Senior Design
ENG EC 463

First Prototype Test Report

Team 27
Venkata Sai Sreeram Andra (vssa8989@bu.edu)
Anirudh Singh (ansingh@bu.edu)
Vansh Bhatia (vansh@bu.edu)
Akhil Bongu (akbongu@bu.edu)

Table of Contents

1 **Required Materials……………………………………………..……………………………3**

1.1 Hardware……………………………………………………………………………...3

1.2 Software……………………………………………………………………………....4

2 **Set Up…………………………………………………………………………....................4**

2.1 Component Setup…………………………………………………………………..4

2.2 Pre-testing Procedure…………………………………………………………………6

2.3 Testing Procedure…………………………………………………………………......7

3 **Prototype Measurements………………………………………………………………8**

4 **Conclusion…………………………………………………………………...….....9**

4.1 Evaluation……………………………………...……………………………..9

4.2 Problems……………………………………………………………………………..10

4.3 Next Steps…………………………………………………………………………...10

5 **Appendix………………………………………………………………...………..11**

1 Required Materials:

The materials used in this project include a combination of hardware and software components to facilitate real-time data collection, processing, and mobile application integration for our smart home system.

1.1: Hardware:

COMPONENT	DETAILS	PURPOSE
Raspberry Pi 5	32GB Amazon Basics microSD memory card included	Acts as the central processor for the system
Raspberry Pi 4B	32GB Amazon Basics microSD memory card included	Serves as an edge processor for the camera module.
ESP32 Microcontrollers	Two microcontrollers with Wi-Fi and Bluetooth capabilities	Collect and transmit data wirelessly
BME688 Sensor	Air quality sensor measuring temperature, humidity, and gas resistance.	Monitors indoor environmental conditions.
Maker Focus Raspberry Camera	5MP camera with IR LED light and photoresistor.	Provides live video feed for theft detection.
HC-SR501 Motion Sensor	Passive infrared sensor with a wide field of view.	Detects motion for real-time alerts.
XL6009 Voltage Boost Converter	Step-up voltage transformer.	Powers ESP32 boards and sensors.
TP4056 Charging Module	Rechargeable lithium battery charger with protection circuit.	Charges LiPo batteries for edge sensors.
LiPo 3.7V 2000mAh Batteries	Rechargeable lithium-ion batteries.	Powers sensors and microcontrollers.
TP-Link KP115 Smart Plug	Wi-Fi-enabled smart plug.	Tracks energy usage and provides toggle control.

Table 1: Hardware Components Outline

1.2: Software:

TOOL/SCRIPT	DETAILS	PURPOSE
Python Scripts	Flask server and WebSocket.	Relays real-time sensor data to the application.
	GStreamer pipeline.	Streams live video footage.
	dlib + OpenCV.	Collect and transmit data wirelessly
C++ Scripts	Used for low-level data fetching from sensors.	Captures real-time sensor data.
React Native Framework	JavaScript-based mobile app development framework.	Displays sensor data and theft detection outputs.

Table 2: Software Components Outline

2 Set Up

2.1: Component Setup

Facial Recognition Model: The Maker Focus 5MP camera was connected to the Raspberry Pi 4 via the CSI port. The Raspberry Pi 4 was configured as an edge processor to stream video footage to the Raspberry Pi 5 using a GStreamer pipeline, ensuring low-latency transmission. The facial recognition system was implemented using OpenCV for image processing and the face_recognition library from dlib for detecting and distinguishing between authorized and unauthorized individuals. Two persons' faces from the team were used to train the model by generating a facial encoding.

Environmental Monitoring: The BME688 air quality sensor was connected to an ESP32 microcontroller powered by a rechargeable LiPo battery, and connected to a 5V boost converter to ensure streamline flow of current. The ESP32 transmitted real-time sensor data (temperature, humidity, gas resistance, and pressure) wirelessly to the Raspberry Pi 5 over the HTTP protocol. Python scripts on the Raspberry Pi 5 processed the data and displayed it in the React Native app, ensuring accuracy and consistency with expected sensor readings.

Motion Detection Setup: The HC-SR501 motion sensor was connected to another ESP32 microcontroller powered by a rechargeable LiPo battery. Motion detection events were transmitted wirelessly to the Raspberry Pi 5 via the HTTP protocol for real-time processing. Motion detection data was displayed in real-time on the React Native app, verifying system responsiveness.

