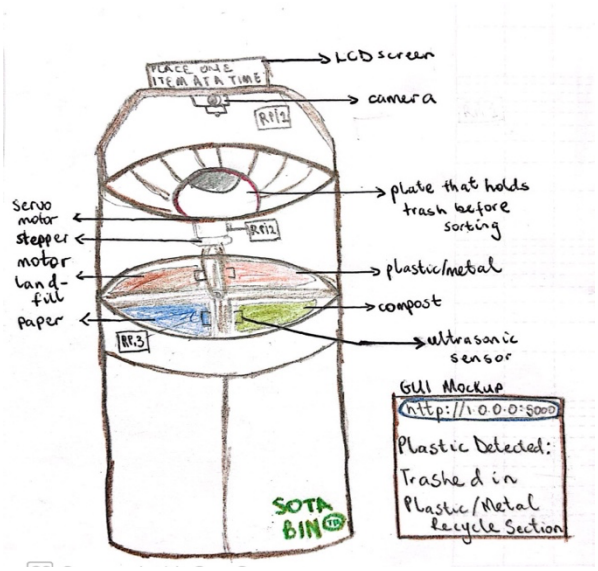
SYSC3010
Computer Systems Development Project

Sota Bin:

State-of-the-Art Sorting for a Sustainable Future

Detailed Design



Group L2-G3

Uchenna Obikwelu, 101241887
Adeyehun Folahanmi, 101237546
Dearell Tobenna Ezeoke, 101245819
Emeka Anonyei, 101209704
Tobiloba Ola, 101244412

TA: Afsoon Khodaee

March 10th, 2025

Table of Contents

1 Problem Statement

Waste misclassification is a persistent issue that contaminates recyclables, increases landfill waste, and reduces recycling efficiency. Many users struggle to sort waste properly due to the lack of real-time guidance and intuitive feedback mechanisms. Conventional multi-compartment bins rely solely on user discretion, leading to frequent sorting errors.

Sota Bin is a sensor-driven and rule-based waste management system that integrates computer vision (CV), sensor-based detection, and automated sorting to improve waste classification accuracy. The system classifies waste into four categories—landfill, paper, plastic/metal, and compost—by analyzing color, shape, texture, and material properties. Infrared sensors help differentiate compostable and paper materials, while a metal sensor ensures plastic and metal are correctly grouped. Ultrasonic sensors monitor bin capacity, ensuring waste is distributed efficiently. Users receive real-time classification feedback via the LED display and web interface, ensuring informed disposal decisions.

To enhance data tracking and waste analytics, Sota Bin integrates a local SQLite database for offline logging and Firebase cloud storage for real-time synchronization. By combining computer vision, material detection, automated sorting, and live status updates, Sota Bin provides a scalable, data-driven waste management solution that enhances recycling efficiency and promotes environmental sustainability.

1.1 Functional Requirements

The Sota Bin system shall meet the following functional requirements:

1. Waste Classification & Sorting:
    - The system shall capture an image of the waste item when placed in the bin.
    - The system shall use OpenCV-based image processing for classification.
    - The system shall sort waste using servo and stepper motors to direct it into the correct compartment.
2. User Feedback & Notifications:
    - The system shall display classification results on an LED screen immediately after detection.
    - The system shall provide real-time user notifications via the web interface.
    - The system shall alert managers when a bin is full through the Sense HAT LED Matrix and Firebase notifications.
3. Data Logging & Analytics:
    - The system shall store classification results in a local SQLite database for tracking disposal patterns.
    - The system shall synchronize bin status updates with Firebase for remote monitoring.

- The system shall generate waste disposal analytics, providing insights for users, businesses, and municipalities.

2 Design Overview

2.1 System Overview Diagram

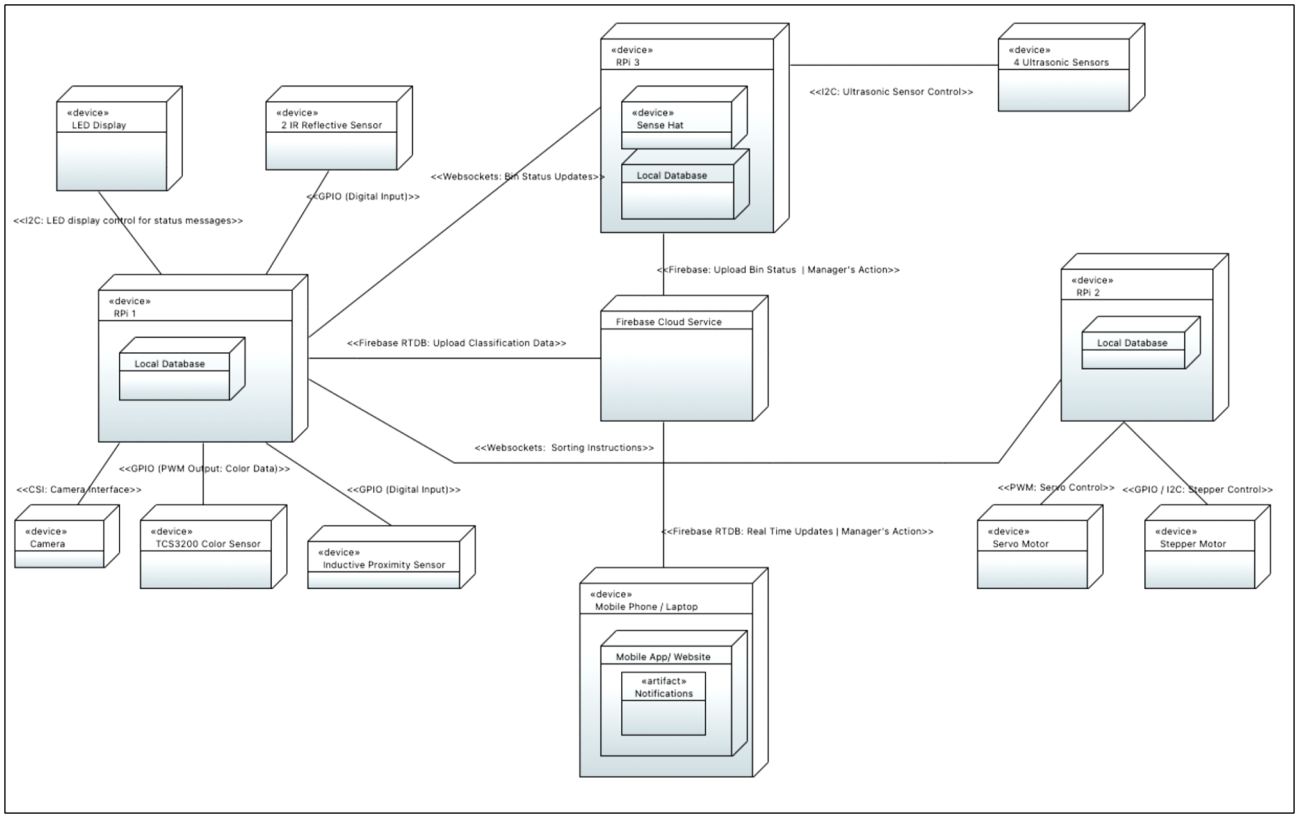


Figure 1: Deployment diagram for Sota Bin

Figure 1 illustrates the high-level architecture of Sota Bin, a smart waste classification system that automates waste sorting using a combination of sensors, actuators, and cloud services. The system consists of three Raspberry Pis, a Firebase Cloud Service, and a web interface for user interaction.

- RPi1 handles object detection and classification, utilizing a camera, color sensor, IR reflective sensor, and inductive proximity sensor. It sends sorting data to RPi2 via WebSockets for real-time processing and uploads classification results to Firebase for cloud storage.
- RPi2 receives sorting data from RPi1 via WebSockets and controls the servo and stepper motors both via GPIO, directing waste into the appropriate sections based on classification.
- RPi3 monitors bin fullness using ultrasonic sensors via the I2C protocol and updates the Firebase Cloud Service when a section is nearly full. RPi3 also contains the programs that controls the web interface and programs that that analyze the classification data.
- The Firebase Cloud Service acts as the central hub, enabling real-time updates and remote data access.

- The web interface provides users with sorting history, real-time bin status, detailed analytics on waste distribution, and SMS notifications. Managers have access to all user features, along with additional insights into the bin sections' status, including whether they are full, half-full, or empty. Once a full bin is emptied, managers can reset the Sense HAT on RPi3 via the web platform. This action triggers a command through the Firebase Realtime Database, ensuring synchronization between the interface and the physical system.

We designed the system this way to keep things modular, scalable, and easy to manage. By using three Raspberry Pis, each handling a specific set of tasks, we avoid overloading a single device and also make it easier to divide work among team members.

WebSockets allow the Raspberry Pis to communicate instantly in real-time. The use of a Backend-as-a-Service platform: Firebase, allows to leverage real-time databases, authentication, and data storage. This setup keeps things organized and makes both testing and future upgrades much simpler.

2.2 Communication Protocols

For an efficient and real-time communication between Sota Bin's components, a combination of GPIO, I2C, WebSockets, and Firebase Realtime Database (RTDB) is used to manage data flow between the Raspberry Pis, sensors and motors.

2.2.1 Communication Between Raspberry Pi 1 and Connected Hardware Devices

RPi1 is responsible for waste classification, object detection, and controlling the LED Display to provide key updates to users. To enable its functioning, 5 hardware devices including, a Camera Module (for object detection and classification), a TCS3200 Color Sensor (to determine the color of waste items), an Inductive Proximity Sensor (to detect metal objects), an IR Reflective Sensor (to sense the presence of waste on the rotating plate) and a LED Display (16x2 I2C LCD) (to provide real-time status updates to users) is connected to it.

Hardware Device	**Protocol**
Camera Module	Camera Serial Interface (CSI)
TCS3200 Colour Sensor	GPIO (PWM Output)
Inductive Proximity Sensor	GPIO (Digital Input)
IR Reflective Sensor	GPIO (Digital Input)
LED Display	I2C

2.2.2 Communication Between Raspberry Pi 2 and Connected Hardware Devices

RPi2 is responsible for motor control, ensuring proper waste sorting by directing waste items into the correct sections. To achieve this, two motors are connected: a Servo Motor (to tilt the rotating plate and drop waste) and a Stepper Motor (to rotate the plate to align with the correct bin section). RPi2 receives sorting instructions from RPi1 via WebSockets and executes the sorting mechanism accordingly.

Hardware Device	**Protocol**
Servo Motor (DFRobot SERVOMOTOR RC 4.8-6V)	GPIO (PWM Output)
Stepper Motor (NEMA 17 Hybrid Stepper)	GPIO (Step, Direction) via A4988 Stepper Driver

2.2.3 Communication Between Raspberry Pi 3 and Connected Hardware Devices

RPi3 is responsible for monitoring bin fullness levels and providing real-time status updates through the Sense HAT. It uses four ultrasonic sensors, each corresponding to a specific bin section (Compost, Paper, Plastic/Metal, Landfill), to measure the fill levels. The Sense HAT displays real-time status updates and can be remotely reset by the Manager via the web interface.

Hardware Device	**Protocol**
4 Ultrasonic Sensors (HC-SR04)	I2C
Sense HAT (LED Matrix)	I2C

2.2.4 Communication Between Raspberry Pi 1, Raspberry Pi 2, and Raspberry Pi 3

To facilitate real-time communication between the Raspberry Pi computers, **WebSockets** are used as the primary IoT communication tool. This enables low-latency, bidirectional communication over a local network, ensuring immediate responses to classification and sorting events without relying on frequent database queries.

- RPi1 to RPi2: After classifying an item using its connected sensors, RPi1 transmits the classification result to RPi2 via WebSockets. Upon receiving this data, RPi2 determines the appropriate motor movements needed to direct the waste item into its designated bin section.

- RPi3 to RPi1: RPi3 continuously monitors bin capacity using ultrasonic sensors. When a bin section is detected as full, RPi3 sends a WebSocket message to RPi1, triggering an update on the LED display to notify users. Additionally, RPi1 ensures that new waste is not directed into a full section, preventing overflow.

