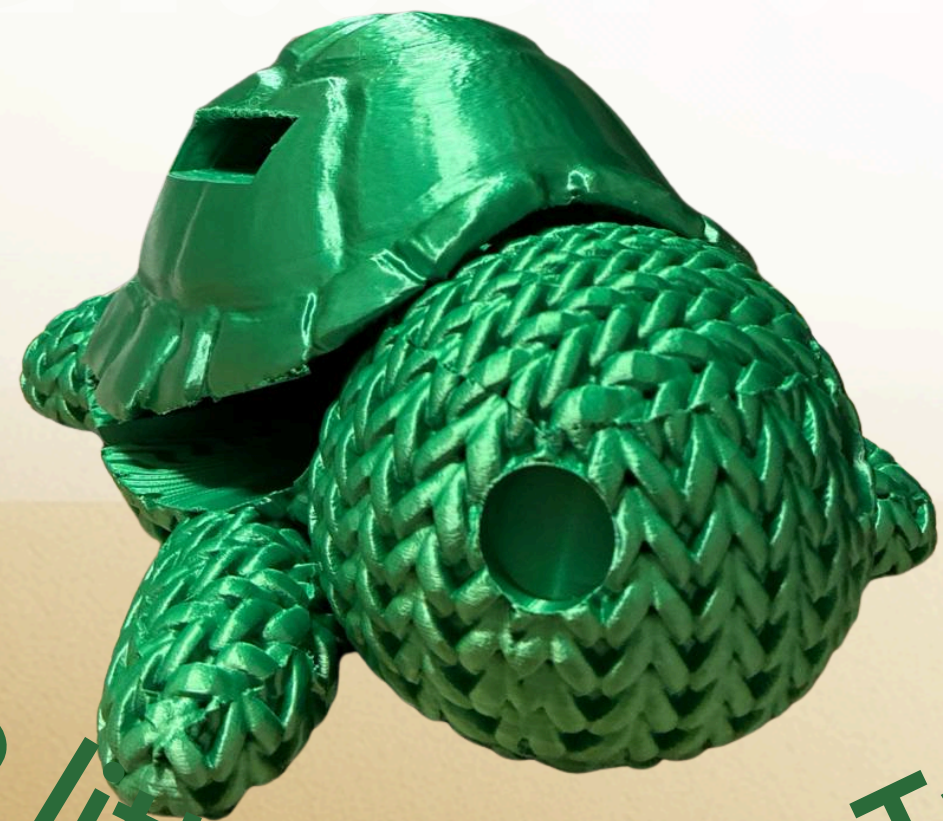


Littertle



No litter, Happy Turtle

Our Freindly Coastline Guardian

LITTERTLE IS A SMART TURTLE-SHAPED BIN THAT KEEPS BEACHES CLEAN BY DETECTING BOTH TRASH INSIDE THE BIN AND LITTER AROUND IT IN REAL TIME.

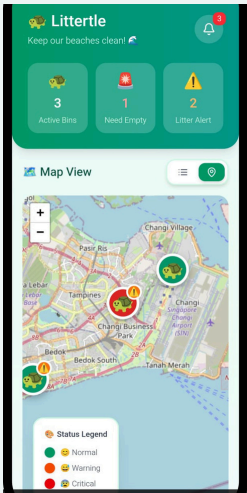
IT COMBINES IR SENSORS AND AN AI CAMERA WITH A RASPBERRY PI AND MOBILE APP SO N-PARK WORKERS CAN SEE WHICH BINS ARE FILLING UP, WHERE LITTER HAS BEEN SPOTTED, AND WHICH LOCATIONS NEED URGENT ATTENTION, REDUCING MANUAL PATROLS WHILE PROTECTING THE COASTLINE AND MARINE LIFE.

