**Boston University**
**Electrical & Computer Engineering**
**EC463 Senior Design Project**

First Semester Report

# Domus

Submitted to

Prof. Alan Pisano
Boston University
8 St. Mary's Street, Boston, MA 02215
apisano@bu.edu

by

Smart Home Control (Team 27)
**(Domus)**

Team Members

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

**Submitted:** 12/08/2024

## Table of Contents

## Executive Summary

Smart Home Control and Security System
Team 27 – Domus

The growing dependence on cloud-based smart home systems has heightened concerns about data privacy and security vulnerabilities. This underscores the need for a solution that ensures real-time control while protecting user information. This project aims to develop a fully operational smart home automation system utilizing edge computing, enabling local data processing to enhance privacy, ensure seamless functionality, and provide secure, real-time control for users. The system will integrate edge-computing technology with IoT-enabled devices, allowing data to be processed on local hardware rather than external servers, supported by secure communication protocols and robust system optimization. This solution uniquely builds on to keep a check of energy consumption, complete control over smart appliances, surveillance as well as multi-device compatibility in a single platform, offering a one stop solution.

*Authored by Vansh Bhatia*

## 1.0 Introduction

Modern homeowners face a fragmented and cumbersome smart home experience due to the disparate nature of current solutions. Systems for security, energy management, and automation often operate independently, requiring multiple platforms or apps. Popular market options, like Google Nest or Amazon Alexa, primarily rely on cloud services, creating vulnerabilities in data privacy and increasing the risk of service disruptions. As these systems collect and store vast amounts of user data externally, users are increasingly skeptical about entrusting sensitive information to them. Furthermore, the product offers insights via the Domus application about energy consumption, air quality, temperature, humidity, as well as real-time streaming from surveillance cameras.

Smart home systems have grown in popularity due to the convenience and automation they offer. However, the heavy reliance on cloud-based operations often leads to challenges, including latency, service outages, and potential data breaches. Customers increasingly prioritize solutions that allow them to maintain control over their data and ensure secure, efficient operation. The industry currently lacks a comprehensive, unified solution that can handle automation, security, and energy management while providing robust privacy and real-time performance.

The purpose of this project is to design and develop an integrated smart home automation and security system leveraging edge computing. By processing data locally on user-owned devices rather than transmitting it to external servers, this system aims to provide a one-stop solution addressing automation, real-time control, energy management, and enhanced security. This approach not only addresses privacy concerns but also ensures seamless functionality without reliance on internet connectivity.

Our team’s approach involves creating a robust smart home platform centered on a Raspberry Pi-based central unit and supported by Coral AI boards. The system will utilize ESP32 Microcontrollers connected to local sensors for real-time data processing and control. Key features include:

1. **Integration of Edge Computing:** This reduces reliance on cloud servers, ensuring data remains private and response times are minimized.
2. **IoT-Enabled Devices:** The Camera, Smart Plugs and edge sensors communicate via secure protocols such as Wi-Fi, C++ code, WebSocket Protocol and Gstreamer Protocol files, respectively to ensure interoperability.
3. **Customizable Mobile Application:** A user-friendly interface allows for seamless management of devices, security alerts, and energy monitoring.
4. **Local LLM Integration:** Pre-trained context given by the user is processed locally, providing intuitive responses without sacrificing privacy.
5. **Real-Time Environmental Metrics:** Energy consumption, air quality, temperature and humidity tracking are augmented by predictive modeling to empower users with actionable insights.

Highlights and Special Features:

1. **Privacy-Centric Design:** By avoiding cloud-based data processing, the system minimizes privacy risks while providing offline functionality.
2. **Energy Optimization:** Advanced analytics provide actionable recommendations to reduce energy consumption and costs.
3. **Comprehensive Integration:** Combines security, automation, and energy management into one seamless platform.
4. **Enhanced Security:** Motion detection and theft alerts are processed locally for faster response times and greater reliability.
5. **Scalability:** Compatible with a wide range of protocols, the system supports up to 30 devices, making it suitable for homes of all sizes.

This project offers a pioneering solution for smart home automation, bridging the gap between user needs and current technological limitations.

**Figure 1:** The DOMUS logo draws its name from the Latin word "Domus," meaning "home"

*Authored by Vansh Bhatia*

## 2.0 Concept Development

### Understanding the Customer's Problem

The modern homeowner seeks an integrated solution to enhance home automation, ensure security, and minimize energy consumption. Current smart home systems, while popular, often fail to address critical customer concerns such as:

1. **Privacy**: Heavy reliance on cloud-based processing raises issues of data security and unauthorized access.
2. **Latency**: Cloud-dependent systems introduce delays in real-time operations, especially in remote or low-connectivity environments.
3. **Scalability and Modularity**: Existing solutions often lack the flexibility to expand or customize according to specific needs.
4. **Lack of streamlined solutions for Surveillance:** Current surveillance systems fail to provide reliable, real-time security solutions. Features like motion detection and facial recognition are often inaccurate or delayed, making them inadequate for proactive threat detection and response.
5. **Lack of Integrated Solutions:** Most smart home systems offer fragmented functionalities, requiring separate devices and apps for energy tracking, environmental monitoring, and security. This lack of a unified system complicates user experience and reduces overall efficiency.

Customers desire a "one-stop" smart home solution that integrates multiple functionalities while ensuring real-time feedback, modularity, and data security. A locally processed, edge-based system addresses these gaps by combining privacy, responsiveness, and adaptability.

### Conceptual Approach

To address the above challenges, the proposed **Domus Smart Home Control System** adopts an **edge-based architecture** with wireless sensor connectivity to a central hub. The system prioritizes privacy, reliability, and scalability by minimizing cloud dependence and leveraging local processing.

1. **Core Features**:
    - **Environmental Monitoring**: Integrate the BME688 sensor to provide data on temperature, humidity, and air quality index.
    - **Motion and Theft Detection**: Use HC-SR501 motion sensors and a Raspberry Pi HQ Camera Module to detect movement and unauthorized access, powered by facial recognition algorithms.
    - **Energy Tracking**: Track and analyze energy consumption using the TP-Link Smart Plug via the Kasa API, providing insights into usage patterns and enabling predictive billing.
    - **User Interaction**: Offer a seamless user experience through a React Native mobile application that enables monitoring and control of all functionalities.

2. **Engineering Requirements**:
    - **Wireless Sensor Integration**: Sensors such as the BME688 and HC-SR501 will communicate via ESP32 microcontrollers to ensure flexibility and modularity.
    - **Localized Processing**: The central Raspberry Pi 5 will handle processing tasks, including facial recognition and environmental data aggregation.
    - **Energy Efficiency**: Smart plugs will ensure minimal energy wastage while supporting API-based tracking for predictive analysis.
    - **Real-Time Feedback**: The system will respond to motion and theft detection alerts in <1 second latency.
3. **Innovative Features**:
    - **Edge-Based Architecture**: Ensures privacy by processing data locally on the Raspberry Pi 5, eliminating dependence on third-party cloud services.
    - **Modularity and Scalability**: The system can easily integrate additional sensors or new functionalities without major architectural changes.
    - **AI-Powered Theft Detection**: Uses machine learning for face recognition to alert homeowners of unauthorized access attempts.
    - **AI-Powered Object Detection/Activity Recognition:** Uses advanced machine learning to enable the system to identify objects and recognize activities in real-time, enhancing situational awareness
    - **Adaptive User Interface**: A user-friendly mobile app provides real-time system insights, energy analytics, and alerts.
    - **LLM Model:** Enables natural interactions, allowing users to query surveillance, energy, and environmental data for real-time insights, enhancing usability.

