect environmental and movement data via sensors, including a motion sensor that detects activity within a 7-meter range and a BME688 sensor that measures temperature, humidity, air pressure, and gas resistance to calculate an Air Quality Index (AQI). This information is processed and displayed locally through a user-friendly iOS mobile application built using React Native.

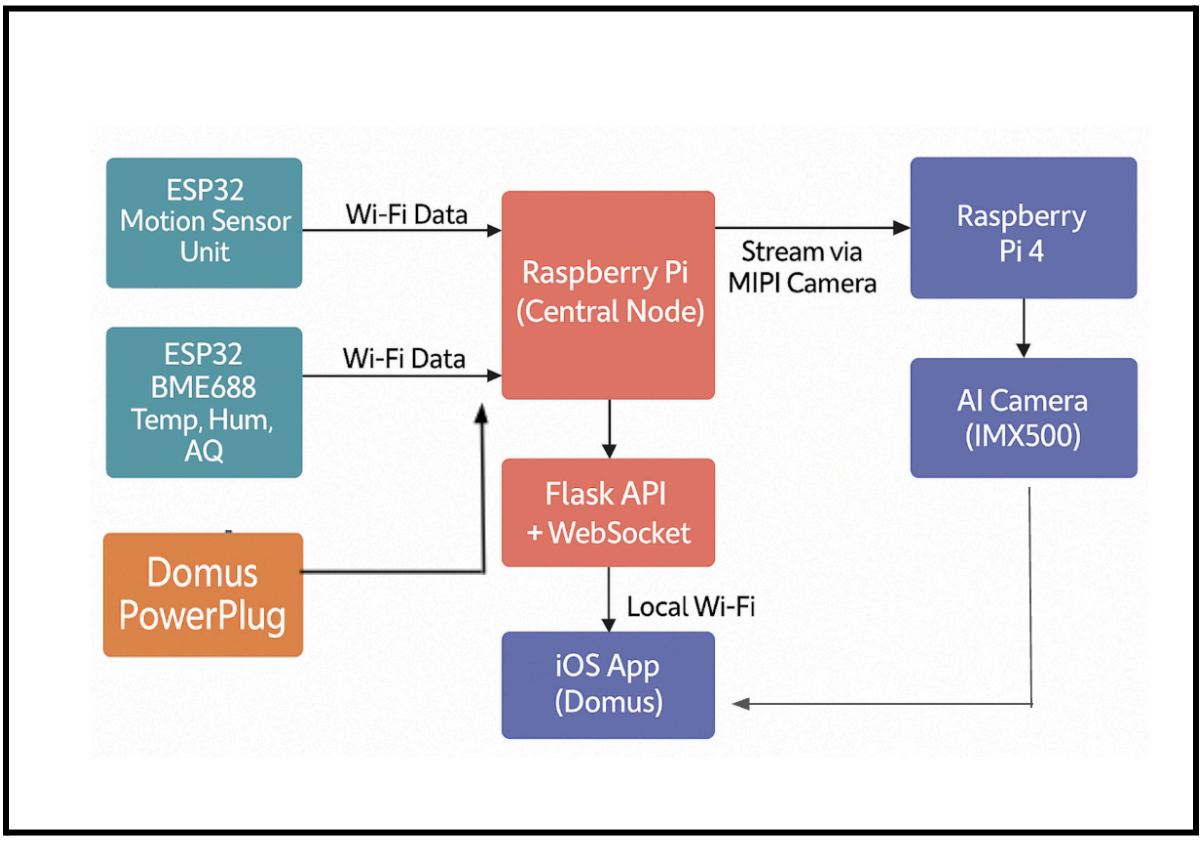
From the user’s perspective, the Domus app offers a streamlined interface with four main screens. The **Home** screen features a set of visual cards that provide real-time summaries of the household environment by pulling live data from the connected smart plug, motion sensor, BME688 environmental sensor, and AI-enabled surveillance camera. The **Energy** screen visualizes both instantaneous and historical energy usage trends using dynamic graphs. A built-in rolling average threshold alerts the user when their power consumption exceeds typical patterns, promoting energy awareness and sustainability.

The **Devices** screen allows users to effortlessly pair and manage sensors on the same network, while the **Surveillance** screen enables live monitoring through the system’s AI camera. This module uses object detection and facial recognition to identify movement or unfamiliar individuals. When an intruder or unknown object is detected, the system processes the information in real-time and sends immediate alerts to the user with contextual visuals, including detection highlights and confidence-based annotations.

Domus addresses not only automation but also the vital need for home safety, energy conservation, and user-centric design. With all computations processed locally and no cloud dependency, Domus ensures your home’s data remains secure. The sections that follow will walk you through the system setup, usage, technical background, and best practices for safe and effective operation.