Energy Monitoring Setup: The TP-Link KP115 smart plug was connected to two home devices (e.g., a fan and phone charger). Python scripts on the Raspberry Pi 5 collected power consumption data from the smart plug via Wi-Fi and sent it to the React Native app for display, through the HTTP protocol. The app allowed toggling the smart plug on and off in real time.

React Native Application Setup:

- Backend Integration: The React Native app was configured to interact with the Flask server hosted on the Pi5. HTTP GET requests and WebSocket connections were established for real-time updates from the sensors and camera feed.
- Mobile App development: The app was built and deployed using npm using an iOS simulator powered by Xcode.
- UI Setup: The app interface was designed using JavaScript as a simple and user-friendly interface to provide alerts, display data and control the smart plug.

2.2 Pre-testing Setup Procedure

1. Power On the Raspberry Pi 5 and Raspberry Pi 4

   a) Connect the Raspberry Pi 5 to a power source using its USB-C cable and ensure the preloaded microSDXC card is inserted.

   b) Connect the Raspberry Pi 4 to a power source using its USB-C cable.

   c) Wait for the Raspberry Pi 5 to boot up and verify its connection to the local Wi-Fi network using terminal commands or the system settings interface.

2. Sensor Validation

   a) Ensure that the BME688 (air quality sensor) and HC-SR501 (motion sensor) are powered through their respective ESP32 edge processors, which are connected to their individual charging circuits.

   b) Execute the pre-installed Python scripts on the Raspberry Pi 5 to validate that sensor data is being transmitted wirelessly and received correctly.

   c) Inspect the console logs for possible transmission errors or lapses in the data relay.

3. React Native App Setup

   a) Launch the React Native app on a connected mobile device or emulator.

   b) Ensure the app loads successfully to the home screen.

4. Feature Verification: Navigate to each screen within the app to verify functionality of Motion Sensing, Theft Detection, Energy Tracking.