## Rationale for Chosen Approach

The chosen approach reflects a balance of privacy, performance, and user-centric design. By prioritizing edge computing and modularity, the system meets diverse customer needs while addressing limitations in existing solutions. Localized processing ensures both responsiveness and security, while the wireless architecture allows for scalability and ease of installation.

1. **Why Edge-Based Architecture?**
    - The privacy concerns associated with cloud-based systems are eliminated as all sensitive data (e.g., facial recognition) is processed locally.
    - Reduced latency ensures real-time operation, crucial for theft detection and motion-triggered responses.
2. **Why Wireless Integration?**
    - Wireless communication between sensors and the central hub ensures ease of installation, scalability, and mobility.
    - ESP32 microcontrollers enable low-power, high-efficiency data transmission.

## Alternative Solutions Considered

During the concept development phase, several alternative solutions were evaluated and subsequently rejected due to their limitations:

1. **Full Cloud-Based Design**:
    - **Reason for Rejection**: Privacy concerns were paramount, as cloud-based systems inherently expose user data to external servers. Additionally, reliance on continuous internet connectivity introduces potential points of failure, particularly in areas with unreliable networks.
    - **Key Learning**: While cloud-based systems excel in remote access, they fail to meet critical user expectations for data security and low-latency operation.
2. **Wired Sensor Connections**:
    - **Reason for Rejection**: Although wired connections offer stability and reliability, they limit system scalability and are impractical for dynamic home environments. Installation complexity also reduces user accessibility and increases deployment costs.
    - **Key Learning**: Wireless integration is essential for providing flexibility and scalability without compromising functionality.
3. **Hybrid Processing Architecture**:
    - **Reason for Rejection**: This approach attempted to balance local processing and cloud reliance but introduced unnecessary complexity and redundancy. Latency in cloud-dependent tasks undermined the system's responsiveness.
    - **Key Learning**: Edge-based processing is sufficient for the proposed functionalities, offering a simpler and more effective solution.

*Authored by Anirudh Singh*

## 3.0 System Description

- **Motion Detection**
  The HC-SR501 motion sensor detects movement within a 5-meter range and transmits data wirelessly to the Raspberry Pi 5 via ESP32 boards, triggering theft detection and activity recognition.
  *Key Metric*: Motion detection latency <1 second.
- **Theft Detection**
  A Raspberry Pi HQ Camera captures video upon motion detection. Facial recognition algorithms on the Raspberry Pi 5 identify unauthorized individuals, while a gyroscope enables dynamic camera movement based on motion detection.
  *Key Metrics*: Recognition latency <2 seconds; camera repositioning latency <1 second.
- **Activity and Object Recognition**
  AI models on the Coral USB Accelerator detect activities (e.g., walking, sitting) and objects (e.g., keys, phones). Detected activities and objects are integrated into the Domus Assistant for real-time user feedback.
  *Key Metrics*: Activity recognition accuracy ≥90%; object detection accuracy ≥85%.
- **Environmental Monitoring**
  The BME688 sensor monitors temperature, humidity, and air quality, transmitting updates every 5 seconds via ESP32 boards. Data is displayed on the mobile app in real time.
  *Key Metrics*: Temperature accuracy ±1°C; humidity accuracy ±2%.
- **Energy Monitoring**
  The system tracks energy usage via the TP-Link Kasa Smart Plug and provides predictive energy analysis, displaying trends on the app to help users manage consumption.
  *Key Metric*: Predictive billing accuracy ±10%.
- **Mobile Application**
  Built using React Native, the app serves as the central interface for monitoring and control. It displays live sensor data, theft alerts, and allows interaction with the Domus Assistant.
  *Key Deliverable*: Multi-platform (iOS/Android) app with an intuitive interface.
- **Domus Assistant (LLM Integration)**
  The AI-powered assistant provides natural language responses to queries like "What is the current air quality?" or "What objects are in view?". It retrieves logs and insights from all subsystems in real time.
  *Key Metric*: Query response time <1 second.

### Data Flow

All subsystems transmit data wirelessly to the Raspberry Pi 5 hub, where processing is performed locally to ensure privacy. Processed data is sent to the mobile app for user interaction. Figure 1 illustrates the system architecture and data flow.

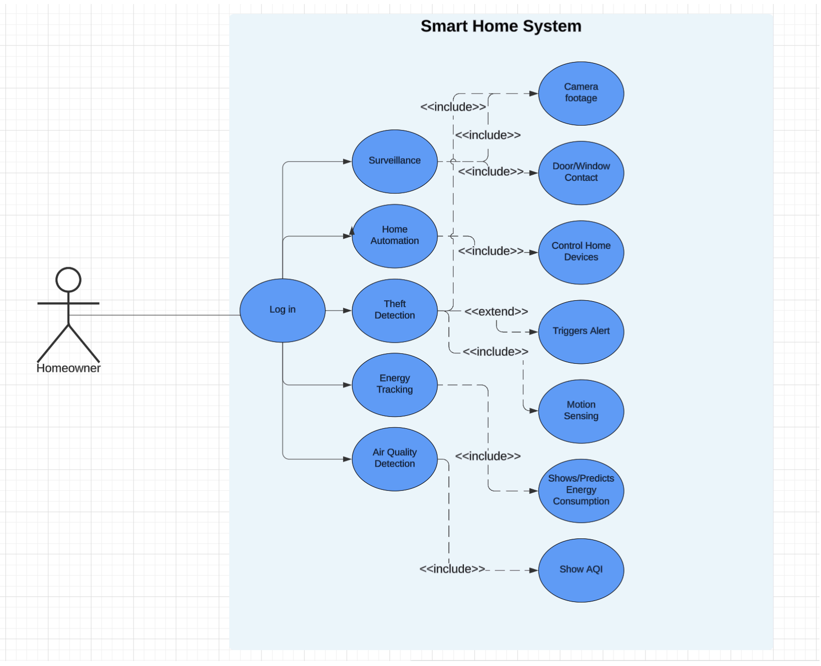
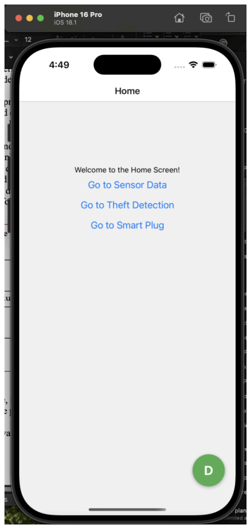


**Figure 2:** Workflow Diagram

**Conclusion:** The Domus Smart Home Control System offers a robust, real-time, and user-centric solution for home automation. By leveraging edge-based processing, wireless communication, and a user-friendly mobile application, the system ensures enhanced privacy, efficiency, and scalability. Its modular design allows for future expansion, making it a versatile and forward-looking solution for modern smart homes.