2 System Overview and Installation

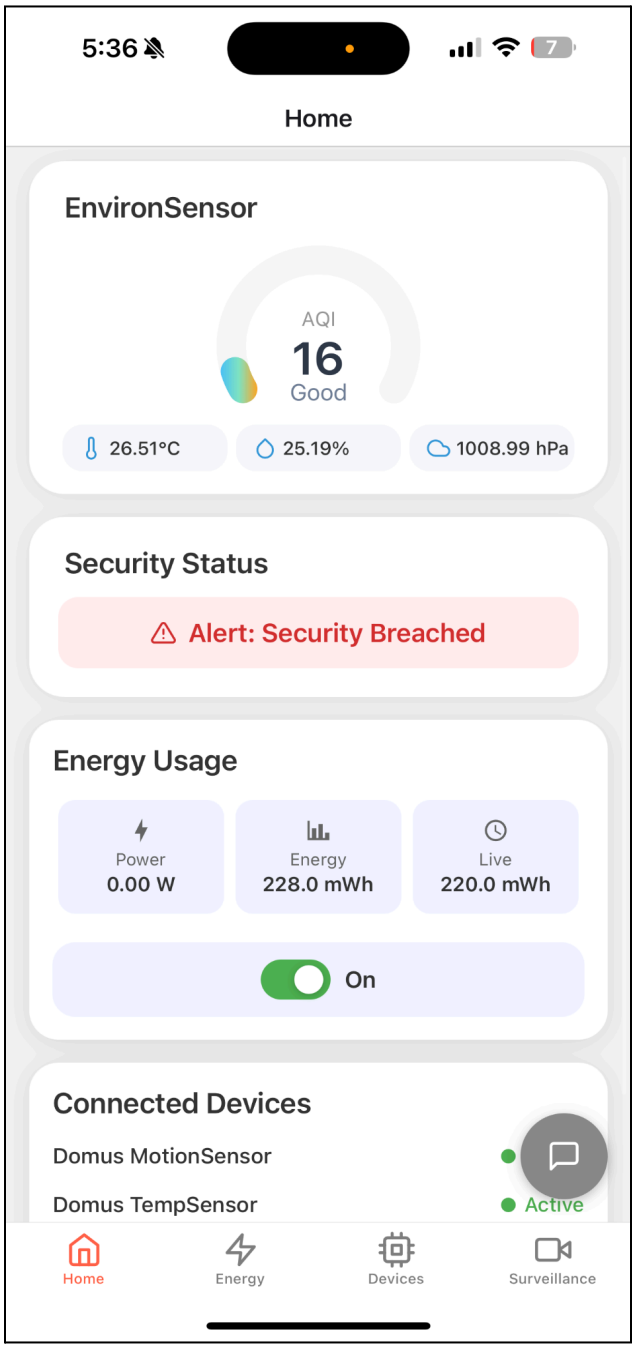
2.1 Overview block diagram



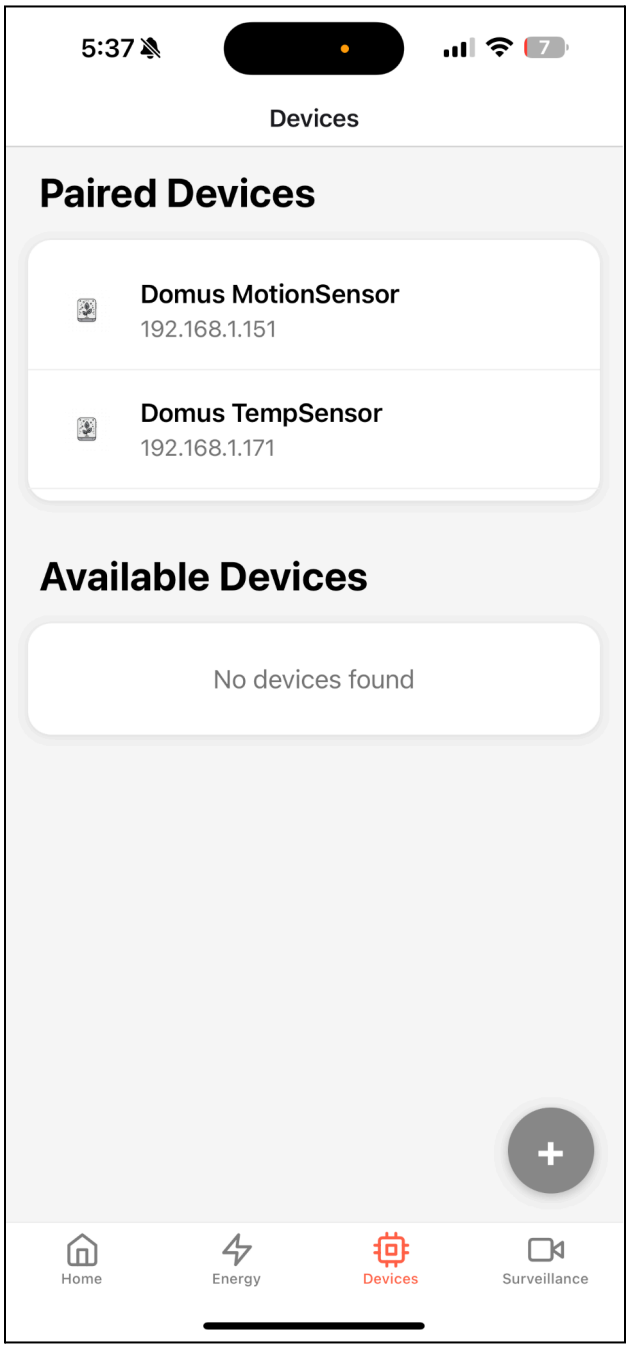
The diagram shows the Raspberry Pi 5 at the center, running a Flask server that exposes HTTP and WebSocket endpoints to the React Native mobile app. It collects environmental data from ESP32‑based Domus EnvironSensors and motion events from Domus MotionSensors over Wi‑Fi, streams video from the Raspberry Pi 4 AI camera module, and controls the Domus PowerPlug; all components communicate on the same local network for real‑time monitoring, control, and LLM‑powered queries.