2.2.5 Communication Between Raspberry Pis, Web Interface, and the Firebase Cloud Server

For centralized data storage and remote access, we are using Firebase Cloud Services, with the Real-Time Database (RTDB) as the IoT tool.

While WebSockets manage real-time communication between Raspberry Pis, Firebase serves as the central cloud service for remote data storage and access and providing real time updates to the web interface.

How system components interact with Firebase for specific tasks:

- RPi1 to Firebase: RPi1 uploads classification data (waste type and timestamp) to Firebase.

- RPi3 to Firebase: RPi3 updates Firebase RTDB with the bin fill level (empty/half/empty).

- Web Interface to Firebase: The web application allows users to monitor waste classification history and bin fill levels in real time. Managers can reset bin statuses through the web interface, which updates Firebase RTDB. RPi3 reads this update and resets the Sense HAT display

accordingly. All interactions between the web interface and the system are stored and retrieved from Firebase.

- Firebase to RPi3: Bin reset commands from the web interface, logged in Firebase, clear the Sense HAT display on the RPi3 in real time, changing it from red to a cleared state.

2.2.6 Communication Protocol Table

Table 1: Communication Between Raspberry Pi 1 and Connected Hardware Devices

Sender	Receiver	Message Type	Data Format	Units/Limits	Storage Location
Camera Module	RPi1	Image Data	JPEG/PNG image frame	N/A	/home/pi/sota_bin/images/
TCS3200 Color Sensor	RPi1	Color Reading	RGB Values (R: 0-255, G: 0-255, B: 0-255)	None	/home/pi/sota_bin/data/ rpi1_sensor_logs.db
Inductive Proximity Sensor	RPi1	Metal Detection	Digital Signal (1 = Detected, 0 = Not Detected)	Binary	/home/pi/sota_bin/data/ rpi1_sensor_logs.db
IR Reflective Sensor	RPi1	Material Classification	Digital Signal (1 = High Reflection, 0 = Low Reflection)	Binary	/home/pi/sota_bin/data/ rpi1_sensor_logs.db
RPi1	LED Display (I2C)	Status Message	Text String ("Sorting", "Error", etc.)	16x2 Char Limit	Not Applicable (Direct Display)

Table 2: Communication Between Raspberry Pi 2 and Connected Hardware Devices

Sender	Receiver	Message Type	Data Format	Units/Limits	Storage Location
RPi2	Servo Motor	Angle Position Command	Integer (0 - 180)	Degrees	Not Stored (Direct Actuation)
RPi2	Stepper Motor (via A4988)	Rotation Command	Step Pulses	N/A	Not Stored (Direct Actuation)

Table 3: Communication Between Raspberry Pi 3 and Connected Hardware Devices

Sender	Receiver	Message Type	Data Format	Units/Limits	Storage Location
Ultrasonic Sensors	RPi3	Distance Measurement	Integer (0 - 400)	Millimeters	/home/pi/sota_bin/ data/rpi3_sensor_logs.db
RPi3	Sense HAT	Bin Status Display	Binary State (Red or Blank)	Red = At least one section is full, Blank = No full sections	Not Stored (Direct Display)

Table 4: Communication Between Raspberry Pi 1, Raspberry Pi 2, and Raspberry Pi 3

Sender	Receiver	Message Type	Data Format	Units/Limits	Storage Location
RPi1	RPi2	Waste Classification	JSON: {"type": "plastic", "bin": "2"}	Sorting Execution	/home/pi/sota_bin/ data/rpi2_classifications.db
RPi3	RPi1	Bin Status Update	JSON: {"bin": "3", "status": "Full"}	Prevent Overflow	/home/pi/SOTA_bin/data/ rpi1_sensor_logs.db

Table 5: Communication Between Raspberry Pis, Firebase Cloud Service, and Web Interface

Sender	Receiver	Message Type	Data Format	Units/Limits	Storage Location
RPi1	Firebase	Classification Data	JSON: {"item": "plastic/metal", "bin": "2"}	Remote Monitoring	Firebase Realtime Database
RPi3	Firebase	Bin Full Status	JSON: {"bin": "3", "status": "Full"}	Web Dashboard	Firebase Realtime Database
Web Interface	Firebase	Manager Action	JSON: {"action": "reset", "bin": "3"}	Allows managers to reset bin status from the web.	Firebase Realtime Database
Firebase	RPi3	Reset Sense HAT	JSON: {"action": "reset"}	Remote Reset	Not Stored (Direct Execution)

2.3 Message Sequence Diagram(s)

Each Message Sequence Diagram (MSD) represents a specific system use case, detailing how messages are exchanged between hardware nodes (shown in the Deployment Diagram) to achieve key system functions. These diagrams illustrate how the system components interact in real-time using the communication protocols listed in the Communication Protocol Table.

Each figure referenced below provides visual representations of how various components work together to handle sorting, classification, bin monitoring, and remote updates.

2.3.1 Message Sequence Diagram 1: Sensor Data Flow to RPi1, RPi2, and Firebase

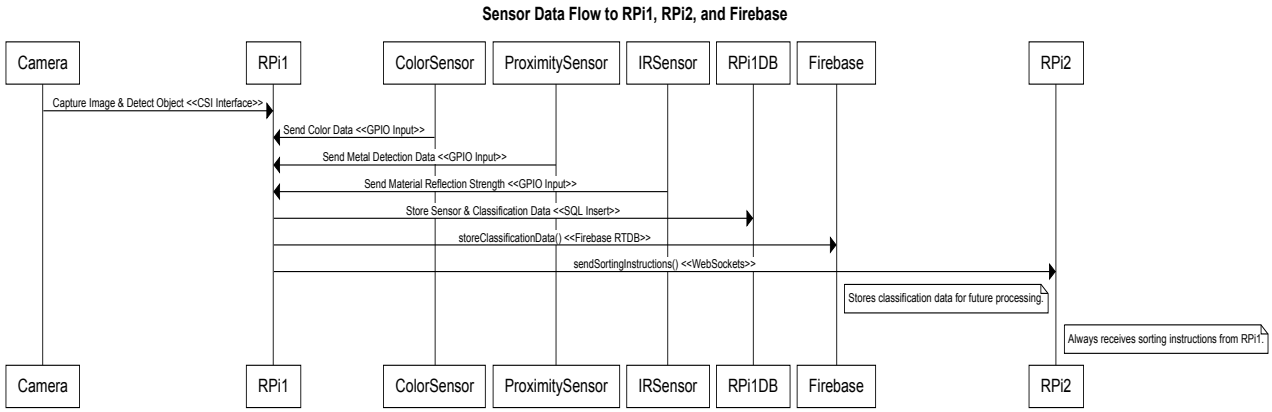


Figure 2: Sensor Data Flow to RPi1, RPi2, and Firebase

Figure 2 illustrates the first use case, where various sensors and the camera collect data from objects and send it to RPi1. The camera captures an image, while the color sensor, proximity sensor, and IR sensor analyze object properties. The processed data is then stored in RPi1's local database and sent to Firebase for storage. Finally, RPi1 sends sorting instructions to RPi2 to begin waste classification.

2.3.2 Message Sequence Diagram 2: RPi2 Controls Servo and Stepper Motor

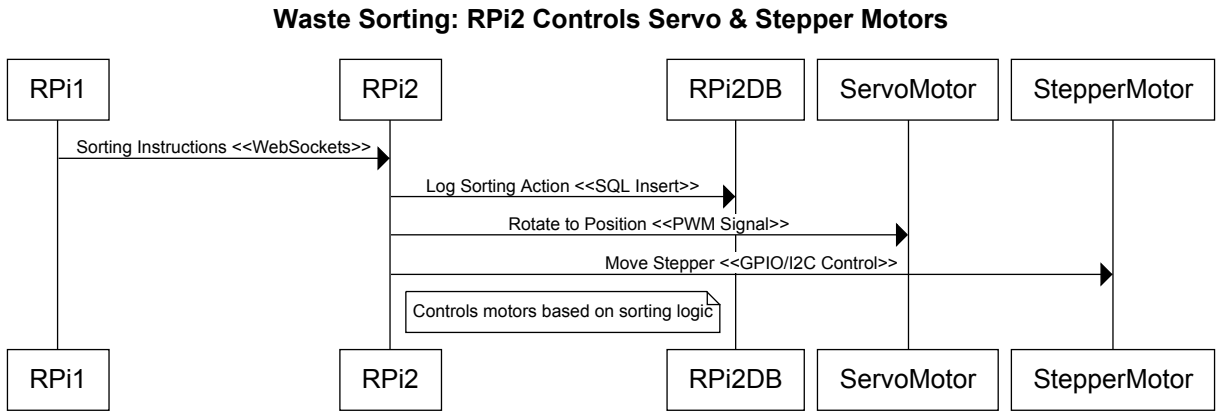


Figure 3: Sorting Mechanism – Communication Between RPi1 and RPi2

Figure 3 illustrates the use case where RPi2 receives sorting instructions from RPi1 and executes sorting operations. The servo motor rotates to direct the waste item, while the stepper motor moves accordingly to complete the sorting process. The action is logged in RPi2's local database to keep track of sorting operations.

2.3.3 Message Sequence Diagram 3: Bin Status Update (Half or Empty)

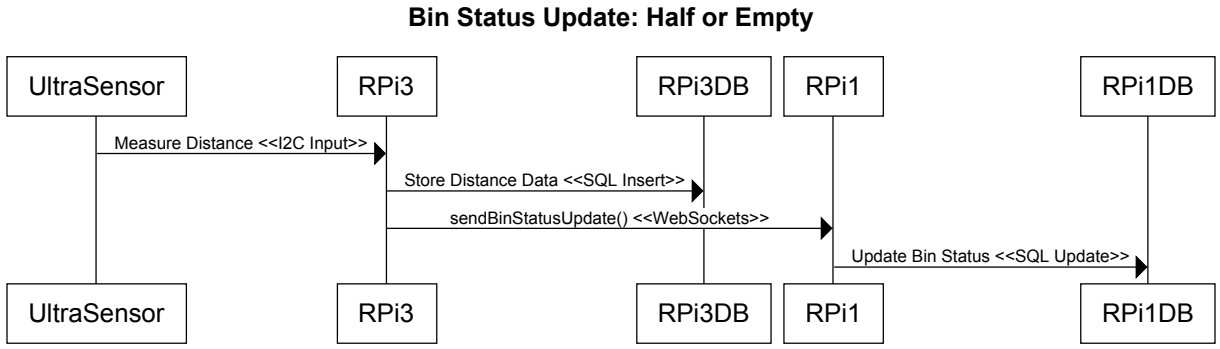


Figure 4: Bin Status Update - Half or Empty

Figure 4 illustrates the use case where an ultrasonic sensor on RPi3 measures the fill level of a bin. If the bin is half-full or empty, the sensor readings are stored in RPi3's local database and forwarded to RPi1 via WebSockets. RPi1 then updates its own database and sends the bin status to Firebase, allowing remote monitoring.

2.3.4 Message Sequence Diagram 4: Bin Full: Alerts & Sense HAT Activation

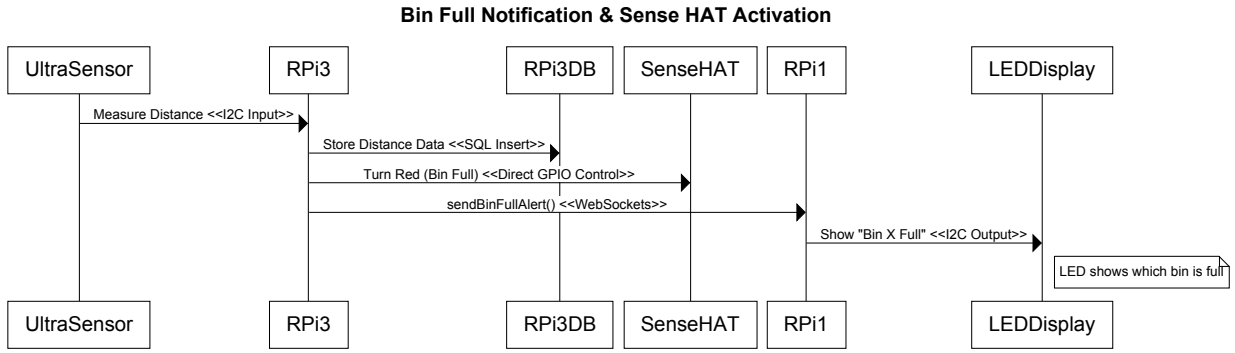


Figure 5: Full Bin Alert – RPi3 Updating LED Display and Firebase

Figure 5 illustrates the use case where an ultrasonic sensor detects that a bin is full. The bin's full status is stored in RPi3's database, and a notification is sent to RPi1. RPi3 activates the Sense HAT, turning it red to signal a full bin. At the same time, RPi1 updates the LED display to indicate which bin is full, providing a local visual alert.