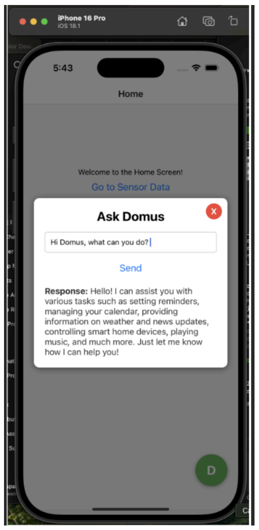
**Sample Images of preliminary Mobile Application User Interface**

Home Screen:



Welcome to the Home Screen!
Go to Sensor Data
Go to Theft Detection
Go to Smart Plug

Ask Domus:



Welcome to the Home Screen!
Go to Sensor Data

Ask Domus

Hi Domus, what can you do?

**Response:** Hello! I can assist you with various tasks such as setting reminders, managing your calendar, providing information on weather and news updates, controlling smart home devices, playing music, and much more. Just let me know how I can help you!

Splash Screen:

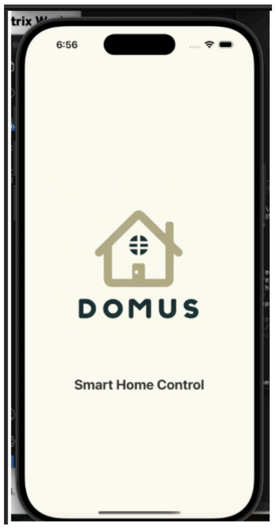


**Figure 3:** Images of Application

**Pseudocode for the system:**

```
# Main system initialization
initialize_hardware()       # Set up sensors, camera, ESP32s, Raspberry Pi, and Coral USB Accelerator
initialize_software()       # Start Flask server, LLM, and React Native app backend
connect_to_app()            # Establish communication with mobile application

# Infinite loop for continuous monitoring and processing
while system_is_running():

    # Step 1: Motion Detection
    if motion_detected():
        send_alert_to_app("Motion detected")  # Notify the user via the mobile app
        activate_camera()                     # Start camera streaming

        # Step 2: Theft Detection
        if unauthorized_face_detected():
            pan_camera_based_on_motion()      # Adjust camera position dynamically
            send_alert_to_app("Unauthorized face detected!")

    # Step 3: Environmental Monitoring
    environmental_data = read_bme688_sensor()
    send_data_to_app("Environment", environmental_data)  # Temperature, humidity, AQI

    # Step 4: Energy Monitoring
    energy_usage = read_kasa_smart_plug()
    send_data_to_app("Energy Usage", energy_usage)
    if energy_threshold_exceeded():
        send_alert_to_app("High energy usage detected!")

    # Step 5: Activity and Object Recognition
    if camera_stream_active():
        detected_objects = perform_object_detection(camera_stream)
        detected_activities = perform_activity_recognition(camera_stream)
        send_data_to_app("Objects", detected_objects)
        send_data_to_app("Activities", detected_activities)

    # Step 6: LLM Interaction
    user_query = get_user_query_from_app()
    if user_query:
        response = process_llm_query(user_query, system_logs, real_time_data)
        send_response_to_app(response)

    # Step 7: Climate Control (e.g., Smart Comfort Control)
    if temperature_below_threshold():
        activate_heating_system()
        send_alert_to_app("Heating system activated")

    # Step 8: Data Logging
    log_system_data(environmental_data, energy_usage, detected_objects, detected_activities)

# System shutdown
shutdown_hardware()
shutdown_software()
```

*Authored by Anirudh Singh*

## 4.0 First Semester Progress

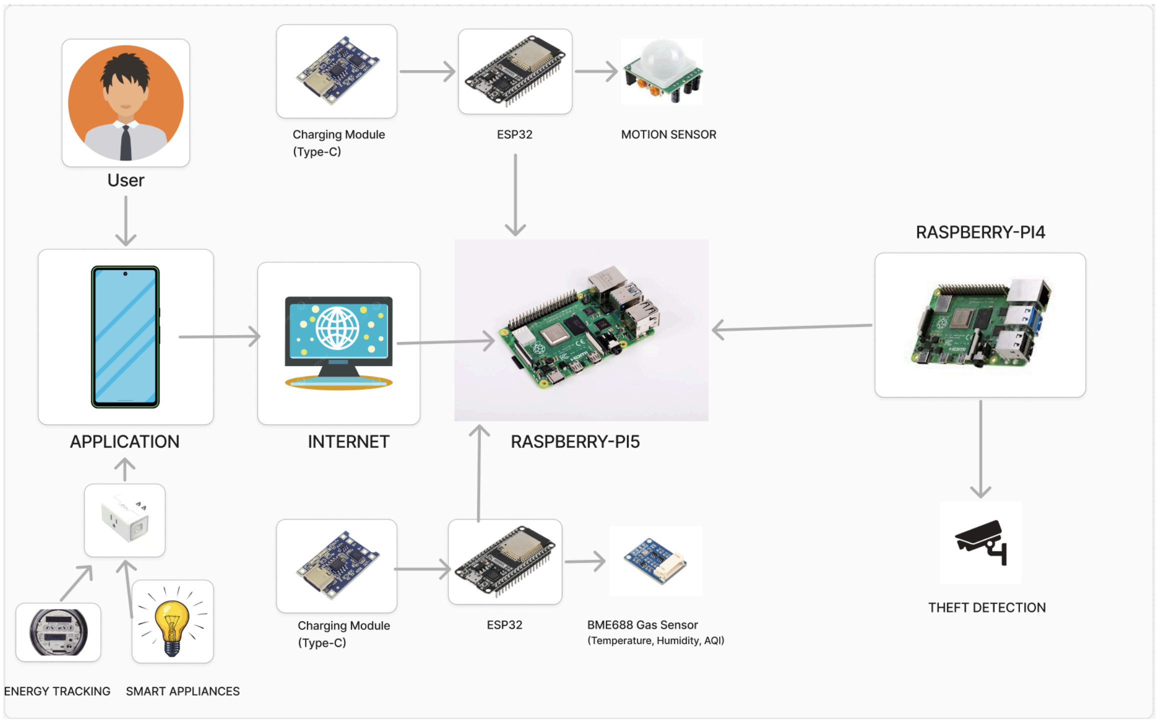


**Figure 4:** High-level Diagram of Prototype

At the end of the Fall 2024 semester, we constructed a functional prototype that integrated several parts of our final project objective. Listed below is the progress that we have made with our project over the Fall 2024 semester:

1. **Motion Detection**
   - The HC-SR501 motion sensor was calibrated to detect movement within a 5-meter range.
   - Data is transmitted wirelessly to the Raspberry Pi 5 via ESP32 boards, triggering the theft detection system.
   - **Key Result**: Initial testing confirmed a detection latency of <1 second, with a 98% success rate in identifying motion during controlled experiments.
2. **Facial Recognition**
   - A facial recognition model was developed and trained using OpenCV and the face_recognition library from dlib.
   - The model was integrated with the camera subsystem to differentiate authorized and unauthorized faces, sending alerts to the mobile app upon detection.
   - **Key Result**: The system achieved 92% accuracy in identifying faces under ideal lighting conditions, with real-time processing latency of <2 seconds.