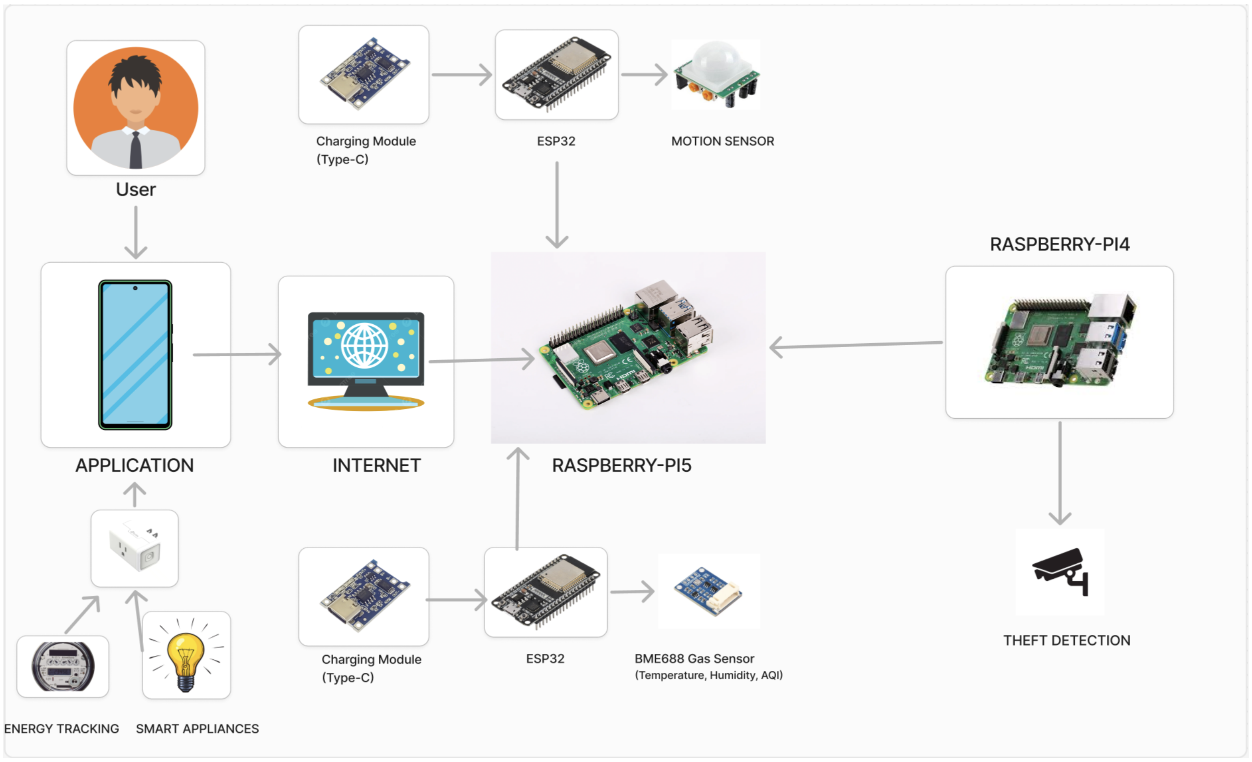


Figure 1: Illustration of Setup and Process Flow

2.3 Testing Procedure:

1. **Live Motion Detection:** Wave hand in front of the HC-SR501 motion sensor. Verify that motion detection events are displayed on the React Native app in real time.

2. **Face Recognition:** Should generate alerts for both authorized and unauthorized personnel in front of the camera, by displaying authorized and unknown face in the camera's field of view.

3. **BME688 Sensor:**

   - Temperature/Humidity: Should display temperature (in °C) and humidity (in %).

   - Air Quality: Gas resistance should be displayed (in Ohms), which is used as an indicator for air quality.

- Atmospheric pressure (in hPa) must be displayed on the app.

4. **Energy tracking:** Video demo of smart plug integration with two home devices (phone charger and fan). Verify that the React Native app displays real time power consumption data for each device and test the toggle functionality to switch between on and off.

3 Prototype Measurements

1. **Motion Detection:** The motion sensor successfully detected movements within its field of view and transmitted data wirelessly to the Raspberry Pi 5. The mobile application displayed these motion events in real-time with 100% accuracy (Table 1).

2. **Facial Detection:** The face detection system differentiated between authorized and unauthorized individuals. The mobile application displayed alerts with facial recognition latency under 1 second, ensuring timely theft detection (Table 1).

3. **Environmental Sensing:** The BME688 sensor transmitted accurate environmental data (temperature, humidity, gas resistance, and atmospheric pressure) to the Raspberry Pi 5. These metrics were displayed in the app with real time updates.

4. **Energy Tracking:** The smart plug monitored the energy consumption of connected devices and allowed toggling devices on/off through the app. Power consumption data was displayed in real-time with toggle latency of less than 1 second (Table 3).

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 3: Summary of outcomes

4 Conclusion

4.1 Evaluation

The setup of the Raspberry Pi 5, Raspberry Pi 4 and connected sensors via ESP32 microcontrollers was consistent with the test plan, ensuring seamless real-time data collection and processing. Each component operated as intended during testing. The motion sensor detected movements with 100% accuracy, and the BME688 provided environmental data reliably to the React-Native iOS application with latency under 1s. Energy consumption data was streamed in real-time, and smart plug toggle functionality with a latency of under 1 second. Furthermore, the facial detection functionality distinguishes between an authorized and unauthorized person with 100% accuracy and minimal/zero latency. Overall, the system demonstrated reliable performance, as shown in the output figures in the Appendix.

4.2 Problems

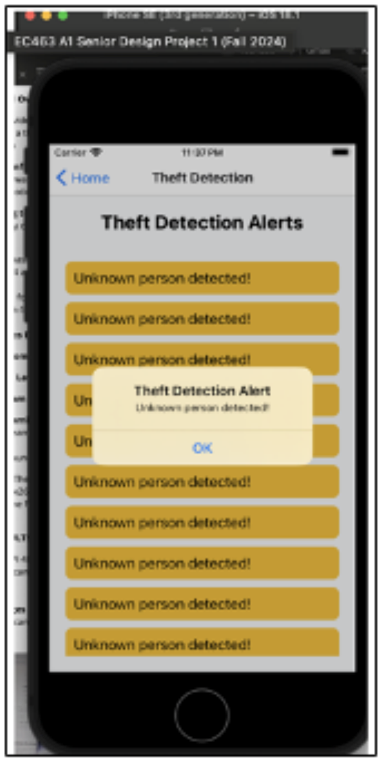
We were using Lithium batteries to power our edge sensors. We used charging modules and step-up voltage transformers to create a rechargeable battery that would constantly power the ESP32 microcontroller that was being used to connect the sensors wirelessly to the Pi5. While this setup works perfectly and we encountered no problems, the professor suggested that we create a safer circuit that takes into consideration the power consumption of each individual sensor, possibly without the use of Lithium Powered batteries for safety concerns. The camera was able to distinguish between the authorized and unauthorized faces, however, the current computer vision model lacks robustness and is potentially vulnerable by an image of the authorized user. The camera also displayed high latency due to poor network connectivity in the testing environment (Senior Design Lab).