2.3.5 Message Sequence Diagram 5: General Bin Status Update to Firebase

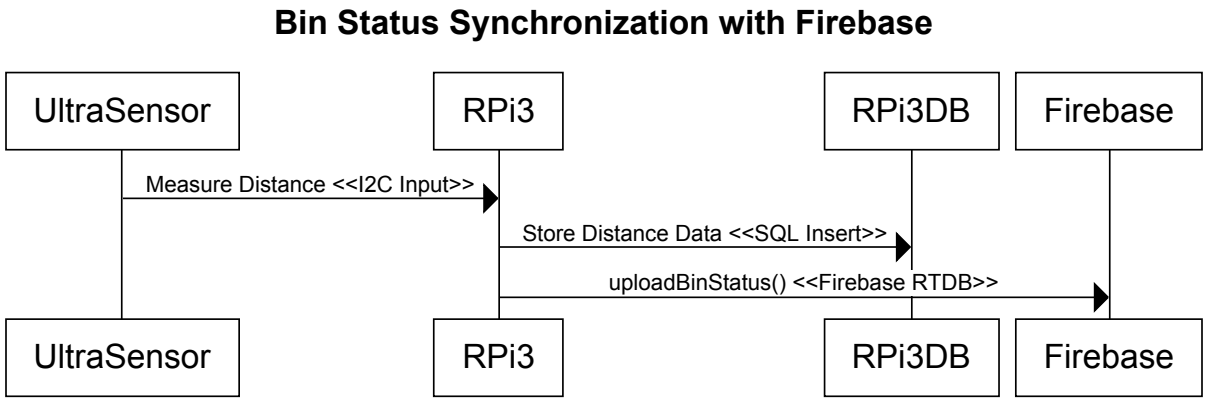


Figure 6: Real-Time Bin Status Updates to Firebase

Figure 6 illustrates the use case where RPi3 periodically updates Firebase with the latest bin status. The ultrasonic sensor continuously monitors the fill level, storing readings in RPi3's database before pushing updates to Firebase. This allows the web interface to display real-time bin statuses remotely.

2.3.6 Message Sequence Diagram 6: Manager Action to Clear Sense HAT via Web Interface

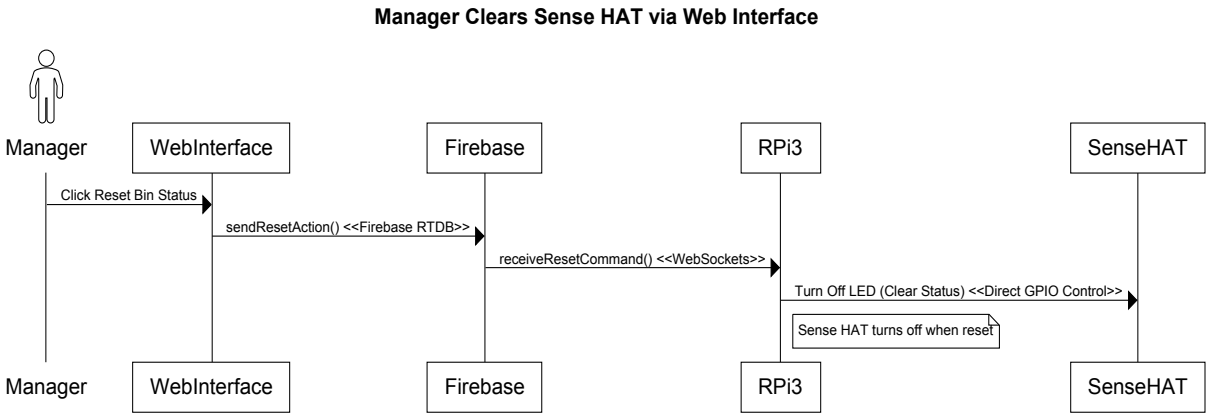


Figure 7: Manager Reset Action – Web Interface to RPi3 via Firebase

Figure 7 illustrates the use case where a manager clears the bin full alert via the web interface. The manager clicks a reset button, which sends a reset action to Firebase. RPi3 detects this action, processes the reset command, and turns off the Sense HAT display, clearing the red warning light.

2.3.7 Message Sequence Diagram 7: Real-Time Updates from to Web Interface

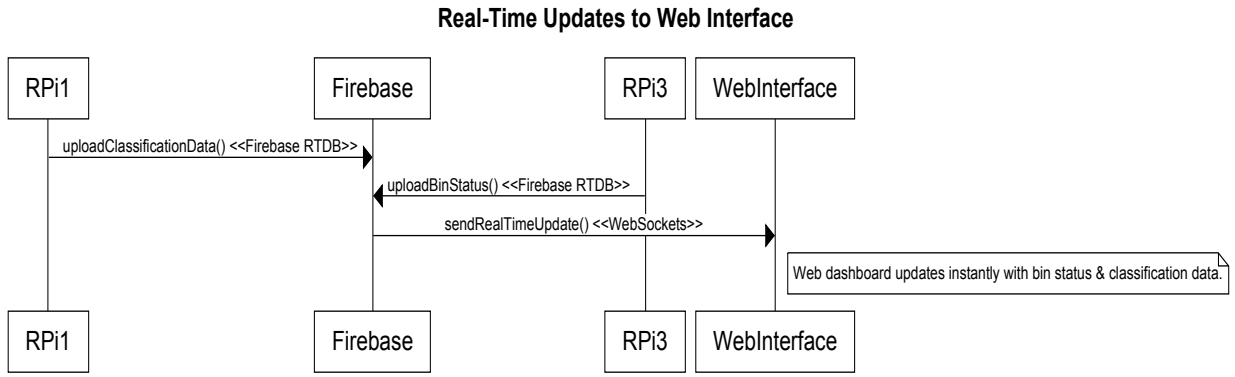


Figure 8: Web Interface Updating in Real-Time with Firebase Data

Figure 8 illustrates the use case where both classification data and bin status updates are sent to Firebase and displayed on the web interface. RPi1 uploads waste classification data, while RPi3 uploads bin status information. Firebase then pushes real-time updates to the web interface via WebSockets, ensuring users can see the latest data instantly.

2.4 Database Table Design/Schema

This section describes the database design used in the system, covering both local SQLite databases and Firebase NoSQL schema. The local databases are deployed on the Raspberry Pi devices, while Firebase is used for cloud-based monitoring and control.

2.4.1 Local Database Schemas (SQLite)

Each Raspberry Pi device maintains its own SQLite database to store relevant data locally.

Raspberry Pi	Database File	Purpose
RPi1	/home/pi/sota_bin/ data/rpi1_sensor_logs.db	Stores all sensor readings (Color Sensor, Proximity Sensor, IR Sensor) and bin status updates received from RPi3.
RPi2	/home/pi/sota_bin/ data/rpi2_classifications.db	Stores waste classification results from the sorting process.
RPi3	/home/pi/sota_bin/ data/rpi3_sensor_logs.db	Stores ultrasonic sensor readings for bin fill level detection.

2.4.1.1 2.4.1.1 RPi1: rpi1_sensor_logs.db (Sensor & Bin Status Logs)

TABLE color_readings
- id INTEGER PRIMARY KEY AUTOINCREMENT - timestamp DATETIME DEFAULT CURRENT_TIMESTAMP - red INTEGER CHECK(red BETWEEN 0 AND 255), - green INTEGER CHECK(green BETWEEN 0 AND 255) - blue INTEGER CHECK(blue BETWEEN 0 AND 255)

TABLE metal_detection
- id INTEGER PRIMARY KEY AUTOINCREMENT - timestamp DATETIME DEFAULT CURRENT_TIMESTAMP - detected INTEGER CHECK(detected IN (0,1))

TABLE presence_detection
- id INTEGER PRIMARY KEY AUTOINCREMENT - timestamp DATETIME DEFAULT CURRENT_TIMESTAMP - present INTEGER CHECK(present IN (0,1))

TABLE bin_status
- id INTEGER PRIMARY KEY AUTOINCREMENT - timestamp DATETIME DEFAULT CURRENT_TIMESTAMP - bin INTEGER CHECK(bin BETWEEN 1 AND 4) - status TEXT CHECK(status IN ('Full', 'Half', 'Empty'))

2.4.1.2 RPi2: rpi2_classifications.db (Waste Classification)

TABLE classifications
- id INTEGER PRIMARY KEY AUTOINCREMENT - timestamp DATETIME DEFAULT CURRENT_TIMESTAMP - bin INTEGER CHECK(bin BETWEEN 1 AND 4) - material TEXT CHECK(material IN ('plastic', 'paper', 'metal', 'organic', 'other'))

2.4.1.3 RPi3: rpi3_sensor_logs.db (Ultrasonic Sensor Readings)

TABLE classifications
- id INTEGER PRIMARY KEY AUTOINCREMENT - timestamp DATETIME DEFAULT CURRENT_TIMESTAMP - bin INTEGER CHECK(bin BETWEEN 1 AND 4) - distance INTEGER CHECK(distance BETWEEN 0 AND 400)

2.4.2 2.4.3 Cloud Database Schema (Firebase NoSQL)

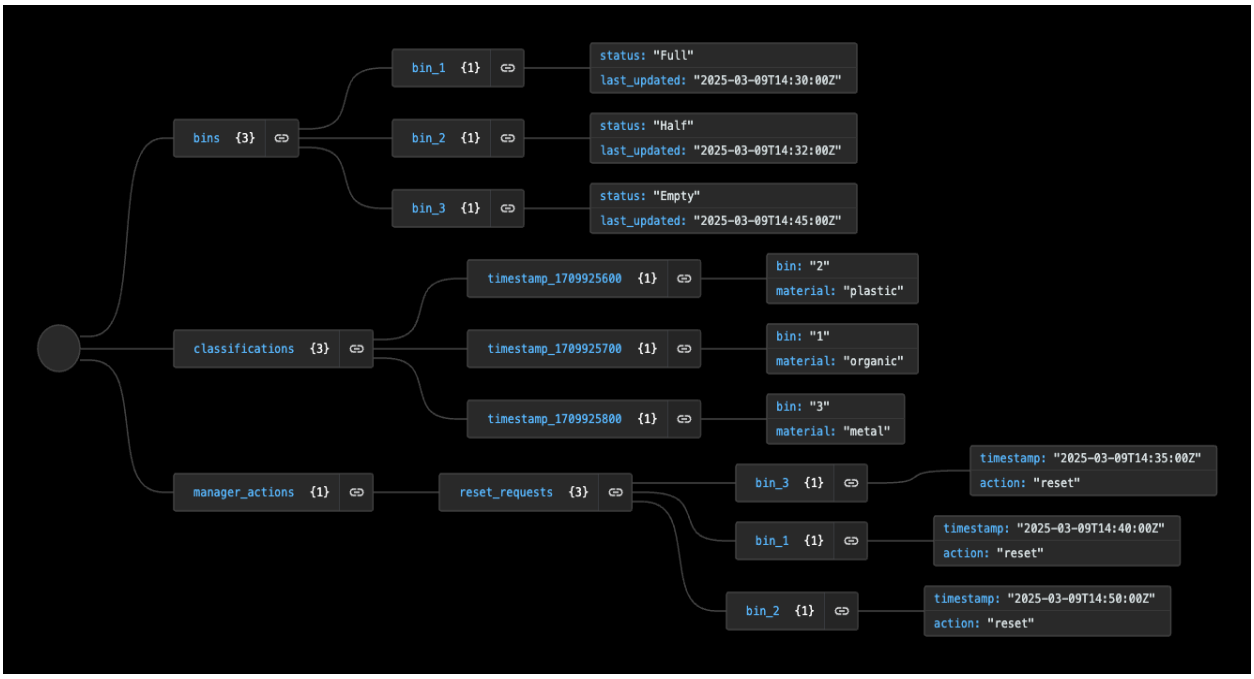


Figure 9: Cloud Database Schema Diagram (NoSQL)

Figure 9 above illustrates the hierarchical structure of the Firebase Realtime Database. The database consists of three main collections:

1. bins/ – Stores real-time bin status updates, indicating whether a bin is "Full," "Half," or "Empty." Each bin entry is indexed by a unique bin ID.
2. classifications/ – Logs waste classification events, associating each classified item with a bin. Data is indexed using a timestamp key to allow chronological sorting.
3. manager_actions/ – Contains administrator actions, such as bin reset requests. Each action is indexed by bin ID and includes a timestamp to track when the request was made.