3. **Environmental Monitoring**
    - The BME688 sensor was configured to provide real-time readings of temperature, humidity, and air quality.
    - Data updates are sent to the Raspberry Pi 5 every 5 seconds via ESP32 boards and displayed on the mobile app.
    - **Key Result**: Sensor accuracy was verified, achieving temperature accuracy of ±1°C and humidity accuracy of ±2% during controlled testing.
4. **Energy Monitoring**
    - The TP-Link Kasa Smart Plug was calibrated to track energy usage and display consumption data on the mobile app.
    - Predictive energy analysis functionality is under development, with initial integration of usage logs into the system.
    - **Key Result**: Energy consumption readings were accurate.
5. **Activity Recognition (In Progress)**
    - Preliminary work was conducted on integrating an AI-based activity recognition model using the Coral USB Accelerator.
    - The model is currently being trained to identify basic activities such as walking and sitting. Integration with the Domus Assistant and the mobile app is ongoing.
    - **Key Result**: Approximately 10% of the activity recognition system has been implemented, with initial model tests yielding an accuracy of 85% on benchmark datasets.
6. **Mobile Application Features**
    - A preliminary React Native app was developed to serve as the central dashboard for the system.
    - Features include real-time data visualization for environmental monitoring, theft alerts, and energy usage tracking.
    - **Key Result**: The app successfully displayed real-time data with <2 seconds latency and allowed seamless navigation between core functionalities.
7. **SQL Database Integration**
    - A SQLite database was integrated into the system for storing sensor logs and event data locally on the Raspberry Pi 5.
    - This database enables the system to maintain historical data, supporting features like predictive analytics and detailed user reports.
    - **Key Result**: The database successfully logged sensor data for up to 24 hours during initial tests, with retrieval times of <1 second for queries.
8. **LLM Integration**
    - A foundational version of the Domus Assistant was implemented using GPT-3.5 Turbo.
    - The LLM supports basic natural language queries.
    - **Key Result**: User queries were answered within <1 second during initial tests, with 90% accuracy for basic questions.

| FEATURE | METRIC | OUTCOME |
|---|---|---|
| **Motion Detection** | Detection Accuracy | 100% |
| **Theft Detection** | Facial recognition latency | <1 second |
| **Environmental Sensing** | Temperature Accuracy | ±1°C |
| | Humidity Accuracy | ±2% |
| **Energy Tracking** | Toggle latency for devices | <1 second |
| | Real-time power consumption display | Accurate |

**Table 1:** Prototype Metrics

**Motion Detection**:

- Achieved a **100% detection accuracy**, ensuring reliable detection of movement within the prototype's range (0 - 5 meters)

**Theft Detection:**

- Optimized the facial recognition system with a **latency of less than 1 second**, enabling swift identification and alert generation.

**Environmental Sensing**:

- **Temperature Accuracy**: Achieved a precision of **±1°C**, ensuring dependable temperature monitoring.
- **Humidity Accuracy**: Maintained an accuracy of **±2%**, providing reliable environmental data.

Both temperature and humidity readings were cross referenced with a indoor hygrometer

**Energy Tracking:**

- **Toggle Latency**: Enabled device toggling with a **latency of less than 1 second**, supporting seamless control.
- **Power Consumption Display**: Provided **accurate real-time power consumption readings**, facilitating effective energy monitoring.

*Authored by Akhil Bongu*

## 5.0 Technical Plan

The **Technical Plan** outlines the tasks and milestones to be achieved in the upcoming semester, focusing on the completion of the Domus Smart Home Control System. Each task is clearly described with assigned responsibilities, measurable deliverables, and deadlines. The tasks aim to integrate advanced features such as object detection, activity recognition, predictive energy analysis, and enhanced LLM capabilities, culminating in a fully functional system.

**Task 1: Object Detection Setup with MobileNet SSD (in progress)**

- **Description**: The MobileNet SSD object detection model shall be deployed using the Coral USB Accelerator. A prebuilt Docker container containing all necessary dependencies, such as TensorFlow Lite and Edge TPU runtime, will be identified and utilized to ensure compatibility. The object detection system will be configured to recognize common household objects such as keys, phones, and bottles.
- **Steps**:
    1. Prebuilt Docker images will be researched, tested, and validated for compatibility.
    2. An object detection pipeline will be set up using the Coral USB Accelerator on the Raspberry Pi.
    3. Detection accuracy across 10 object categories under varying lighting conditions will be tested.
- **Metrics**: The system shall achieve a detection accuracy of at least 85% for household objects with an inference latency of <50 milliseconds.
- **Lead**: Venkata Sai Andra
- **Assisting**: Akhil Bongu, Anirudh Singh, Vansh Bhatia
- **Deliverable**: A fully configured object detection system integrated with the Domus Assistant for responding to user queries about detected objects.
- **Deadline**: February 15, 2025.

**Task 2: Activity Recognition Integration**

- **Description**: An activity recognition system using an open-source model such as MoveNet or OpenPose will be implemented. The Coral USB Accelerator will be used to optimize inference performance. Activities like walking, sitting, and waving will be recognized, enabling interaction tracking.
- **Steps**:
    1. Suitable open-source models for activity recognition will be researched and selected.
    2. The chosen model will be trained and tested on benchmark datasets such as MS COCO or HMDB51.
    3. The model will be deployed on the Coral USB Accelerator and fine-tuned for real-time performance.

- **Metrics**: The system shall recognize five predefined activities with an accuracy of 90% in test scenarios, maintaining inference latency <100 milliseconds.
- **Lead**: Venkata Sai Andra
- **Assisting**: Anirudh Singh, Akhil Bongu, Vansh Bhatia
- **Deliverable**: A functional activity recognition system integrated with the Domus Assistant and user alerts.
- **Deadline**: March 1, 2025.

**Task 3: Motion-Triggered Camera Streaming**

- **Description**: The motion sensors will be integrated with the camera to trigger live streaming upon detecting movement. The WebSocket protocol will be employed to ensure real-time communication and low latency.
- **Steps**:
    1. The motion sensors (e.g., HC-SR501) will be configured to send alerts to the Raspberry Pi.
    2. The camera will be programmed to begin streaming upon receiving motion sensor alerts.
    3. The system's responsiveness will be tested in various room layouts.
- **Metrics**: The latency between motion detection and camera activation shall remain <1 second.
- **Lead**: Akhil Bongu
- **Assisting**: Venkata Sai Andra, Anirudh Singh, Vansh Bhatia
- **Deliverable**: A motion-triggered live streaming system with real-time alerts.
- **Deadline**: February 20, 2025.