2.2 User Interface Overview

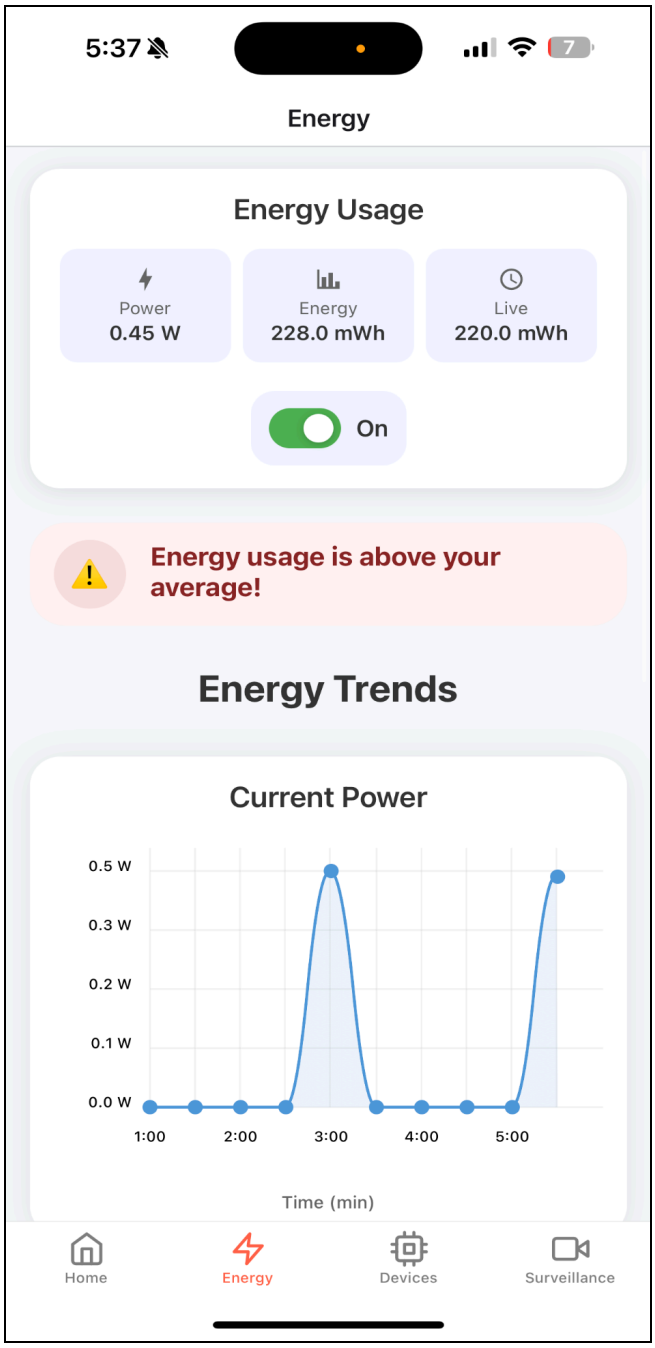
The iOS mobile app is built using **React Native** and serves as the control and display hub for the system. Once logged in via Google, the user can interact with multiple modules:

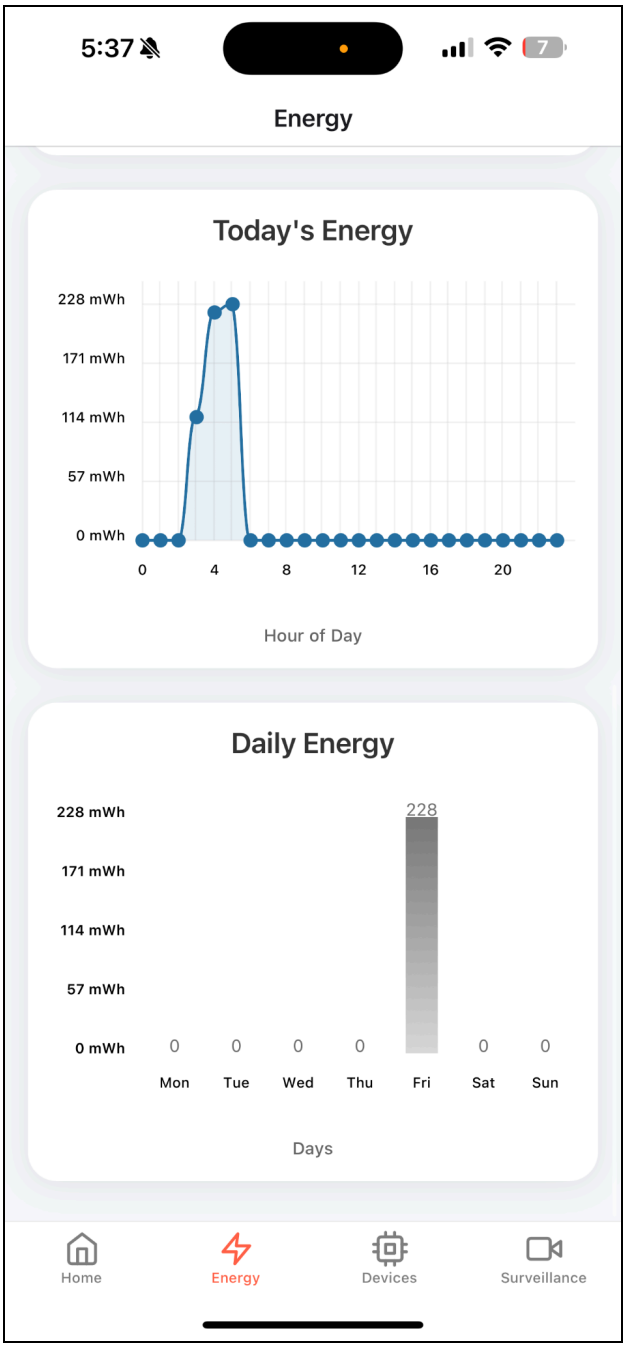


Home Screen
Displays live temperature, humidity, AQI, and surveillance status.