Software

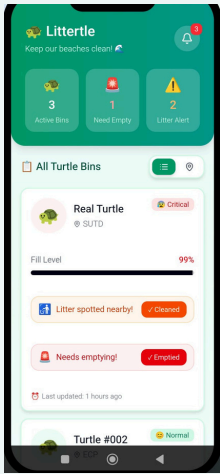
Littertle App



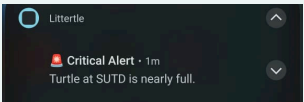
Used for N-Park workers to access turtle status and get notified when trash can is full and when litter is detected by camera



Map view



List view



Notification on phone

Hardware

Interior Housing

ESP 32

Controls the functionality of the IR sensors and sends data to the database



Raspberry Pi 4B

The main engine of the product , integrating the esp32, sensors, camera and app.



Camera Module

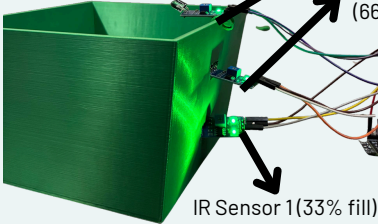
Uses YOLOV11 to detect trash on the coastline . Trash that is detected by the camera is sent as information to N-Park workers to go on site



Camera Module in turtle eye

IR Sensors Bin

Used for Sensing when Trash goes up to what level the trash is in the bin



IR Sensor 3 (Full bin)
IR Sensor 2 (66% fill)
IR Sensor 1(33% fill)

Problem Statement

HOW MIGHT WE IMPROVE REAL TIME MONITORING OF WASTE ACCUMULATION HAPPENING AT THE COASTLINE OF ECP SO THAT N-PARK CONSERVATION STAFF CAN RESPOND TO THREATS

Littertle Design Process

SC12 Group D

Design Goals

1. Create a trash monitoring system that allows N-Park workers to monitor trash in the East Coast Park more easily
2. Motivate Parkgoers to throw their trash in the trash can.

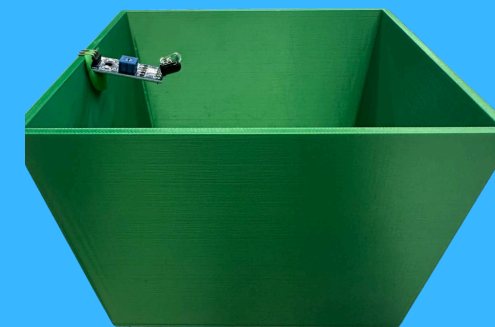
Features

1. Trash Can with IR sensors to detect when the trash can is full
2. Camera to detect if litter is spotted around the beach
3. App to relay all the data from the turtles

Potential Improvements

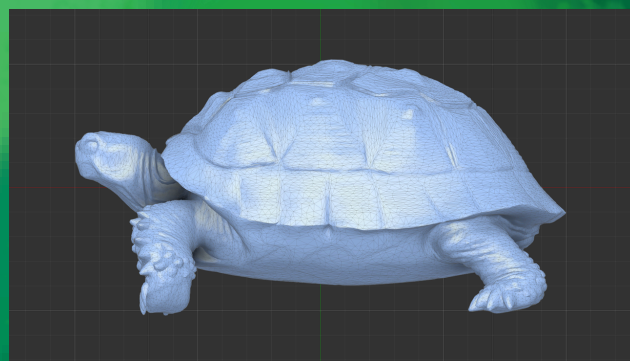
1. Add sounds and alerts to notification for higher risk situations
2. Make the system work better with multiple turtles
3. Add sound feedback for when someone throws trash into the turtle
4. Change of energy source to be solar