**Task 4: Theft Detection System**

- **Description**: The theft detection functionality shall be enhanced by integrating facial recognition with dynamic camera movement. A gyroscope will be employed to enable the camera to reposition itself based on motion sensor feedback.
- **Steps**:
    1. A gyroscope-controlled pan/tilt mechanism will be implemented to adjust the camera.
    2. The facial recognition model will be optimized to handle multiple faces and low-light conditions.
    3. Stress tests will be conducted by simulating unauthorized entry scenarios.
- **Metrics**: The system shall detect and identify faces within 2 seconds, with camera repositioning completed in <1 second.
- **Lead**: Akhil Bongu
- **Assisting**: Venkata Sai Andra, Anirudh Singh, Vansh Bhatia
- **Deliverable**: A theft detection system combining facial recognition and dynamic camera movement.
- **Deadline**: March 15, 2025.

**Task 5: User Facial Data Training**

- **Description**: A feature will be developed to allow users to train the system with their own facial data via the mobile app. A secure mechanism for data storage and real-time updates to the facial recognition database will be implemented.
- **Steps**:
    1. An intuitive user interface for uploading and managing facial data will be created.
    2. The system will be programmed to dynamically update the facial recognition database with new inputs.
    3. Data encryption will be implemented to ensure secure storage and transmission.
- **Metrics**: Users shall successfully train the system with new facial data within 30 minutes, with zero data breaches during tests.
- **Lead**: Vansh Bhatia
- **Assisting**: Anirudh Singh, Akhil Bongu, Venkata Sai Andra
- **Deliverable**: A secure and user-friendly facial data training feature in the app.
- **Deadline**: February 28, 2025.

**Task 6: Surveillance-LLM Integration**

- **Description**: Surveillance features, including object detection, activity recognition, and theft detection, will be integrated with the LLM. Users will query the Domus Assistant for real-time logs and insights on detected events.
- **Steps**:
    1. APIs to fetch logs from the surveillance systems will be developed.
    2. Prompts for the LLM will be designed to generate concise and accurate responses.
    3. The system will be tested under high workload scenarios to ensure performance.
- **Metrics**: The system shall respond to user queries within 1 second, achieving a >95% accuracy rate in summarizing events.
- **Lead**: Anirudh Singh
- **Assisting**: Venkata Sai Andra, Akhil Bongu, Vansh Bhatia
- **Deliverable**: An integrated LLM system providing real-time surveillance logs.
- **Deadline**: March 30, 2025.

**Task 7: Voice Feedback for LLM**

- **Description**: Voice feedback capabilities will be implemented for the LLM using lightweight text-to-speech (TTS) models such as Google TTS or eSpeak. Users will interact with the Domus Assistant through voice commands and receive verbal responses.
- **Steps**:
    1. Suitable TTS models for Raspberry Pi compatibility will be identified and tested.
    2. The TTS system will be integrated with the Domus Assistant.

3. User testing will be conducted to validate speech clarity and latency.

- **Metrics**: Voice responses shall be generated within 2 seconds, with >90% intelligibility in user tests.
- **Lead**: Anirudh Singh
- **Assisting**: Venkata Sai Andra, Akhil Bongu, Vansh Bhatia
- **Deliverable**: A voice-enabled Domus Assistant.
- **Deadline**: April 15, 2025.

**Task 8: Final Housing Units**

- **Description**: Ergonomic and aesthetically pleasing housing units for all hardware components will be designed and produced using SolidWorks. Focus will be placed on branding, durability, and usability.
- **Steps**:
    1. CAD models for housing units will be designed and iterated upon.
    2. Components will be tested for durability through drop tests.
    3. Final designs will be evaluated for aesthetics and ergonomic functionality.
- **Metrics**: Housing units shall withstand drop tests from 1 meter and pass usability evaluations.
- **Lead**: Vansh Bhatia
- **Assisting**:Venkata Sai Andra, Anirudh Singh, Akhil Bongu
- **Deliverable**: Finalized, branded housing units.
- **Deadline**: March 20, 2025.

**Task 9: Finalizing Power Sources**

- **Description**: Evaluate and select batteries and power delivery methods prioritizing safety ratings, efficiency, and compatibility.
- **Steps**:
    1. Research and test batteries for runtime, thermal stability (≤10°C rise), load capacity (~485.86 Wh/day), and portability
    2. Assess USB-C PD for Raspberry Pi 5 and centralized charging for sensors.
    3. Finalize the safest and most efficient option based on testing.
- **Metrics**: ≥24-hour runtime, Safety certifications, stable operation under load.
- **Lead**: All
- **Assisting**: All
- **Deliverable**: **Deliverable**: Final power source selection and circuitry
- **Deadline**: February 20, 2025.

**Task 9: Predictive Energy Analysis**

- **Description**: A predictive energy analysis model shall be developed to estimate electricity usage and billing based on historical data from KP115 plugs. Results will be displayed on the app's energy monitoring screen.
- **Steps**:
    1. Energy usage data over 30 days will be gathered and preprocessed.
    2. A regression-based predictive model will be built and tested.

3. Predictions will be integrated into the app for user access.

- **Metrics**: Predictions shall achieve an accuracy of ±5% compared to actual usage in test cases.
- **Lead**: Venkata Sai Andra
- **Assisting**: Akhil Bongu, Anirudh Singh, Vansh Bhatia
- **Deliverable**: Predictive energy analysis integrated into the app.
- **Deadline**: March 25, 2025.

**Task 10: Automated Climate Control (Tentative)**

- **Description**: The "Smart Comfort Control" feature shall be implemented to activate heating devices when the temperature falls below a user-defined threshold.
- **Steps**:
    1. A threshold-setting interface will be developed in the app.
    2. The system will be programmed to activate heating devices upon threshold breach.
    3. Performance will be validated under varying environmental conditions.
- **Metrics**: Heating activation shall occur within 5 seconds of a threshold breach with 100% reliability.
- **Lead**: Anirudh Singh
- **Assisting**: Venkata Sai Andra, Akhil Bongu, Vansh Bhatia
- **Deliverable**: A fully functional Smart Comfort Control system.
- **Deadline**: March 10, 2025.

**Task 11: Full System Testing**

- **Description**: Comprehensive functional and stress testing shall be conducted for all components, simulating real-world conditions. Results will be documented, and any failures will be addressed iteratively.
- **Steps**:
    1. Comprehensive test cases will be developed for each feature.
    2. Iterative tests will be conducted to ensure stability and reliability.
    3. Results will be documented and used to refine system performance.
- **Metrics**: All features shall achieve ≥95% reliability in end-to-end testing.
- **Lead**: All
- **Assisting**: All
- **Deliverable**: A final test report documenting successful system functionality.
- **Deadline**: April 20, 2025.