The hierarchical JSON structure ensures that data retrieval is fast and efficient, with each category of information stored in its appropriate section for easy access.

```
{
  "bins": {
    "bin_1": {
      "status": "Full",
      "last_updated": "2025-03-09T14:30:00Z"
    },
    "bin_2": {
      "status": "Half",
      "last_updated": "2025-03-09T14:32:00Z"
    },
    "bin_3": {
      "status": "Empty",
      "last_updated": "2025-03-09T14:45:00Z"
    }

  },
  "classifications": {
    "timestamp_1709925600": {
      "bin": "2",
      "material": "plastic"
    },
    "timestamp_1709925700": {
      "bin": "1",
      "material": "organic"
    },
    "timestamp_1709925800": {
      "bin": "3",
      "material": "metal"
    }
  },
  "manager_actions": {
    "reset_requests": {
      "bin_3": {
        "timestamp": "2025-03-09T14:35:00Z",
        "action": "reset"
      },
      "bin_1": {
        "timestamp": "2025-03-09T14:40:00Z",
        "action": "reset"
      },
      "bin_2": {
        "timestamp": "2025-03-09T14:50:00Z",
        "action": "reset"
      }
    }
  }
}
```


Figure 10: Sample JSON structure for Firebase NoSQL Database

3 Software Design

This section describes the software architecture and implementation of Sota Bin, covering the Raspberry Pi physical nodes, backend server, WebSocket communication, and frontend web interface. The system follows a modular and scalable design to ensure efficient communication between hardware components, the server, and the web interface.

3.1 Software Design for Raspberry Pi Physical Nodes

Each Raspberry Pi (RPi) is responsible for a specific function within the system:

- RPi1: Object detection and waste classification
- RPi2: Motor control for waste sorting
- RPi3: Bin full monitoring and status updates

The software on these nodes is written in Python, utilizing OpenCV, RPi.GPIO, Firebase SDKs, and SQLite for local storage and real-time communication with the backend.

3.1.1 Software Design for Object Detection Subsystem (RPi1)

- Camera Module captures images of waste items.
- Computer Vision (OpenCV) processes images to classify waste into compost, paper, plastic/metal, or landfill.
- TCS3200 Color Sensor & Proximity Sensor provide additional classification input.
- Classification data is stored in the local SQLite database for offline use.
- Classification result is sent to the LED display via I2C.
- Data is uploaded to Firebase for synchronization across all system components.
- WebSockets are used to communicate sorting instructions to RPi2.

3.1.2 Software Design for Motor Control Subsystem (RPi2)

- Fetches sorting instructions from RPi1 via WebSockets.
- Servo Motor tilts the waste sorting plate to deposit waste correctly.
- Stepper Motor rotates the plate to align with the correct bin section.
- Motors are controlled via PWM and GPIO.

- If a network failure occurs, default sorting logic ensures waste is still processed.

3.1.3 Software Design for Bin Full Monitoring Subsystem (RPi3)

- Ultrasonic Sensors measure bin fill levels.
- Sense HAT LED matrix indicates full-bin status (red light = full, blank = normal).
- Data is stored in SQLite and transmitted to Firebase via WebSockets.
- If a bin is full, sorting is halted, and a notification is sent to the LED display, Firebase, and the Web Interface.

3.2 Software Design for Backend Server

The backend server acts as the central hub for communication between the Raspberry Pi nodes and the web interface. The backend is implemented using Flask, supporting SQLite for data management and WebSockets for real-time updates.

3.2.1 Software Design for SQLite Server

- The SQLite database stores structured data, including:
    - Waste classification results
    - Sorting instructions
    - Bin status updates
    - System logs and diagnostics
- Database integrity tests ensure all entries remain valid.
- Firebase Authentication is used for secure remote access.

3.2.2 Software Design for WebSocket Server

- WebSockets enable real-time, bidirectional communication between the Raspberry Pis and the Web Interface.
- Events like bin full alerts, classification updates, and sorting instructions are instantly pushed to connected clients.
- The system automatically reconnects in case of network failures.

3.3 Software Design for WebSocket Client

Both the Raspberry Pis and the Web Interface function as WebSocket clients:

- RPi nodes send sensor data and receive commands from the server.
- The Web Interface subscribes to live updates (e.g., classification results, bin full status).
- Persistent WebSocket connections ensure low-latency data exchange.

3.4 Software Design for Frontend Web Interface

The frontend provides a real-time dashboard for users to monitor and control the Sota Bin system. It is built using HTML, CSS, and JavaScript, with Firebase integration for real-time updates.

Key Features:

- Live Waste Classification Feed: Displays sorting results in real-time.
- Bin Status Monitoring: Notifies managers when a bin is full.
- Manager Controls: Allows remote bin resets.
- Analytics Dashboard: Displays waste trends and statistics.

3.5 Class Diagram

A class diagram is included to show the object-oriented structure of the software, depicting relationships between key components such as the CameraModule, SortingController, DatabaseManager, and WebSocketClient.

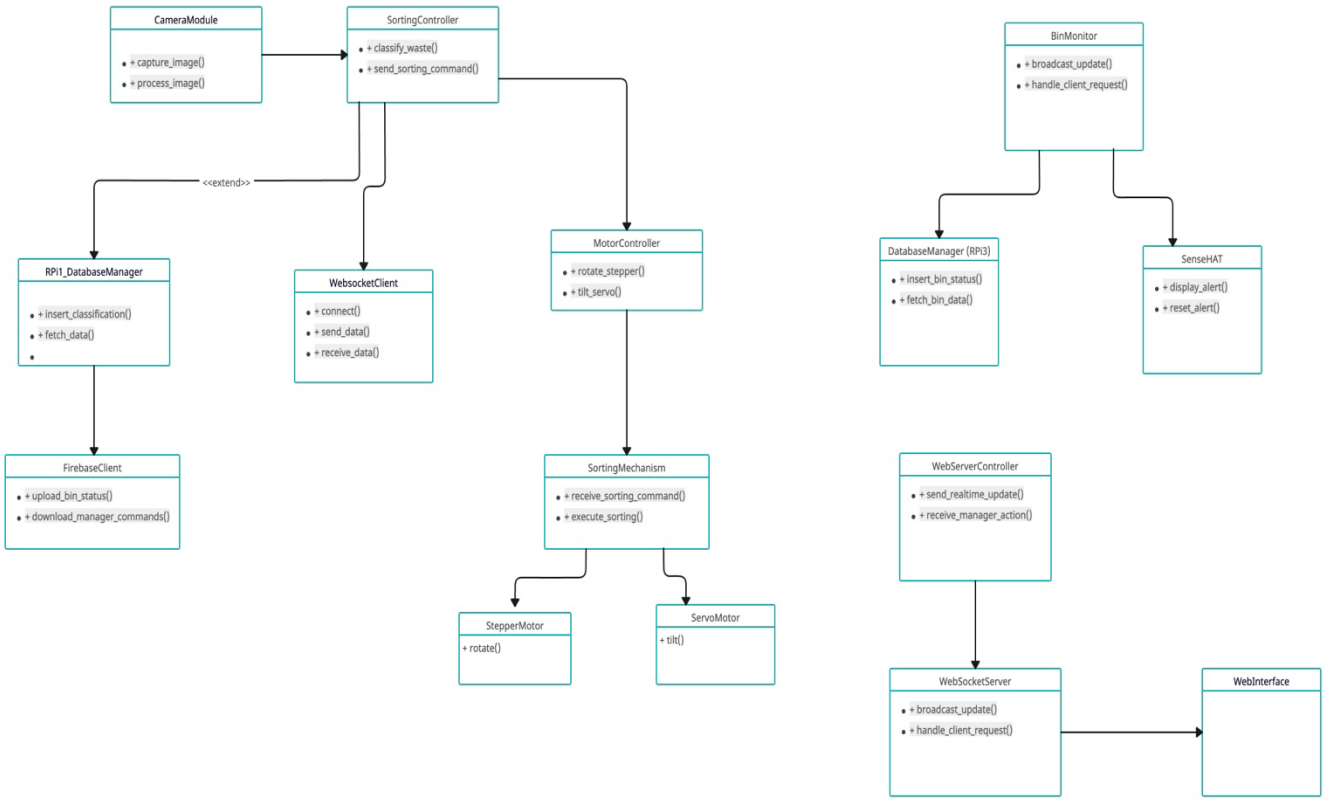


Figure 11: Sota bin class diagram

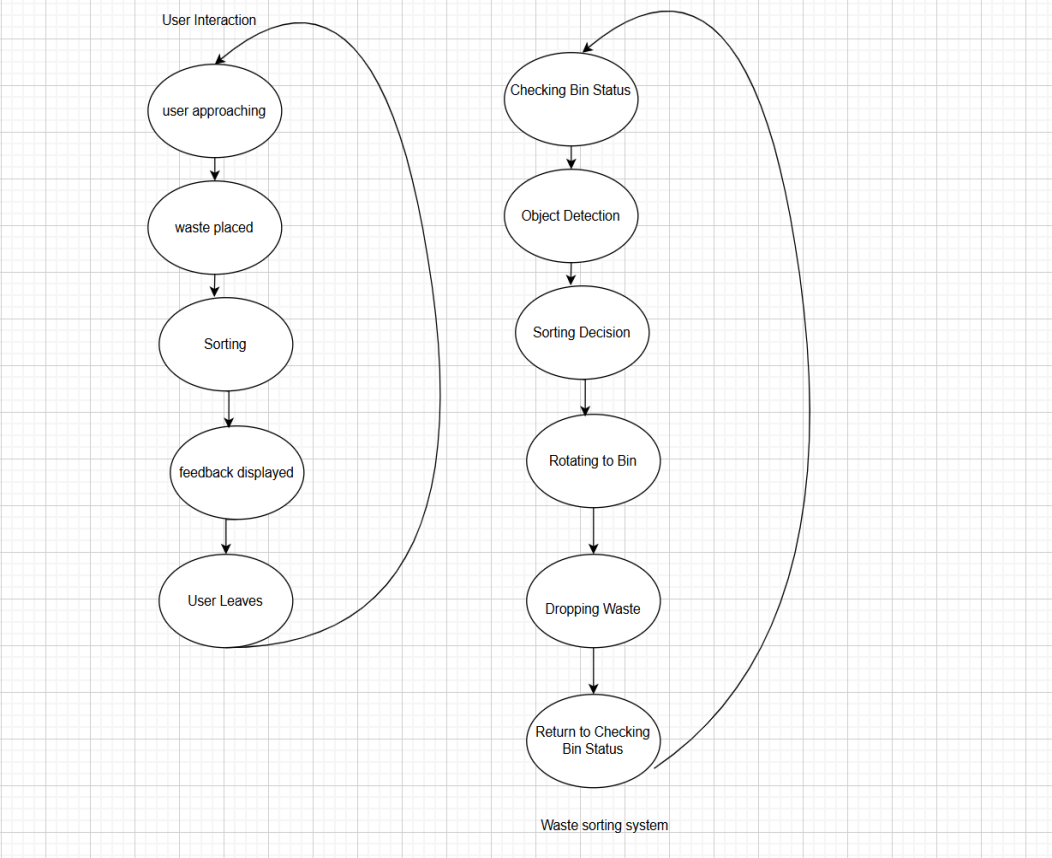


Figure 12: State Diagram for Waste Sorting System and User Interaction

4 Hardware Design

The Sota Bin system integrates multiple hardware components to facilitate waste classification, sorting, and bin monitoring. The system consists of three Raspberry Pis (RPis) that interact with motors, sensors, a camera, an LED display, and an external web server. The hardware components are designed for modularity, scalability, and reliability, ensuring efficient operation and future expansion.