4.3 Next Steps

- Upgrade camera module for better quality footage and reduce latency of the stream. Redesign the circuit focusing on safety and find a safer alternative for the LiPo Batteries.
- Train a more advanced computer vision learning model for better facial recognition functionality and leverage this model to achieve other use cases including object detection and motion detection-based camera control.
- Integrate the Coral USB Accelerator for better inference speed and efficiency of our AI models.
- Incorporate the LLM for predictive energy usage analysis, and quantitative modeling.
- Upgrade to a more ergonomic housing unit focusing on better aesthetics and usability.

5 Appendix

Images of iOS Application (showing output)


EC463 A1 Senior Design Project 1 (Fall 2024)

Carrier 11:07 PM

Home Theft Detection

Theft Detection Alerts

Unknown person detected!

Unknown person detected!

Unknown person detected!

Un

Un

Unknown person detected!

Unknown person detected!

Unknown person detected!

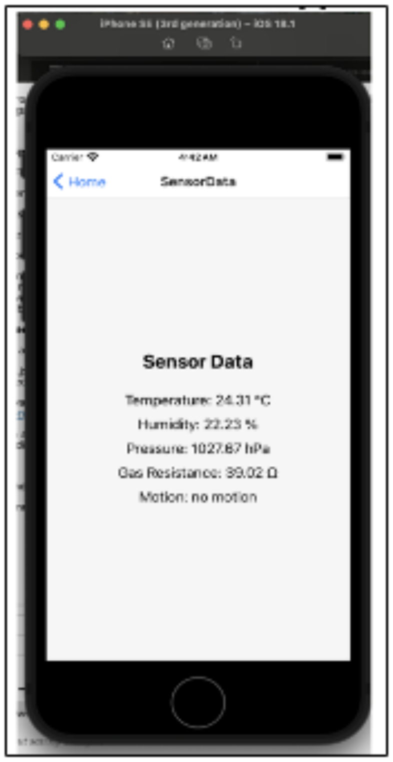
Unknown person detected!

Unknown person detected!

Theft Detection Alert

Unknown person detected!

OK


Carrier 4:42 AM

Home SensorData

Sensor Data

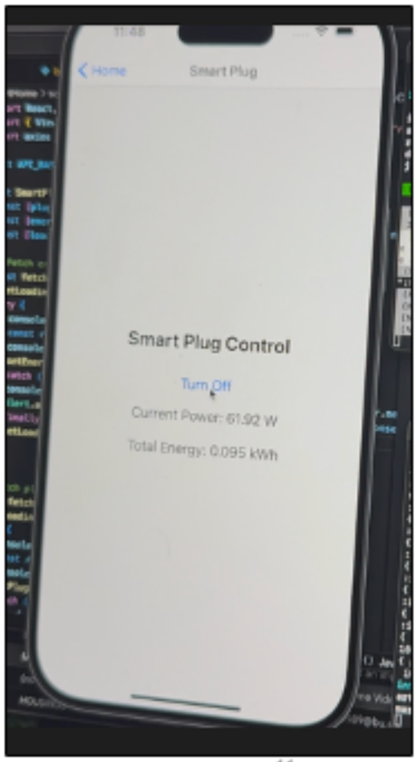
Temperature: 24.31 °C

Humidity: 22.23 %

Pressure: 1027.87 hPa

Gas Resistance: 39.02 Ω

Motion: no motion


11:48

Home Smart Plug

Smart Plug Control

Turn Off

Current Power: 61.92 W

Total Energy: 0.095 kWh

Video Demonstration (of smart plug functionality):

https://drive.google.com/file/d/1WXRDXypXxjN8U8YxBPOfypv0rjeruogk/view?usp=share_link