*Authored by Venkata Sai Andra*

## 6.0 Budget Estimate

| Item | Description | Total Cost |
|---|---|---|
| 1 | Raspberry Pi 5 | $92.99 |
| 2 | DHT22 | $11.80 |
| 3 | IR Motion Sensor | $9.49 |
| 4 | MakerFocus Camera | $22.99 |
| 5 | BME688 Sensor | $28.79 |
| 6 | Raspberry PI case | $14.98 |
| 7 | ESP32 | $16.69 |
| 8 | Coral USB Accelerator | $73.99 |
| 9 | Voltage Booster | $12.99 |
| 10 | Charging Module | $9.99 |
| 12 | Smart Plugs | $10.00 |
| 13 | Raspberry Pi 4 | $61.29 |
| 14 | Raspberry Pi HQ Camera | $74.99 |
| **Total Spent = $440.98** | | |
| **Remaining = $309.02 (**Departmental Budget) | | |

**Table 2:** Budget

**Budget Narrative**

1. **Major Budget Items**:
   - The **Raspberry Pi 5** and **Coral USB Accelerator** represent the largest expenditures, accounting for 42% of the total budget. These components are vital for system processing and machine learning tasks, making their inclusion necessary.

2. **Sources of Components**:
   - All components are sourced from Amazon, a reliable vendor known for competitive pricing. This ensures consistency and availability of parts.
   - The project relies on standard off-the-shelf components to keep costs manageable and ensure easy replacements if necessary.

3. **Budget Constraints**:

   - With a total budget of $1,000 and a department allowance of $750, the project remains well within limits, leaving $560 unspent overall and $309 within the department allowance.
   - This surplus provides flexibility to address unforeseen expenses or include additional components if required.

*Authored by Akhil Bongu*

## Attachments

### 1.1 Appendix 1 – Engineering Requirements

Team #27 Team Name: Smart Home Control and Security System

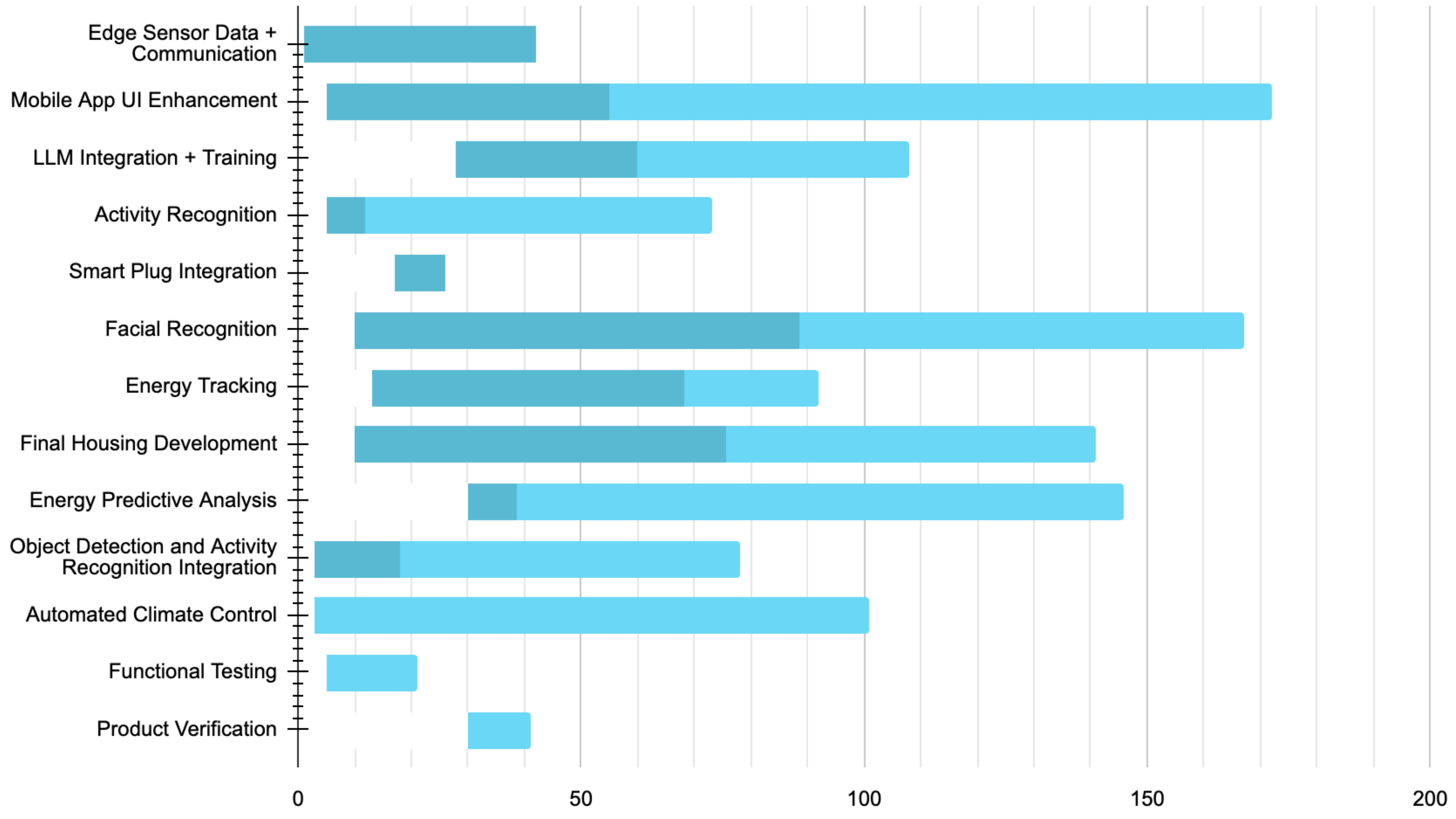
| Requirement | Value, Range, Tolerance, Units |
|---|---|
| Modular Sensor Integration | Wireless Communication using ESP32 microcontrollers, each capable of supporting connections of up to 5 sensors; communication range approx > 10 m |
| Data Processing and Privacy | Localized edge computing on Raspberry Pi 5, ensuring real-time processing with latency ≤ 100ms and no reliance on cloud services. |
| Motion Detection | Detection range: 0–5m with response time ≤ 1 second. |
| Theft Detection | Facial recognition accuracy ≥ 90%, response time ≤ 1 second for alerts. |
| Environmental Monitoring | Temperature: ±1°C, Humidity: ±2%, AQI accuracy ±2%. Data update frequency ≤ 5 seconds. |
| Energy Tracking | Real-time energy tracking with usage accuracy ±0.01%. Data updated every 2 seconds. |
| Integration with the Mobile Application | React Native app displaying sensor data, theft detection alerts, and energy monitoring with data refresh rate ≤ 2 seconds. |
| Power Efficiency | ESP32 boards powered via Type-C charging modules. Battery life ≥ 24 hours with power consumption ≤ 5W per module. |
| Scalability | Scalable architecture allows integration of at least two additional sensors or modules without significant reconfiguration. |
| System Performance | End-to-end system reliability ≥ 95%, including seamless communication between sensors, the Raspberry Pi 5, and the mobile application. |