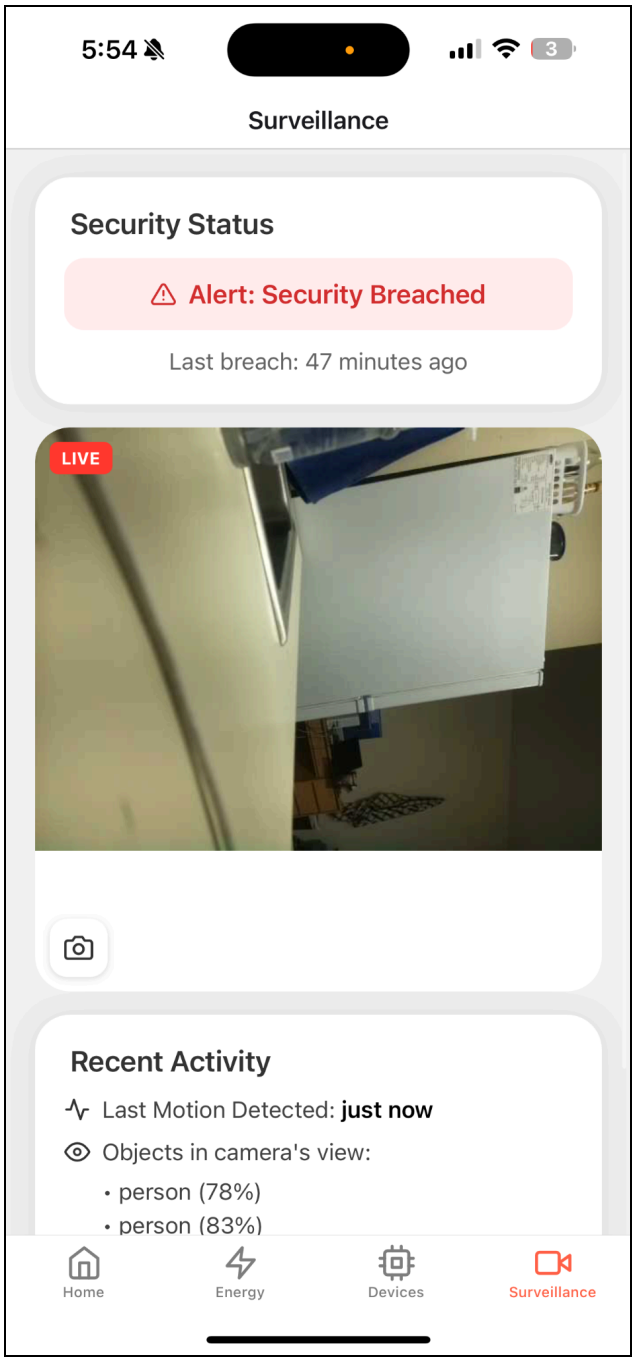


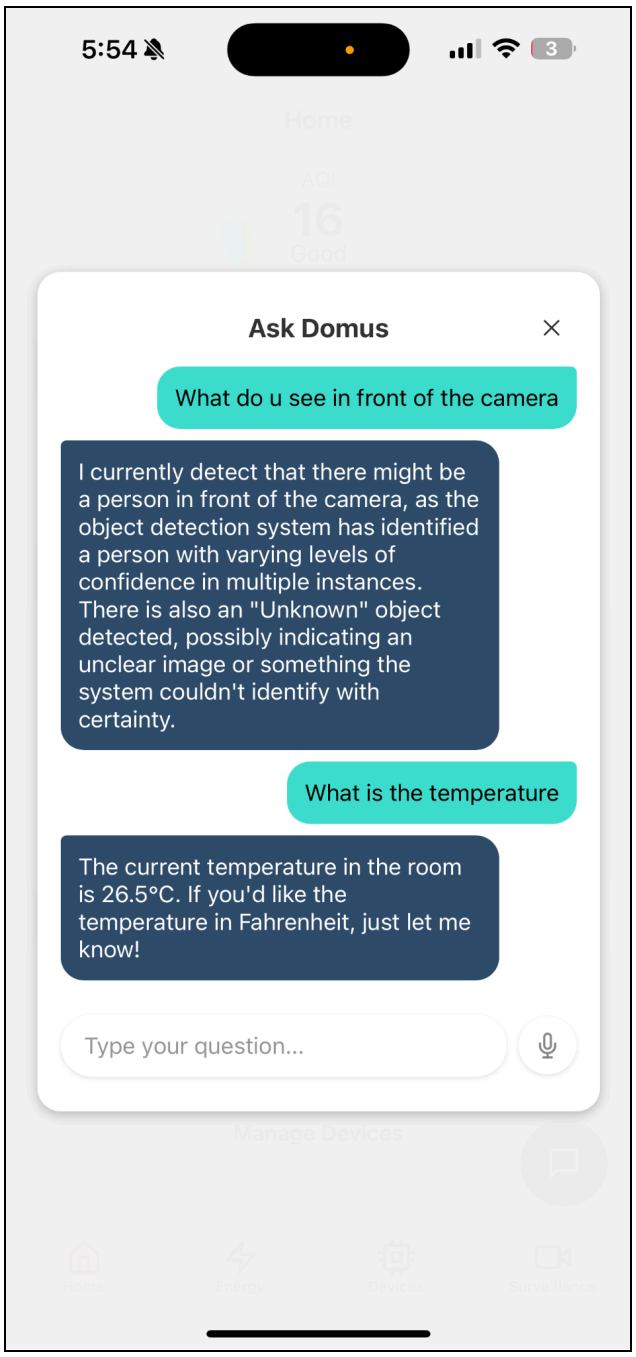
Devices Screen:
Allows user to view paired devices and pair with new devices





Energy Screen:
Allows user to toggle live state of plug to switch on/off devices and shows a live tracking page of energy and power tracking.
Alerts the user if daily energy usage is above longer term rolling average