4.1 Hardware Components Overview

The hardware design consists of the following key components:

Component	Function
Raspberry Pi 1 (RPi1)	Controls object detection, classification, and LED display for user feedback.
Raspberry Pi 2 (RPi2)	Controls stepper and servo motors for waste sorting.

Raspberry Pi 3 (RPi3)	Monitors bin status using ultrasonic sensors and controls the web interface
Camera Module	Captures images of waste items for classification.
LED Display	Provides real-time feedback to users.
Stepper Motor	Rotates the bin plate for waste sorting.
Servo Motor	Tilts the bin plate to drop waste into the correct section.
Ultrasonic Sensors	Detects bin fullness by measuring waste levels.
Sense HAT (LED Matrix)	Displays bin status (turns red if any section is full).
Power Supply Unit	Provides stable power to all components.

Each Raspberry Pi is responsible for specific subsystems, ensuring parallel processing and modular testing.

4.2 Hardware Subsystems

4.2.1 Object Detection and User Feedback (RPi1)

- Camera Module: Captures images of waste items placed in the bin.
- LED Display (I2C): Displays classification results and system status messages.
- Local Database (SQLite): Stores waste classification results for offline use.
- Wi-Fi Module: Connects to the backend server for real-time updates.

Workflow:

1. User places an item in the bin.
2. The camera captures an image and sends it to RPi1 for classification.
3. Classification results are displayed on the LED screen and sent to RPi2 and Firebase

4.2.2 Waste Sorting Mechanism (RPi2)

- Stepper Motor (GPIO/I2C): Rotates the bin plate 360° to align with the correct bin section.
- Servo Motor (PWM): Tilts the plate to drop waste into the correct section.

- Motor Driver Board: Provides power and control signals to the motors.
- WebSocket Client: Listens for sorting instructions from the RPi1

Workflow:

1. RPi2 receives sorting instructions from RPi1.
2. The stepper motor rotates the bin plate to the correct section.
3. The servo motor tilts the plate, dropping the waste into the correct section.

4.2.3 Bin Full Monitoring System and Web Interface Control (RPi3)

- Ultrasonic Sensors (GPIO): Measure waste levels in each section.
- Sense HAT LED Matrix (I2C): Displays bin status (red when any bin is full).
- Wi-Fi Module: Sends bin status updates to Firebase.
- Web Server Controller: Manages real-time updates between Raspberry Pis and web dashboard.

Workflow:

1. The ultrasonic sensors continuously measure waste levels.
2. If a bin section is full, the Sense HAT LED matrix turns red.
3. The bin status is sent to the backend server, triggering an alert to the manager.

4.3 Power Supply and Connectivity

Each Raspberry Pi and its connected components require a stable power supply. The power system consists of:

- 12V DC Adapter: Powers the motors and motor driver board.
- 5V USB Power Supply: Powers the Raspberry Pis and sensors.
- Voltage Regulators: Ensure stable voltage for different components.

All Raspberry Pis communicate via Wi-Fi using WebSockets for real-time data exchange with Firebase and the Web Interface, while local components use GPIO, I2C, UART, and CSI protocols for hardware communication.

4.4 Schematic for Raspberry Pi 1 (RPi1) – Object Detection & User Feedback

4.4.1 Components in the Schematic:

1. Raspberry Pi 1 (RPi1)
2. Raspberry Pi Camera (CSI Interface)
3. I2C LED Display
4. Wi-Fi Module (built-in)
5. 5V Power Source (USB-C Adapter)
6. J1 proximity sensor
7. QRB1114 reflective object sensor
8. TCS3200 Color Sensor

4.4.2 Connections:

1. Camera (CSI) → Raspberry Pi 1 (CSI Port)(built in connection)
2. I2C LED Display (SDA/SCL) → RPi1 GPIO Pins 2 (SDA) & 3 (SCL)
3. Power (5V & GND) → RPi1, LED Display, Camera Module
4. J1 proximity sensor VCC → 3.3V, VOUT → Gpio pin GND → GND
5. TCS3200 Color Sensor This sensor has 8 pins:
    a. VCC → 3.3V or 5V,GND → GND
    b. S0 & S1 → GPIO4 & GPIO5 (to set frequency scaling),S2 & S3 → GPIO6 & GPIO13 (to select color filter)
    c. OUT → Any GPIO pin with PWM support (e.g., GPIO18),OE (Output Enable) → GND (to enable output)
6. QRB1114 reflective object sensor
    a. IR LED Circuit (Pins 1 & 2) Anode (A) → 5V (through a 220Ω resistor), Cathode (K) → GND
    b. Phototransistor Circuit (Pins 3 & 4) Emitter (E) → GND, Collector (C) → GPIO input pin (e.g., GPIO17) with a 10kΩ pull-up resistor

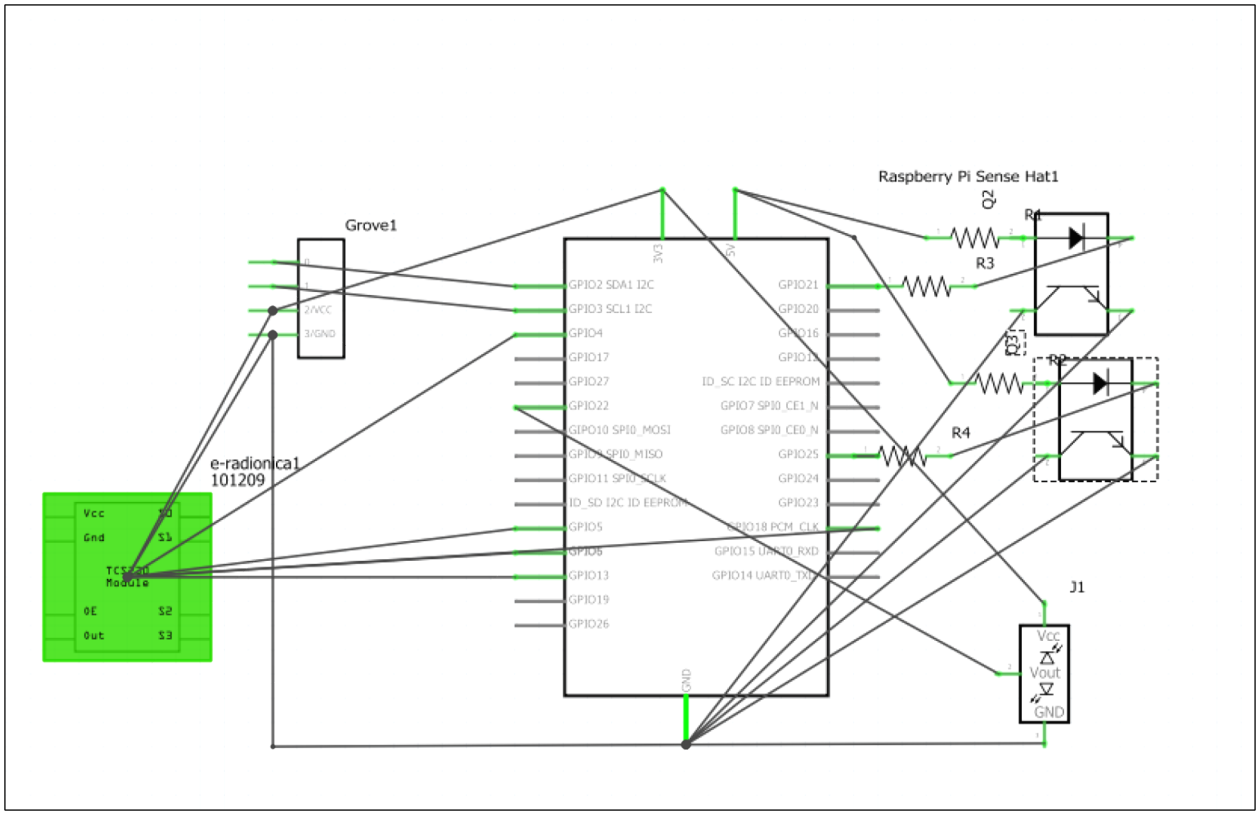


Figure 13: Circuit Schematic for RPi1 – Object Detection & User Feedback

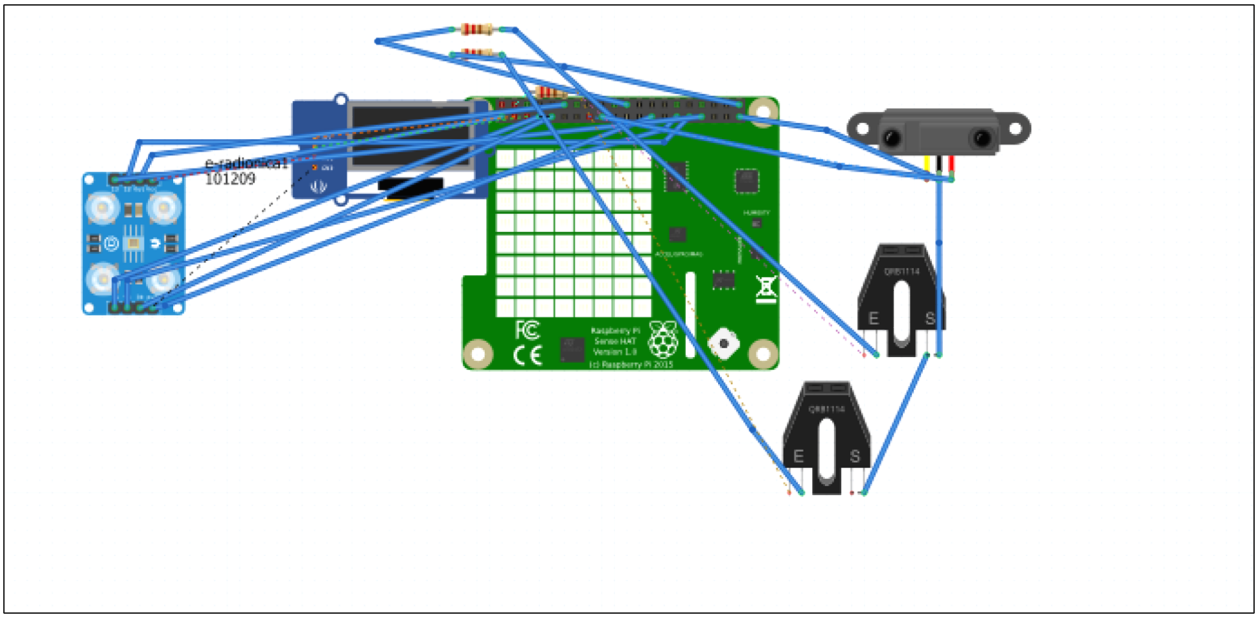


Figure 14: Physical Wiring Diagram for RPi1 – Object Detection & User Feedback

4.5 Schematic for Raspberry Pi 2 (RPi2) – Motor Control

4.5.1 Components in the Schematic:

1. Raspberry Pi 2 (RPi2)
2. Stepper Motor + Motor Driver (A4988/DRV8825)
3. Servo Motor (PWM Control, SG90 or MG995)
4. 5V & 12V Power Supplies

4.5.2 Connections:

1. Stepper Motor Driver (A4988/DRV8825):
    a. STEP & DIR pins → RPi2 GPIO
    b. VCC & GND → 12V Power
    c. Motor Coil Wires → Stepper Motor
2. Servo Motor (PWM Controlled):
    a. Control Signal → RPi2 GPIO (PWM Pin, e.g., GPIO18)
    b. Power (5V & GND) → External Power Supply
3. Power Connections:
    a. 5V USB Power → RPi2
    b. 12V Power → Motor Driver
    c. Common GND for all components