**Table 3:** Engineering Requirements

## 1.2 Appendix 2 – Gantt Chart.

| TASK NAME | START DATE | DAY OF MONTH* | END DATE | DURATION* (WORK DAYS) | DAYS COMPLETE* | DAYS REMAINING* | PERCENT COMPLETE |
|---|---|---|---|---|---|---|---|
| Edge Sensor Data + Communication | 10/1 | 1 | 11/10 | 41.00 | 41.00 | 0.00 | 100% |
| Mobile App UI Enhancement | 10/5 | 5 | 3/20 | 167.00 | 50.10 | 116.90 | 30% |
| LLM Integration + Training | 11/28 | 28 | 2/15 | 80.00 | 32.00 | 48.00 | 40% |
| Activity Recognition | 12/5 | 5 | 2/10 | 68.00 | 6.80 | 61.20 | 10% |
| Smart Plug Integration | 11/17 | 17 | 11/25 | 9.00 | 9.00 | 0.00 | 100% |
| Facial Recognition | 10/10 | 10 | 3/15 | 157.00 | 78.50 | 78.50 | 50% |
| Energy Tracking | 11/13 | 13 | 1/30 | 79.00 | 55.30 | 23.70 | 70% |
| Final Housing Development | 11/10 | 10 | 3/20 | 131.00 | 65.50 | 65.50 | 50% |
| Energy Predictive Analysis | 11/30 | 30 | 3/25 | 116.00 | 8.80 | 107.20 | 10% |
| Object Detection and Activity Recognition Integration | 12/3 | 3 | 2/15 | 75.00 | 15.00 | 60.00 | 20% |
| Automated Climate Control | 12/3 | 3 | 3/10 | 98.00 | 0.00 | 98.00 | 10% |
| Functional Testing | 3/5 | 5 | 3/20 | 16.00 | 0.00 | 16.00 | 0% |
| Product Verification | 3/18 | 30 | 3/25 | 11.00 | 0.00 | 11.00 | 0% |

**Table 4:** Project Schedule



**Duration**

**Figure 5:** Gantt Chart

## 1.3 Appendix 3 – Other Appendices

Other typical attachments that are added to bolster the competitiveness of your proposal:

- Technical references (in proper bibliographic form) including key URLs.
- Your drawings and schematics (rather than embedding in text)