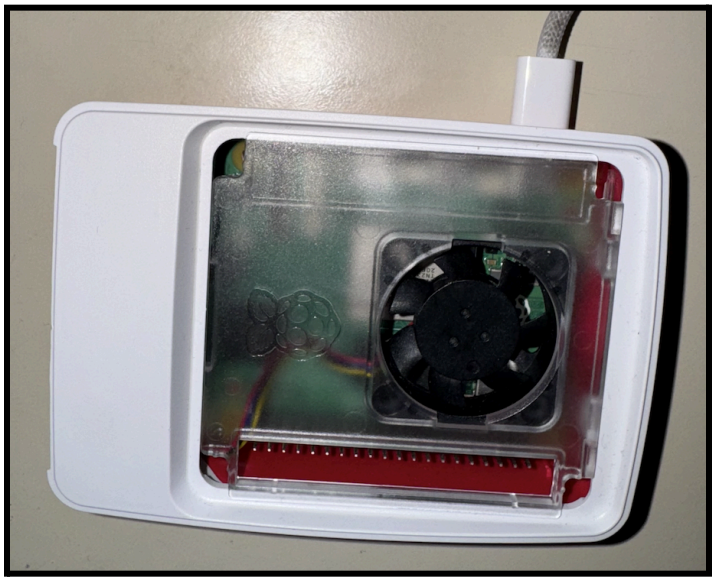


Surveillance Screen:
Alerts users of unknown faces, object detection results, and displays real-time footage

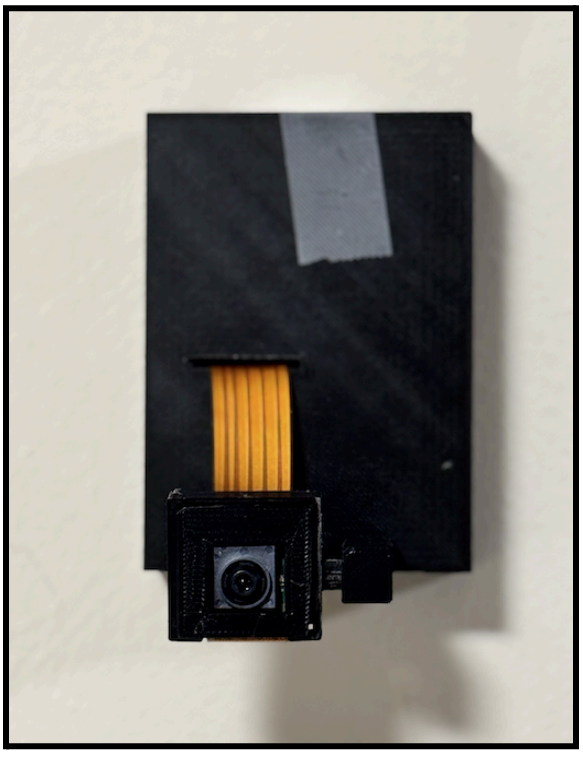
LLM Screen:
Users can query about the sensor data and surveillance via the in-app LLM. Communication can be through typing and querying or speech recognition.

2.3 Physical Description:

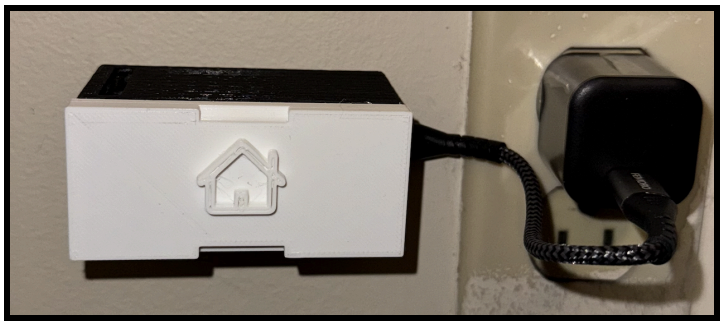
Hardware Overview



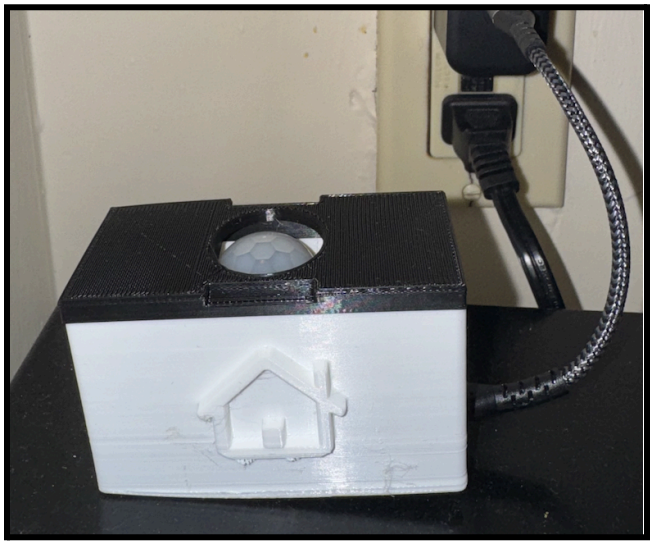
Domus Central Hub



Domus Vision Module



Domus EnvironSensor Module



Domus MotionSensor Module

Domus Power Plug Module

2.4 Installation, setup, and support

Hardware Setup

1. **Power On Devices**
    - RPi 5 and RPi 4 powered via power outlets
    - ESP32 devices connected to wall sockets
    - IMX500 camera connected to RPi 4 (MIPI interface)
    - Smart Plug powered via outlet and connected to Wi-Fi

2. **Placement**
    - Place housed sensors in desired areas of the room
    - Position camera near main door or hallway
    - Ensure all devices are within Wi-Fi range of the home router
    - Pair the device within the app's ecosystem

Software Setup

1. **React Native iOS App Setup**
    - Install app via TestFlight
    - Login via Gmail
    - Navigate to "Devices" screen to pair sensors
    - Confirm live data on Home and Energy screens

2. **Open Source Opportunity**
    - Users can download the containerized Domus environment to replicate and add to the existing framework to enable their specific hardware setup

3 Operation of the Project

Overview

In this mode, the Domus system operates under intended conditions to monitor and control various aspects of a smart home, including motion detection, environmental conditions, facial recognition, and energy consumption.