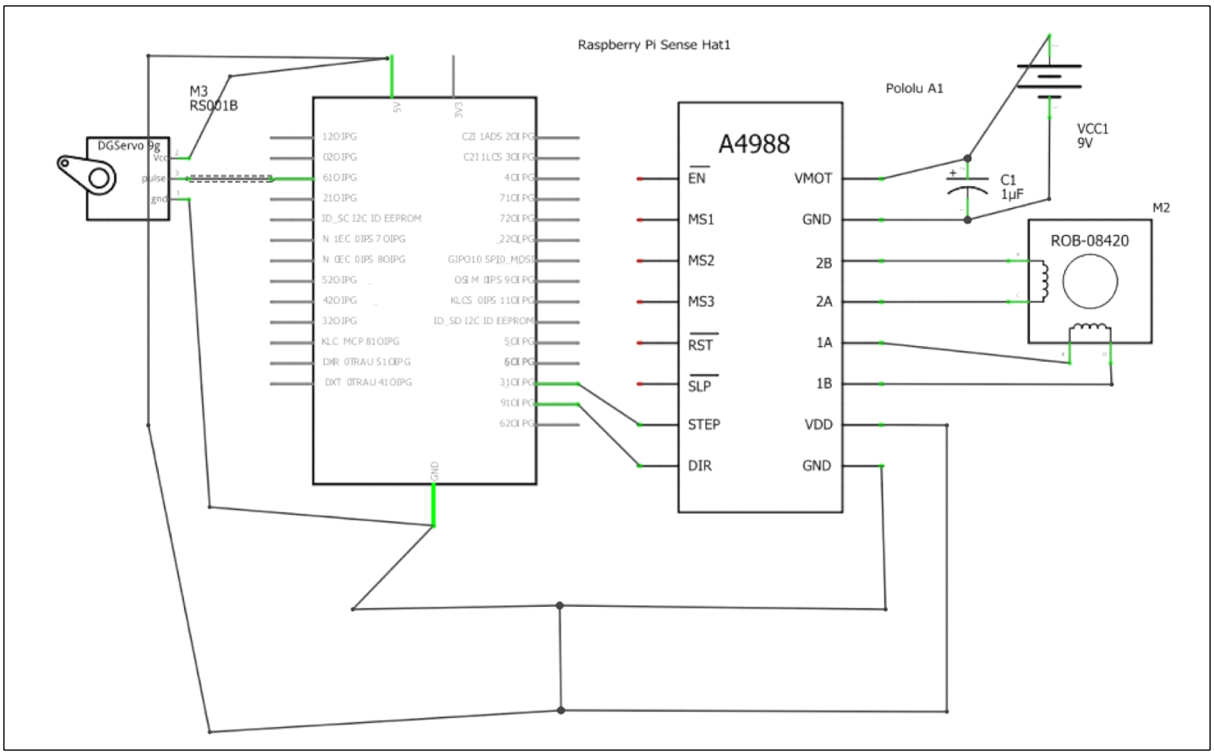


Figure 15: Circuit Schematic for Raspberry Pi 2 (RPi2) – Motor Control

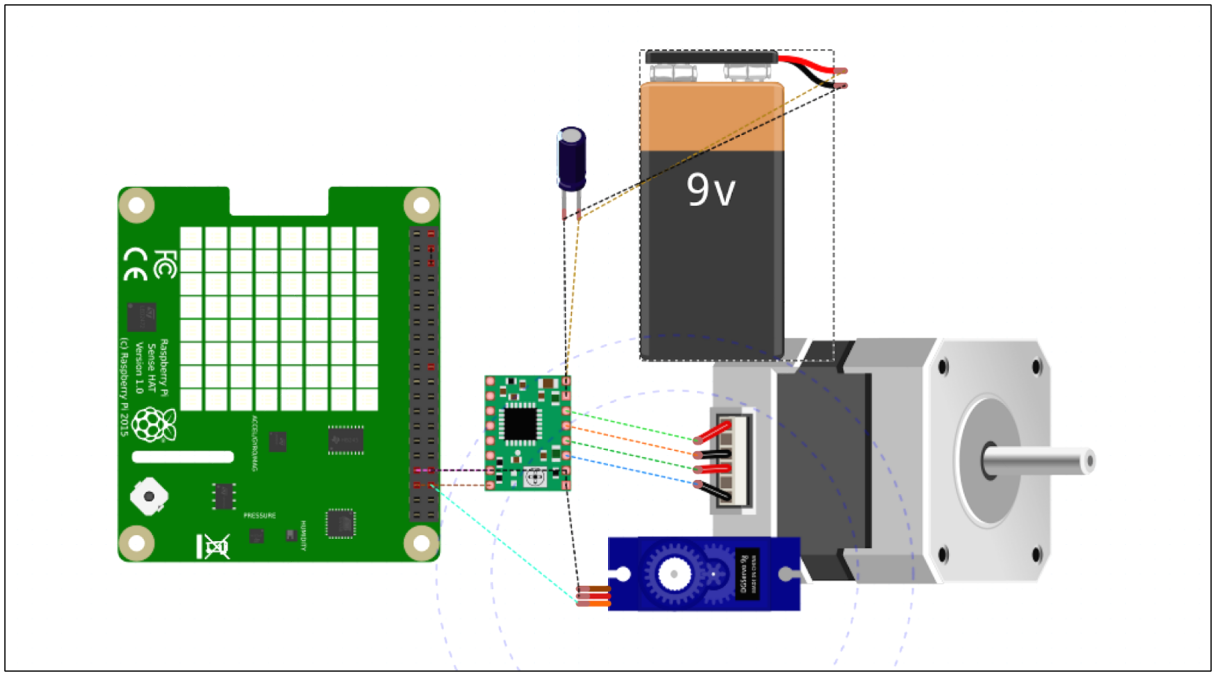


Figure 16: Physical Wiring for Raspberry Pi 2 (RPi2) – Motor Control

4.6 Schematic for Raspberry Pi 3 (RPi3) – Bin Full Monitoring

4.6.1 Components in the Schematic:

1. Raspberry Pi 3 (RPi3)
2. Ultrasonic Sensors (HC-SR04, x4 for each bin section)
3. Sense HAT LED Matrix (I2C Communication)
4. 5V Power Supply

4.6.2 Connections:

1. Ultrasonic Sensors (GPIO Controlled):
    a. Trigger Pins → RPi3 GPIO (e.g., GPIO5, GPIO6, GPIO13, GPIO19)
    b. Echo Pins → Voltage Divider → RPi3 GPIO Inputs
    c. Power (5V & GND) → Ultrasonic Sensors
2. Sense HAT LED Matrix (I2C Device built in connection):
    a. SDA/SCL → RPi3 GPIO2 (SDA) & GPIO3 (SCL)
    b. 5V Power & GND → RPi3

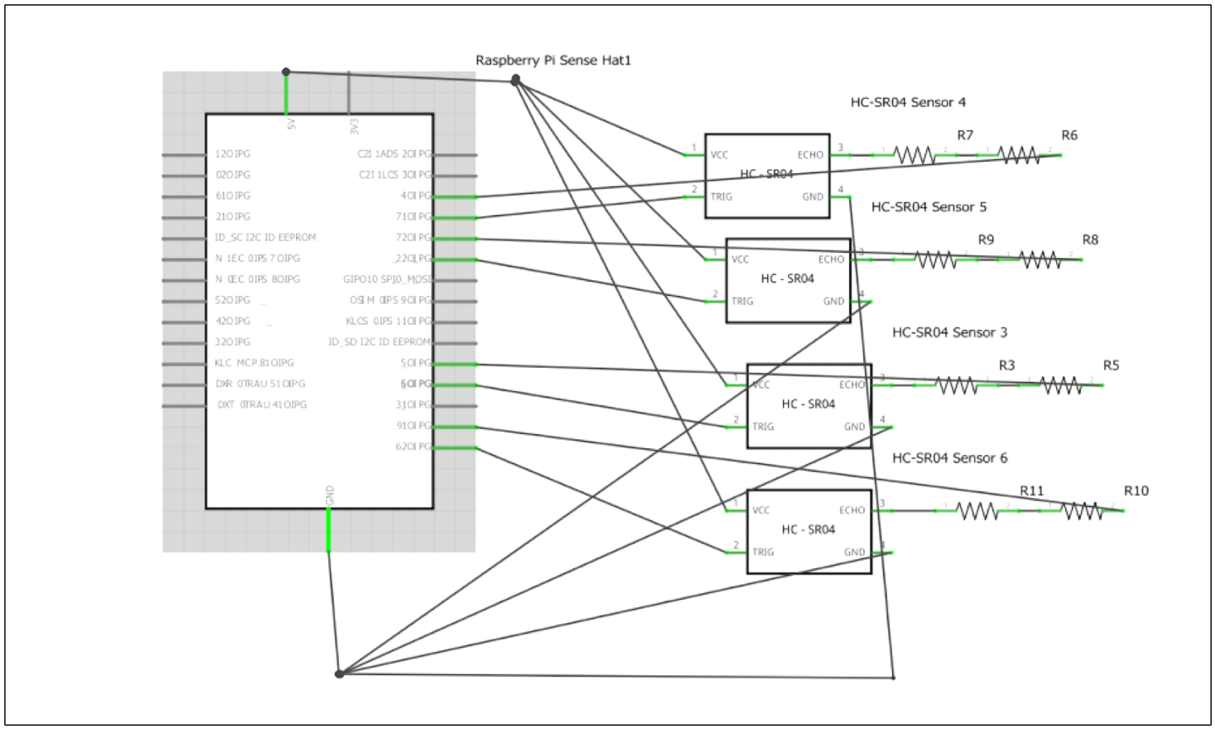


Figure 17: Circuit Schematic for Raspberry Pi 3 (RPi3) – Bin Full Monitoring

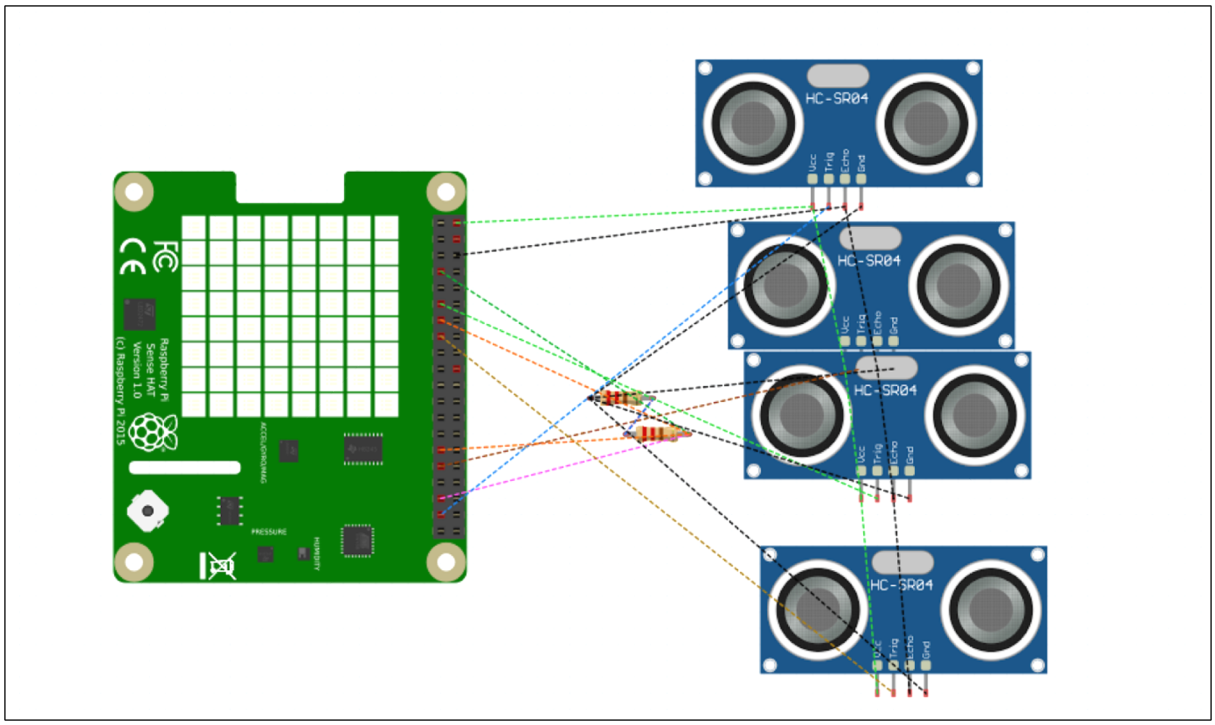


Figure 18: Physical Wiring Schematic for Raspberry Pi 3 (RPi3) – Bin Full Monitoring

4.7 Schematic for Raspberry Pi 3 (RPi3) - Bin Full Monitoring & Web Control

4.7.1 Components:

1. Raspberry Pi 3 (RPi3)
2. Ultrasonic Sensors (HC-SR04, x4 for each bin section)
3. Sense HAT LED Matrix (I2C Communication)
4. Web Server Controller
5. 5V Power Supply

4.7.2 Connections:

1. Ultrasonic Sensors (GPIO Controlled):
    - Trigger Pins → RPi3 GPIO (e.g., GPIO5, GPIO6, GPIO13, GPIO19)
    - Echo Pins → Voltage Divider → RPi3 GPIO Inputs
2. Sense HAT LED Matrix (I2C Device built-in connection):
    - SDA/SCL → RPi3 GPIO2 (SDA) & GPIO3 (SCL)
    - 5V Power & GND → RPi3
3. Web Server Controller:
    - Hosts web dashboard for real-time bin status updates.