Hand-drawn Schematic of the wireless circuit for edge sensors

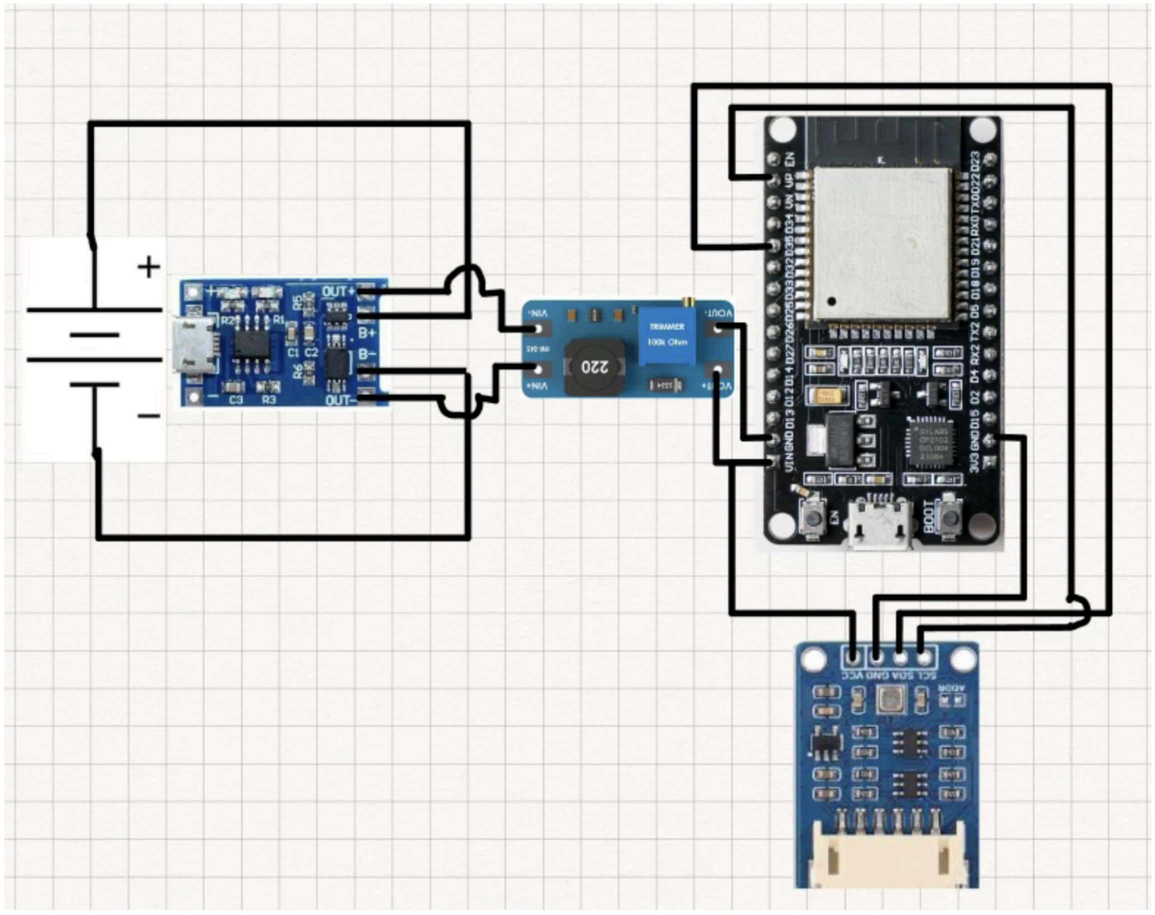


**Figure 6:** Hand-drawn schematic

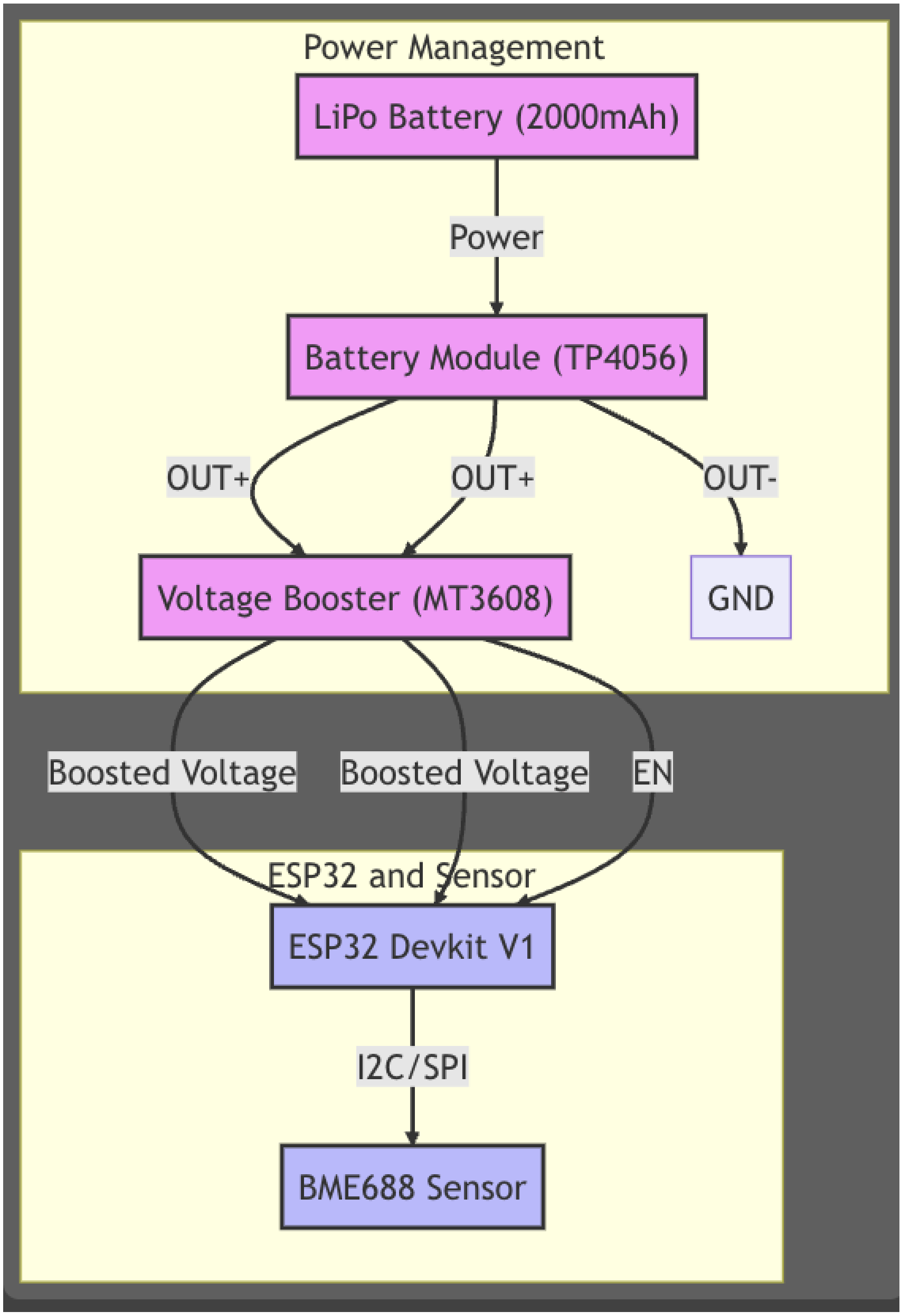


**Figure 7:** Butterfly diagram representing Current flow from battery to edge sensor

**Power Budget:**

| Component | Voltage (V) | Current (mA) | Power (W) | Operating Time (hrs/day) | Energy Consumption (Wh/day) |
|---|---|---|---|---|---|
| Raspberry Pi 5 | 5.0 | 3000 | 15.0 | 24 | 360 |
| ESP32 Boards (x3) | 3.3 | 200 each | 0.66 each | 24 | 47.52 |
| HC-SR501 Motion Sensor | 5.0 | 50 | 0.25 | 24 | 6.0 |
| Raspberry Pi HQ Camera | 5.0 | 250 | 1.25 | 8 (if motion detected) | 10.0 |
| BME688 Sensor | 3.3 | 15 | 0.0495 | 24 | 1.188 |
| TP-Link Kasa Smart Plug | 110-240 (AC) | 10 (standby) | 1.1 (standby) | 24 | 26.4 |
| Coral USB Accelerator | 5.0 | 500 | 2.5 | 12 | 30.0 |
| LED Indicators (x3) | 3.3 | 20 each | 0.066 each | 24 | 4.75 |
| Mobile App (React Native) | N/A | N/A | N/A | N/A | N/A (powered by user devices) |

**Table 5:** Power Budget

**Observations:**

1. Total Daily Energy Consumption: Adding up all the components, the system's total daily energy consumption is approximately 485.86 Wh/day.
2. Power-Hungry Components: The Raspberry Pi 5 and Raspberry Pi4 are the most power-intensive components. Optimizing their usage (e.g., powering down when idle) could significantly reduce energy consumption.
3. Low-Power Components: The BME688 sensor and motion sensor (HC-SR501) consume negligible power compared to other components, making them ideal for continuous monitoring.
4. Optimization Strategies:

- Used power-saving modes for ESP32 boards and sensors during idle periods.
- Limited camera and Coral USB Accelerator activity to motion-triggered events.
- Explored scheduling tasks on the Raspberry Pi to reduce active power usage.

## Team Information Sheet

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.

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

**Team Story:**

The Domus team was formed out of a shared vision among its members to redefine smart home technology by prioritizing privacy, efficiency, and user-centric design. Comprising engineers from diverse disciplines, the team recognized the growing need for home automation systems that deliver real-time control and enhanced security without compromising data privacy. Inspired by the Latin word "domus," meaning "home," the team set out to create an innovative, edge-computing-based solution that integrates seamlessly into modern lifestyles. Together, they aim to make Domus not just a product, but a step forward in creating smarter, safer, and more connected living spaces.

## References

- **Datasheets and Documentation**:
  Raspberry Pi 5: *"Raspberry Pi 5 Specifications and User Manual." Raspberry Pi Foundation, 2024.*https://www.raspberrypi.com/documentation
  ESP32: *"ESP32 Technical Reference Manual." Espressif Systems, 2024.*https://www.espressif.com/en/support/download/documents
  Coral USB Accelerator: *"Coral USB Accelerator Datasheet." Google Coral, 2024.*https://coral.ai/products/accelerator

- **AI and ML Frameworks**:
  OpenCV: *"OpenCV Python Documentation." OpenCV Team, 2024.*
  https://docs.opencv.org
  TensorFlow Lite for Coral: *"TensorFlow Lite Model Optimization." TensorFlow Team, 2024.*https://www.tensorflow.org/lite

- **Coral USB Accelerator:**
  *"Coral USB Accelerator." Google Coral, 2024.* Available at:
  https://coral.ai/products/accelerator

- **Environmental Sensors**:
  BME688: *"BME688 Integrated Environmental Sensor Datasheet." Bosch Sensortec, 2024.*https://www.bosch-sensortec.com

- **Energy Monitoring**:
  TP-Link Kasa Smart Plug: *"KP115 User Manual." TP-Link, 2024.*
  https://www.tp-link.com

- **Technical Reference for Coral USB Accelerator Setup:**
  *"Setup Coral TPU USB Accelerator on Raspberry Pi 5." DIY Engineers, May 18, 2024.* Available at:
  **https://www.diyengineers.com/2024/05/18/setup-coral-tpu-usb-accelerator-on-raspberry-pi-5/#google_vignette**

- **React Native Framework**:
  *"React Native Documentation." Meta, 2024.* https://reactnative.dev

*Authored by Venkata Sai Andra*