User Interaction and Steps

1. Initial Setup:

- Power on all hardware components:
    - Raspberry Pi 5 (central processor)
    - Raspberry Pi 4 (video processing unit)
    - ESP32 sensors (motion, BME688)
    - TP-Link Kasa Smart Plug
    - Raspberry Pi AI camera
- Connect devices to wall outlets as needed.

2. App Launch and Sensor Pairing:

- Open the Domus iOS app.
- Log in using Gmail authentication.
- Navigate to the **Devices** tab
- Add devices by following the on-screen instructions
- Return to the **Home** tab to confirm real-time data display.

3. Feature Navigation in the App:

Home Screen

- Displays real-time status and environmental data from all connected edge devices.
- Summarizes key metrics such as motion detection, temperature, humidity, and air quality.
- Provides an at-a-glance overview of the smart home environment.

Energy Screen

- Presents real-time data from TP-Link KP115 smart plugs.
- Visualizes power usage and total energy consumption via interactive time-series graphs.
- Enables remote control of connected appliances through ON/OFF toggles directly within the app interface.

Devices Screen

- Allows users to discover, add, and pair new ESP32-based sensors within the local network.
- Displays a list of available devices with status indicators and device-specific icons.
- Provides a streamlined pairing process to ensure seamless integration into the system.

Surveillance Screen

- Streams live footage from the Raspberry Pi AI camera module.
- Uses face recognition to detect and alert the user of unauthorized or unknown individuals.
- Implements object detection to identify and label visible objects within the camera frame in real time, with confidence scores.

4. Functional Interactions:

- **Motion Detection**: Walk past sensor. App updates movement status.
- **Face Recognition**: System sends real-time alerts if an unauthorized person is detected.
- **Object Detection**: Displays label and confidence score of identified objects.
- **BME688 Environmental Monitoring**:
    - Temperature (°C)
    - Humidity (%)
    - Gas resistance (Ohms)
    - Air Quality Indicator
    - Pressure (hPa)

- **Energy Tracking**:
  - View current energy/power metrics
  - Control smart plug remotely

- **Sensor Management**:
  - Add/Pair devices

Normal Consequences of User Actions

User Action	System Response
Add device in app	Device added to UI and paired with Pi
Trigger motion sensor	Movement status updated instantly
Unauthorized person enters	Alert sent to app
Connect appliance to smart plug	Energy usage graph updates

Exiting Normal Mode / Stopping Operations

- Turn off Raspberry Pi 5 and Raspberry Pi 4 using sudo shutdown now.
- Disconnect power sources from all components.
- Log out of the Domus iOS app.
- Remove the smart plug from the outlet if needed.
- Stop the Python Flask server manually or by stopping the terminal session.

Operating Mode 2: Abnormal Operations

Abnormal Mode	Detection Method	Recovery Method	User Action
Device not pairing	No response on app	Restart ESP32, verify WiFi	Try pairing again
App crashes	iOS crash dialog	Restart app, report bug	Contact support
Sensor offline	No data shown in app	Check power + network	Reconnect sensor
Surveillance error	No detection	Restart CV services	Restart Raspberry Pi 4

Getting Help

- Access troubleshooting documentation inside the Domus app (under Settings → Help).
- GitHub README includes common errors and commands.
- Contact team via:
    - Anirudh Singh: ansingh@bu.edu - Head of Customer Relations

Safety Issues

User and Bystander Safety

- Ensure sensors and Raspberry Pis are placed out of walkways to prevent tripping hazards.
- Do **not** expose Raspberry Pi or ESP32 boards to moisture or overheating environments.
- Ensure proper insulation of power cables.

Property Safety

- Smart plug must only be used with appliances below its rated load.
- BME688 and other sensors should be placed away from stoves, water sources, or high electromagnetic fields.

Data Safety

- Face recognition data is **processed locally**; no cloud storage is used.
- WiFi credentials are transmitted only via the secured React Native app and stored securely on-device.
- Device pairing requires manual initiation from the user.

System Dependency

Domus relies on:

- WiFi availability
- Raspberry Pi 5 functions as the central node. If the Pi fails, the entire system halts. The app will then display the most recent data.

4 Technical Background

1. System Architecture and Communication Model

The **Raspberry Pi 5** functions as the central processing hub. It receives data from peripheral microcontrollers (ESP32 units), processes visual data from a dedicated **Raspberry Pi 4 with an AI-enabled camera**, and exposes processed data to a React Native mobile application through HTTP endpoints and WebSocket streams.

Key components include:

- **ESP32 Microcontrollers**: These serve as edge sensors, responsible for motion detection (via HC-SR501) and environmental sensing (via BME688). They transmit sensor data wirelessly over WiFi using HTTP requests or JSON-based payloads.
- **Raspberry Pi AI Camera**: Processes raw video streams for facial and object recognition. The use of MobileNet SSD V2 and dlib enables lightweight, real-time inference.
- **Flask Server**: Hosts API endpoints on the Raspberry Pi 5 to relay sensor data and alerts to the mobile app. It also manages device registration, status monitoring, and plug control via the Kasa API.

Underlying Principle: The system relies on edge computing, which brings computation close to the data source(s), reducing latency and preserving bandwidth. This is essential for real-time systems and enhances data security by limiting cloud dependencies.

2. Environmental Sensing and Physics Principles

Temperature and Humidity Measurement
The **BME688 sensor** measures temperature using a bandgap temperature sensor, which leverages the temperature-dependent voltage of a silicon junction. Humidity is measured via a capacitive sensing element that changes its dielectric constant based on the moisture level in the air.