5 GUI Design

The Smart Recycling Bin's graphical user interface (GUI) acts as the central point of interaction between users and the system. It provides real-time feedback on waste classification, bin status and data analytics to encourage better recycling habits. The interface is designed for accessibility via a dashboard-style web application.

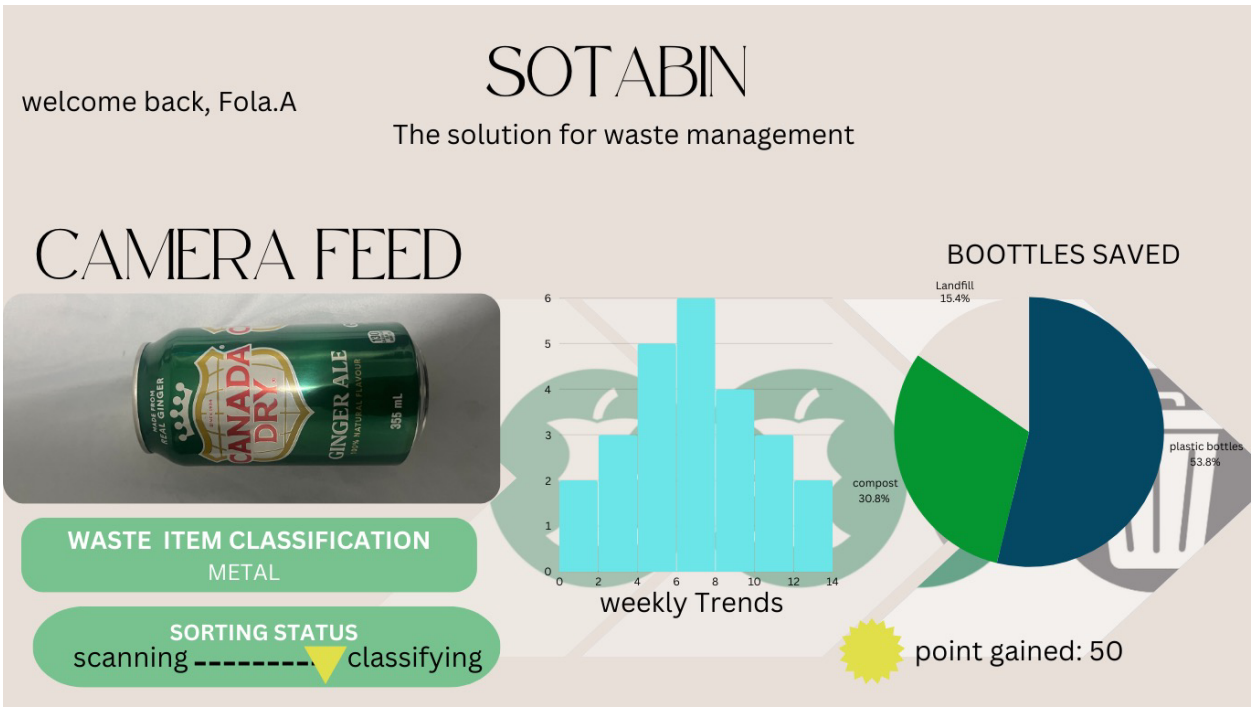


Figure 19: GUI mock-up for users

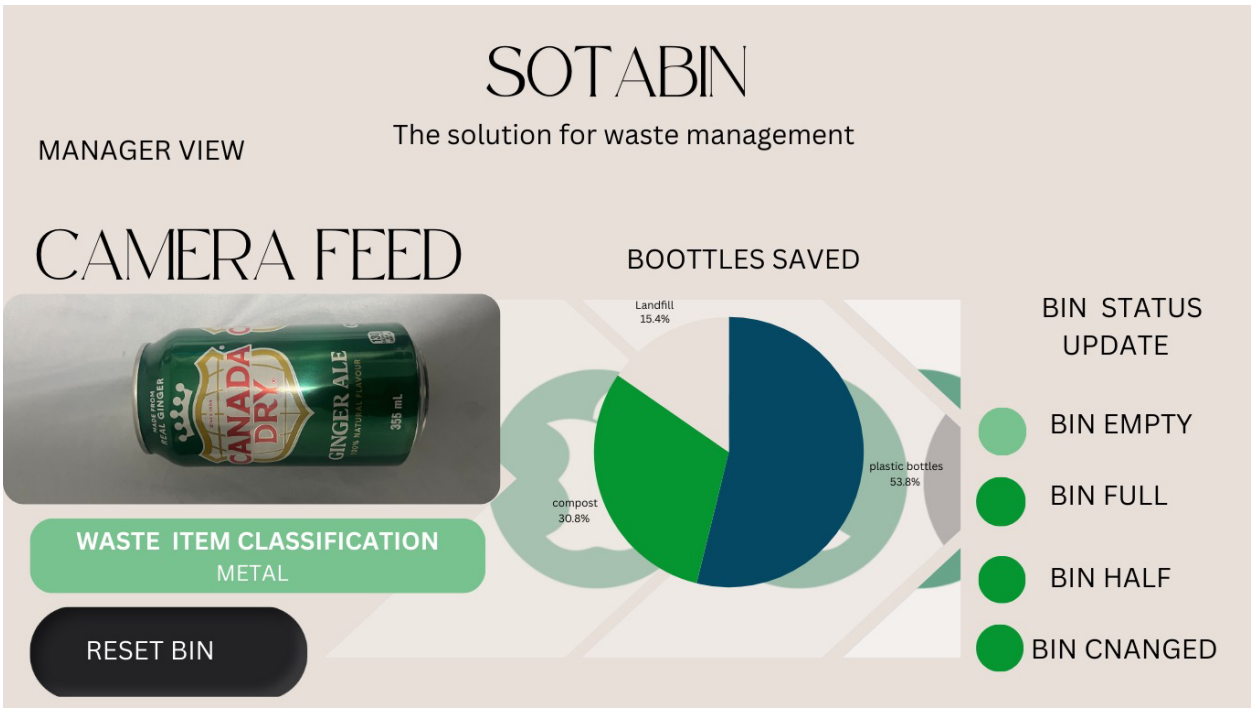


Figure 20: GUI mock-up for manager

5.1 Key GUI Functionalities

The GUI integrates multiple system components and presents data interactively. The key functionalities include:

- Real-time Waste Classification Feedback – Users receive instant classification results.
- Bin Status Monitoring – Users can track if a bin section is empty, half-full, or full.
- Sorting Status – Displays sorting animations based on motor movement.
- Manager Actions – Managers can reset bin full status and override sorting if needed.
- Data Insights & Reports – Provides historical trends and analytics on waste disposal patterns.

5.2 GUI Workflow

The system follows a structured workflow to ensure seamless communication between hardware nodes (RPi1, RPi2, RPi3) and the GUI:

1. User places waste in the bin.
2. Camera captures an image and sends it to RPi1 for classification.
3. RPi1 processes classification using computer vision and sensors, then:
    - Sends classification results to GUI via WebSockets.
    - Logs classification data in SQLite for local storage.
    - Sends sorting instructions to RPi2 via WebSockets.
4. RPi2 receives sorting instructions and:
    - Activates the stepper and servo motors to direct waste.
    - Updates GUI sorting animations via WebSockets.
5. RPi3 continuously monitors bin levels using ultrasonic sensors:
    - If a bin is full, RPi3 updates the GUI and triggers an alert.
6. Managers can send reset actions from the GUI, which:
    - Sends a WebSocket command to RPi3 to clear Sense HAT LED status.
    - Updates Firebase for record-keeping.

5.3 Table of Users/Roles

The system supports two primary roles, each with access to different functionalities:

Role	Description
User	General users who interact with the bin by disposing of waste. They receive real-time classification feedback and can monitor bin status through the LED display and web interface. Users also have access to waste disposal analytics, allowing them to view trends and track recycling habits.
Manager	Managers oversee system operations, including resetting bin full alerts, monitoring real-time bin status, and analyzing waste disposal trends via the analytics dashboard. They can also manage system configurations and handle manual overrides if needed.

5.4 GUI Message Flow and Communication

The GUI retrieves real-time updates from Firebase, while WebSockets are mainly used for sorting actions between RPis. The following structured messages are exchanged:

Message	Description
Waste Classification Request	Sent from RPi1 to Firebase upon waste disposal. The GUI fetches classification updates from Firebase.
Sorting Confirmation	Sent from RPi2 to Firebase, and the GUI retrieves updates for sorting animations.
Bin Status Update	RPi3 updates Firebase when a bin section is full. The GUI retrieves bin status from Firebase.
Manager Reset Action	Sent from GUI to Firebase, which then updates RPi3 to clear the full-bin alert**.**

5.5 GUI Enhancements and Future Considerations

The Smart Recycling Bin's GUI is designed for scalability, allowing future enhancements such as:

- Predictive analytics for waste disposal trends.
- Remote monitoring & control via mobile app integration.
- Integration with municipal waste management systems for large-scale deployments.

6 Test Plans

6.1 End-to-end Communication Demo Test Plan

The Sota Bin system is a distributed setup with three Raspberry Pis (RPi1, RPi2, RPi3) and a central web server for waste classification, sorting, and bin monitoring. The communication demo aims to verify that all nodes can effectively send and receive messages as defined in the Communication Protocol Table. It will validate the integration of hardware and software by ensuring functional interfaces and inter-node communication. The demo will use test scripts and simulators to mimic sensor inputs and actuator

responses, focusing on communication protocols like GPIO, I2C, CSI, and Firebase/WebSocket.

Test ID	Test Name/Description	Test Setup (including arguments or preconditions)	Expected Results	Actual Result
1	Validates communication between RPi 1 and connected hardware devices.	- The IR sensor detects approaching objects. - The Inductive Proximity Sensor determines if the object is metal. - The TCS3200 Color Sensor identifies the object's color. - The camera captures an image of the object.	RPi 1 processes the collected data and validations happen through the LED Display	RPi 1 processes the collected data and validations happen through the LED Display.
2	Validate communication between RPi 2 and connected hardware devices.	- Connect RPi 2 to the Servo Motor via GPIO and send an angle position command (integer value between 0 and 180). - Connect RPi 2 to the Stepper Motor via the A4988 driver and send a rotation command in step pulses.	The Servo Motor moves to the specified angle. The Stepper Motor rotates as expected based on the given step pulses.	The Servo Motor moves to the specified angle. The Stepper Motor rotates as expected based on the given step pulses.
3	Validate communication between RPi 3 and connected hardware devices.	- The ultrasonic sensor measures the distance to detect whether the bin is full. - The sensor sends a distance value (0 to 400 mm) to RPi 3. - RPi 3 processes the data and displays the bin status on the LCD screen.	-The measured distance is correctly displayed in millimeters. - RPi 3 successfully writes the bin status to the LCD screen. - The bin content status is accurately displayed.	- The measured distance is correctly displayed in millimeters. - RPi 3 successfully writes the bin status to the LCD screen. - The bin content status is accurately displayed.
4		a) Rpi1 sends classification data to	a) New entry to classification_data	a) New entry to classification_data

	Validates communication between Raspberry Pis, Firebase Cloud Service, and Web Interface	firebase. The python file rpi1_to_firebase.py handles this. The automation script for this is run_sota_bin.sh b) Rpi3 sends the bin status to firebase. The python file rpi3_to_firebase.py handles this. c) Web interface to firebase: When the manager clicks the reset button on the web interface, the manager's reset action is recorded in the firebase real time database. d) Firebase to RPi3: Once a reset action is recorded on firebase, RPi3 SenseHat is cleared The automation script for the 3 testcases above is run_sota_bin1.sh.	table is: {"item": "plastic", "bin": "2"} b) New entry to bin_status: {"bin": "3", "status": "Full"} c) New entry to manager_actions table : {"action": "reset", "bin": "3"} d) SenseHat turns RED initially and then clears	table is: {"item": "plastic", "bin": "2"} b) New entry to bin_status: {"bin": "3", "status": "Full"} c) New entry to manager_actions table : {"action": "reset", "bin": "3"} d)SenseHat turns RED initially and then clears
5	Validates communication between RPi1, RPi2, RPi3.	- Start WebSocket server on RPi2 and RPi1. - RPi1 sends waste classification JSON {"type": "plastic", "bin": "2"} to RPi2. - RPi3 sends Bin Full Notification JSON {"bin": "3", "status": "Full"} to RPi1.	- RPi1 sends classification data: {"type": "plastic", "bin": "2"} to RPi 2 - RP3 sends bin status data: {"bin": "3", "status": "Full"} to RPi1	- RPi1 sends classification data: {"type": "plastic", "bin": "2"} to RPi 2 - RP3 sends bin status data: {"bin": "3", "status": "Full"} to RPi1

6.2 Unit Test Demo Test Plan

6.2.1 Hardware Tests

Each piece of hardware will be tested individually to ensure proper functionality before integrating with the system. These tests will confirm that the sensors, motors, and display components perform as expected.

Test ID	**Student**	**Test Name/Description**	**Test Setup (including arguments or preconditions)**	**Expected Results**	**Actual Results**
HW-1	Uchenna Obikwelu	Camera Module (RPi1)	- The camera module captures images for waste classification. - A test script captures an image of a known object (e.g., paper, plastic) and validates object detection via OpenCV.	The image is successfully captured and recognized by the classification algorithm.	
HW-2	Dearell Tobenna Ezeoke	Color Sensor (TCS3200)	- The color sensor differentiates between compost and paper. - A script tests the sensor against pre-selected color samples and compares RGB values to expected values.	RGB values are correctly mapped to known samples.	
HW-3	Adeyehun Folahanmi	Metal Detector Sensor (RPi1)	- A metal can is placed near the sensor. - The sensor should detect metal and	The sensor correctly detects metal and updates its signal output.	

			return a signal change.		
HW-4	Adeyehun Folahanmi	IR Reflective Sensor (RPi1)	- The IR sensor detects material classification by reflection strength. - Test with a sample dataset of objects and verify the sensor's classification output.	Sensor correctly differentiates reflective (plastic/metal) and non-reflective (paper/organic) materials.	
HW-5	Tobiloba Ola	Servo Motor (Tilting Plate)	- A command sets the motor to a 45-degree tilt. - The physical movement is measured to ensure proper bin placement.	The servo motor tilts to the correct angle.	
HW-6	Tobiloba Ola	Stepper Motor (Rotating Plate)	- A command rotates the plate 90 degrees. - The rotation is visually verified with bin placement.	The stepper motor rotates correctly.	
HW-7	Emeka Anonyei	Ultrasonic Sensors (Bin Full Detection)	- Objects are placed at known distances. - Sensor readings are compared to expected values.	Correct bin fill levels are detected and updated.	
HW-8		LED Display	- Commands are sent to display status messages. - Visual verification of correct message output.	The LED displays the correct messages at the appropriate time.	

6.2.2 Software Tests

These tests ensure that the software modules process data correctly and respond appropriately to system events.

Test ID	Student	Unit Under Test	Software Test Description	Test Setup (including arguments or preconditions)	Expected Results	Actual Results
SW-1	Uchenna Obikwelu	Object Detection (RPi1)	- Uses OpenCV for waste classification. - A test script inputs known images (e.g., paper, plastic) and asserts correct classification.	Classification function is called with test images.	The system correctly classifies objects based on image input.	
SW-2	Adeyehun Folahanmi	Motor Control (RPi2)	- Sends commands to move the servo and stepper motor. - Asserts correct state transitions.	The motors receive and execute position/rotation commands.	Motors perform the expected movements.	
SW-3	Dearell Tobenna Ezeoke	Bin Full Monitoring (RPi3)	- Processes ultrasonic sensor data. - Simulates distance readings and verifies correct status updates.	Test distances are fed into the system.	Bin levels update accurately in the database.	
SW-4	Emeka Anonyei	Web Server (Firebase)	- Handles data exchange via Firebase. - Tests sending and receiving	Mock data is uploaded and retrieved from Firebase.	Firebase correctly stores and retrieves classification data.	

			classification data.			
SW-5	Tobiloba Ola	GUI (Frontend)	- Displays real-time feedback and user interaction. - Uses a React testing library to simulate UI interactions.	Simulated user interactions are performed.	UI updates reflect real-time system status.	
SW-6	Tobiloba Ola	Local Database (SQLite)	- Logs classification results. - Tests insert and retrieve operations with a mock dataset.	Database operations (insert, query) are executed.	Data integrity is maintained, and results match expectations.	

6.2.3 Database Integrity Tests

These tests ensure that the system correctly logs and manages data.

Test ID	**Student**	**Database Under Test**	**Test Description**	**Test Setup (including arguments or preconditions)**	**Expected Results**	**Actual Results**
DB-1	Uchenna Obikwelu	RPi1 Sensor Logs DB	- Ensures sensor readings are logged correctly. - Inserts mock sensor data and queries the database.	Sensor data entries are verified.	Data integrity is maintained.	
DB-2	Adeyehun Folahanmi	RPi2 Classifications DB	- Ensures classifications are stored correctly. - Inserts and retrieves classification results.	Database maintains correct waste sorting data.	Data is correctly stored and retrieved.	

DB-3	Dearell Tobenna Ezeoke	RPi3 Bin Status DB	- Ensures bin status updates are handled correctly. - Simulates bin full and reset actions.	Database reflects real-time bin status updates.	Data is updated correctly.	
DB-4	Emeka Anonyei	Firebase Real-Time Database	- Ensures Firebase updates reflect system status. - Simulates data upload and retrieval operations.	Firebase accurately reflects the latest system state.		

6.3 Final Demo Test Plan

For the final demo, the following scenarios will be tested to demonstrate that each functional requirement of the system has been met. Each test case ensures the system components function correctly under real-world conditions.

Test ID	**Test Name / Description**	**Requirement**	**Test Setup (including arguments or preconditions)**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
FD-1	Object Detection & Classification	The system must classify waste accurately using computer vision.	- Place known objects (paper, plastic, metal) under the camera. - Run the classification algorithm.	The system correctly classifies each object type.		
FD-2	Sorting Mechanism (Servo & Stepper Motor)	The sorting system must direct waste to the correct bin.	- Issue a classification result (e.g., "plastic"). - Observe servo and stepper motor behavior.	The motor moves correctly to place the object in the designated bin.		

FD-3	Bin Status Updates (Half/Full)	The system must detect bin fill levels and update status.	- Place test objects inside the bin. - Observe sensor readings at different levels.	The bin level updates to "Half" or "Full" as expected.		
FD-4	Sense HAT Notification	The system must visually indicate when a bin is full.	- Overfill a bin. - Observe the Sense HAT display.	Sense HAT turns red when a bin is full.		
FD-5	LED Display Status Updates	The LED must display sorting and bin full messages.	- Trigger sorting and full bin conditions. - Observe LED messages.	The correct status messages appear at the right time.		
FD-6	Real-Time Updates to Web Interface	The web interface must reflect classification and bin status updates in real time.	- Change bin status. - Add a classification result. - Observe the web dashboard.	The web interface updates in real time with new classification and bin statuses.		
FD-7	Remote Manager Reset	A manager must be able to reset a bin full alert remotely.	- Click "Reset" in the web interface. - Observe Sense HAT and database update.	The bin full alert clears, and the Sense HAT resets.		
FD-8	Database Integrity	The database must correctly log and retrieve system data.	- Insert sample sensor and classification data. - Retrieve and verify entries.	The database maintains data integrity, with correct inserts and queries.		
FD-9	System Resilience (Power Cycle Test)	The system must recover correctly after a reboot.	- Reboot the Raspberry Pi devices. - Check if the system	The system reboots and resumes operation		

			resumes normal operation.	without errors.		

Frameworks & Preparations

Testing Tools:

- Python Unittest/Pytest for automated validation.
- Firebase Emulator for cloud database testing.
- Mock Sensor Data for simulated input values (used for edge cases, stress tests, and fallback if sensors fail).

Test Setup:

- All Raspberry Pis connected and running.
- Web interface available and accessible.
- Physical test objects (plastic, metal, paper, compost) available for classification tests.
- Sense HAT and LED display verified for visual notifications.
- Database setup completed with test entries for validation.

7 Project Update

Our team has made great progress in advancing the Sota Bin project. We have successfully obtained all electrical components required for the system, including sensors, motors, and Raspberry Pis. However, we have not yet acquired the physical bin structure, which is necessary for final integration.

Despite this, we have established end-to-end connections, ensuring proper communication between sensors, Raspberry Pis, and the web interface. An end-to-end demonstration was successfully conducted, validating the data flow and real-time updates.

Challenges & Adjustments:

- Physical bin structure acquisition: Still pending. The team is currently evaluating possible fabrication options or alternative temporary structures for testing.
- Final system layout: The arrangement of components within the bin needs to be finalized.
- Calibration and fine-tuning: Although all components are functional, sensor calibration and motor control refinement are ongoing.

To maintain project momentum, we have reallocated tasks to focus on:

- Finalizing the physical structure while continuing software testing in parallel.

- Fine-tuning waste classification accuracy and bin full detection.
- Ensuring database synchronization and real-time feedback remains stable.

We remain on track for full system integration and expect to complete the remaining tasks within the next few weeks.

7.1 Project Milestones

Milestone Number	Milestone name	Description	Original Timeline	Status	Revised Deadline
1	**System Foundation & Hardware Setup**	Acquired all electrical components (sensors, motors, Raspberry Pis). Physical bin structure still pending. Completed initial component testing	February 15	In Progress	March 13
2	**Waste Classification System (RPi-1)**	Established communication between system components. Successfully conducted an end-to-end demonstration of classification, sorting, and bin monitoring.	February 20	Completed	N/A
3	**Sorting Mechanism & Control (RPi-2**	Integrated servo and stepper motors with sorting logic. Implemented basic sorting instructions from RPi1. Pending motor calibration for precise sorting.	February 23	In progress	March 15
4	**Web-based GUI & Local Database**	Developed a basic web-based GUI for system monitoring. Established real-time communication using WebSockets.	February 28	In progress	March 17

		Pending final database synchronization with Firebase.			
5	**Cloud Integration & Analytics**	Co Connected Firebase for real-time updates. Created initial database structure for waste classification storage. Pending refinement of data logging and analytics dashboard.	March 2nd	In progress	March 19
6	**Testing & Optimization**	Full integration of waste classification, sorting, and bin monitoring in progress. Pending sensor calibration and final testing with real waste samples.	March 15th	In progress	March 22