Air Quality Sensing
Gas resistance, a proxy for air quality, is derived from a heated metal oxide sensor. When volatile organic compounds (VOCs) are present in the air, the sensor's resistance decreases due to chemical reactions on the oxide surface. These readings are processed to produce an AQI (Air Quality Index) with a color-coded gauge for user readability.

Pressure Measurement
Barometric pressure is calculated from changes in membrane deflection caused by atmospheric pressure. This is relevant for altitude estimation and weather prediction.

3. Motion Detection and Energy Management

Motion Detection
We use **HC-SR501 Passive Infrared (PIR) Sensors**, which detect changes in infrared radiation. All warm-blooded bodies emit infrared; when a person moves across the sensor's field of view, the IR pattern shifts, triggering the detection circuit. The detection range is approximately 7 meters, with a field of view of ~120°. It is used to trigger the camera.

Smart Plug Integration and Energy Tracking
The **TP-Link KP115 Smart Plug** communicates via the Kasa API and allows Domus to:

- Track energy consumption and current draw
- Display this data in time-series graphs
- Remotely control the power state of connected appliances

Internally, the KP115 uses current and voltage sensors with sampling circuits (typically Hall-effect sensors and RMS calculators) to estimate wattage in real time.

4. Vision-Based Surveillance and Security

Face Recognition
Utilizing the dlib library in conjunction with **OpenCV**, Domus extracts 128-dimensional facial embeddings from a live camera feed. These embeddings are compared to known user profiles using Euclidean distance to determine identity. The system runs these computations on the Raspberry Pi 4, reducing strain on the central unit.

Object Detection
We implement **MobileNet SSD V2**, a lightweight convolutional neural network optimized for embedded hardware. This model detects and classifies objects in real time with a confidence score threshold. Visual bounding boxes and labels are streamed to the app to inform the user.

Live Streaming
Real-time video from the Raspberry Pi camera is encoded using an **MJPEG pipeline**, which transmits compressed frames over WiFi to the central processor and then to the mobile client.

5. Mobile Application Integration

The React Native mobile app acts as a visualization layer and control interface. It connects to the central server using:

- **WebSocket** for real-time alerts (e.g., motion or unauthorized face detected)
- **HTTP** for polling historical energy and environmental data
- **WiFi Credential Transfer** to ESP32 devices for seamless pairing

The app ensures users can monitor and control all devices regardless of location, assuming network connectivity is maintained.

6. Large Language Model Integration

The Domus system integrates a Large Language Model (LLM) to allow users to query sensor status and surveillance insights via natural language. The backend, hosted on the Raspberry Pi 5, uses the OpenAI Python API (`openai` library) to communicate with GPT-4. User queries from the React Native app are sent to a Flask endpoint over WebSocket or HTTP. The server composes a prompt by embedding real-time data (e.g., AQI, motion status, power usage) into a structured query, which is sent to the OpenAI API. The LLM response is then returned to the app and displayed in the assistant interface.

To ensure privacy, only structured prompts (no raw sensor logs or user identifiers) are transmitted externally. All sensor data is stored locally using SQLite, and credentials are handled on-device using React Native AsyncStorage.

Summary

Domus integrates principles from embedded systems, wireless networking, environmental science, and computer vision to provide a reliable smart home experience. Every component was chosen to balance performance, cost, and user accessibility. The system avoids cloud dependencies for privacy and leverages efficient edge computing to reduce response time and system load.

Whether monitoring a room's air quality, remotely toggling a device, or identifying an unknown visitor at the door, Domus operates as a cohesive, responsive system built on well-understood technical foundations.

5 Relevant Engineering Standards

Our senior design project, *Domus*, adheres to a variety of engineering standards spanning software development, wireless communication, electrical safety, and data protocols. Below are the most relevant categories:

1. Internet and Communication Protocols

The system uses widely adopted communication protocols:

- **HTTP/1.1** for RESTful sensor data exchange, adhering to W3C and IETF standards for interoperability and reliability.
- **WebSocket (RFC 6455)** for real-time theft detection alerts, enabling low-latency bidirectional communication between the Flask backend and mobile app.
- **mDNS (Multicast DNS, RFC 6762)** is used for hostname resolution of ESP32 and Raspberry Pi devices on the local network, eliminating the need for manual IP configuration.

2. Software Design and Coding Standards

All backend services are implemented in **Python**, following **PEP8** style guidelines for consistency and readability. The React Native mobile application uses a **component-based, modular architecture**, supporting scalability and separation of concerns. CI/CD pipelines were implemented for automated builds and testing. The frontend integrates GPT APIs, SQLite, and AsyncStorage using secure practices to ensure user data integrity.

3. Wireless and Electrical Standards

All communication between devices is conducted over **Wi-Fi (IEEE 802.11)**, ensuring compatibility with consumer networking equipment and compliance with FCC standards for home wireless devices. The ESP32 and other prebuilt modules used in the system comply with general **CE/FCC certifications**, as provided by their manufacturers. The **Kasa Smart Plug**, which interfaces with high-voltage AC power, conforms to **UL listing standards** and aligns with **National Electrical Code (NEC)** and **IEC 60364-1** guidelines regarding safe electrical design in residential settings.

4. Computer Vision and Machine Learning Standards

Object and face detection tasks are implemented using **MobileNet SSD** and **OpenCV-based face recognition**, aligning with conventional models in embedded AI. Camera streaming is implemented using **MJPEG over HTTP**, offering browser and React Native compatibility,

while also conforming to **MIME type standards (image/jpeg)** and typical streaming setups used in IoT applications.

5. Platform and Device Compatibility

The iOS mobile application is built using **Xcode** and conforms to **Apple Human Interface Guidelines**, ensuring usability and platform compliance. Integration with device hardware and OS-specific features (e.g., WebView rendering, voice permissions) follows standard React Native and iOS development practices.

6. Environmental and Use Standards

The system is designed exclusively for **indoor residential use**, except for the AI camera module, which can be installed in semi-outdoor covered locations. All exposed components (e.g., camera housings and adapters) are mounted in enclosures that reduce risk of environmental exposure and comply with general **IP20 safety classifications**.

7. Real-Time Data Integrity

Since Domus operates as a distributed system involving multiple ESP32 devices, a central Raspberry Pi, and a mobile app, **time synchronization** is critical for logging motion events, environmental readings, and alerts accurately. While initial synchronization is handled through internal clocks, the architecture supports future integration with the **Network Time Protocol (NTP)** to ensure consistent timestamping and data integrity across all modules.

8. Open Source Licensing and Legal Use

Domus integrates several open-source frameworks, including **OpenCV**, **MobileNet SSD**, **FFmpeg**, and various Python libraries. Each of these components is used in compliance with its respective licenses (e.g., **Apache 2.0**, **MIT License**, **LGPL**), ensuring legal and ethical integration into our system. GPT access and external APIs are used in accordance with the provider's terms of service.

6 Cost Breakdown

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	**Quantity**	**Description**	**Unit Cost ($)**	**Extended Cost ($)**
1	1	Raspberry Pi 5	92.99	92.99
2	1	Raspberry Pi 4B	61.29	61.29
3	2	ESP32	8.35	16.7
4	1	Raspberry Pi 5 Case	14.98	14.98
5	1	Raspberry Pi AI Camera	90.00	90.00
6	1	HC-SR501 Motion Sensors	4.75	4.75
7	1	BME688 Sensor	28.79	28.79
8	1	Kasa Smart Plug	19.54	19.54
9	2	USB-C Cable	4.25	8.49
10	2	USB-C Adapters	2.70	5.39
Beta Version-Total Cost				**342.9**

The total estimated cost for producing a single beta version of our smart home control system is **$342.90**, based on current market prices as of April 2025. This cost assumes no reused or donated components and reflects standard retail pricing from vendors like Adafruit, SparkFun, and Amazon.

The most significant expenses are the Raspberry Pi 5 ($92.99) and the AI Camera module ($90.00), which form the core of our system's central processing and computer vision functionality. The ESP32 boards, BME688 air quality sensor, and Kasa smart plug together enable wireless sensor data collection, environmental monitoring, and real-time energy tracking. We included both the Raspberry Pi 5 and Pi 4B in the beta version to support modular testing and distributed computation, as done in our alpha prototype. Additional accessories such as USB-C cables, adapters, and the Pi 5 case are necessary to ensure safe, robust, and portable deployment of the system.

This cost estimate provides a realistic baseline for small-scale manufacturing and helps us evaluate the commercial viability of our design moving forward.

7 Appendices

7.1 Appendix A - Specifications

Module	Description	Specification
Domus EnvironSensor	Monitors temperature, humidity, pressure, and air quality. Based on the BME688 sensor powered by ESP32.	±0.5°C temp, ±3% RH, ±1 hPa pressure, AQI derived from gas resistance
Domus MotionSensor	Detects motion and triggers security alerts. Uses HC-SR501 sensor and ESP32 node.	7-meter range, 120° FOV, ~1s response latency
Domus Vision Module	Detects faces and objects using an AI camera connected to Pi 4.	IMX500 sensor, MobileNet SSD V2, dlib face match, ~20 FPS
Domus Power Plug	Smart plug for energy tracking and control.	TP-Link KP115, ±5% accuracy, 15A/1800W rated load
Domus App Interface	iOS app for real-time monitoring and control. Built with React Native.	Displays live graphs, controls plug, shows alerts, Google login
Domus Central Hub	Raspberry Pi 5 hosting Flask server, APIs, and data storage.	Quad-core ARM Cortex-A76, 8GB RAM, SQLite storage, HTTP + WS APIs
System Communication	Network protocols used across all devices and sensors.	Wi-Fi IEEE 802.11, HTTP/1.1, WebSocket (RFC 6455), mDNS

7.2 Appendix B – Team Information

Akhil Bongu is a Computer Engineering major, with experience in training and deploying AI and Deep Learning models as an intern for NTT Data Business Solutions. He has worked with frameworks such as Pytorch and Tensorflow for a variety of use cases, from basic classification to computer vision.

Phone No: +1 (704)-605-7441
Email: akbongu@bu.edu

Anirudh Singh is a Computer Engineering major interested in pursuing AI and various other data driven solutions. Past experiences include Software Engineering intern at Cadence Design System under the Custom IC and Packaging group, with work spanning from API development to training and creating data pipelines for various machine learning models. Following his BU education, he will be joining Cadence Design Systems as a full-time Software Engineer in the System Design Analysis.

Phone No: +1 (857)-272-1560
Email: ansingh@bu.edu

Vansh Bhatia is a Electrical Engineer major with a minor in Manufacturing Engineering, passionate about advanced technological developments in the field of sustainable energy systems and power electronics. His experience in the Steel Industry at Frontier Alloy Steel Ltd, has made him well equipped in addressing real world manufacturing challenges.

Phone No: +1 (857)-318-7454
Email: vansh@bu.edu

Sreeram Andra is a dual degree candidate in Computer Engineering and Economics. He is passionate about Finance, Technology and the intersection of the two. His experience ranges in various fields of Fintech and Machine Learning; he is particularly passionate about the applications of Quantitative methods and Machine Learning in real-time market analysis.

Phone No.: +1 (440)-836-4899
Email: vssa8989@bu.edu