IR Sensor Module



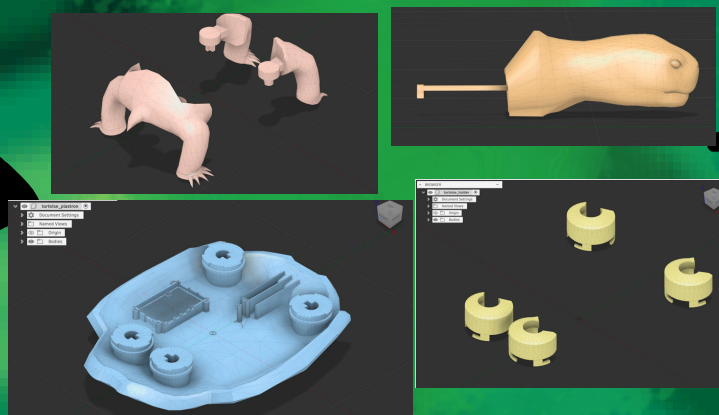
Turtle Shell Design Process

Model 1



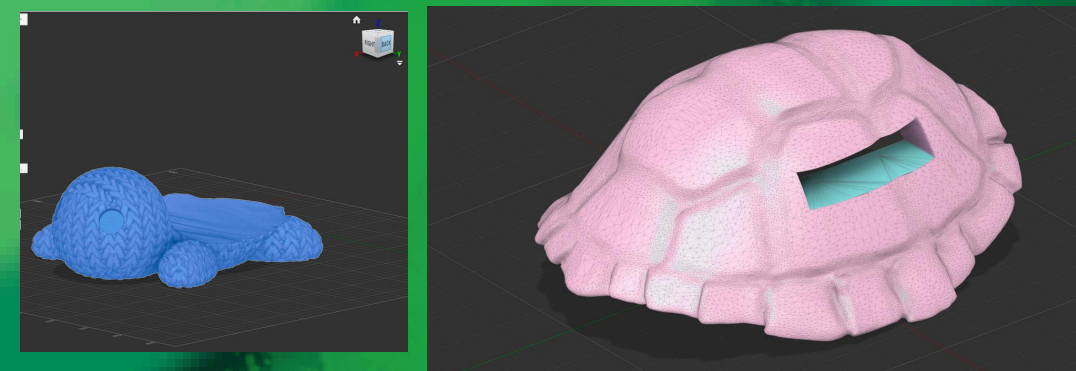
First Design Prototype, made large and sturdy to fit in the context

Model 2



Second Design Prototype, parts were redesigned to be scaled down but the structure was not perfect, causing difficulty in printing

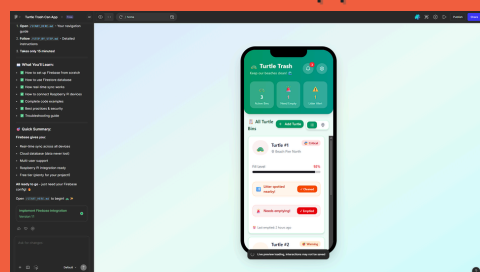
Model 3



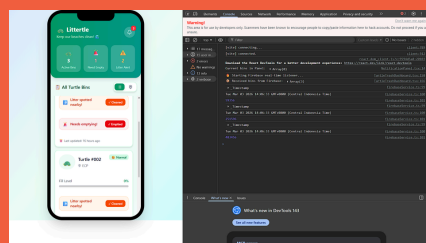
Final Design, Incorporating the **cute and appealing** design as well as scaling it properly for our prototype, including the hole for easier throwing of trash, rough body to be planted on sand more easily

App Design Process

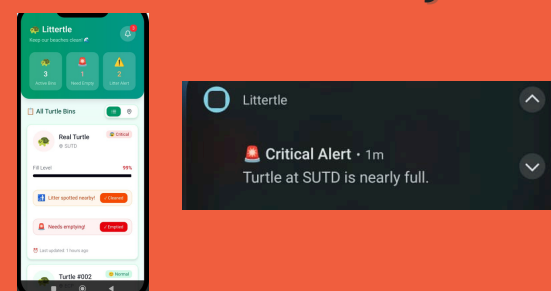
UI Designing Phase - Using Figma to generate and edit the UI design and test out the app functionality



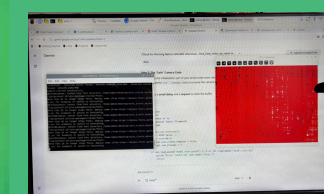
Database Testing Phase - Converting the Figma UI to a webapp useable on a computer, and integrating Firebase Database into the system



Mobile App Integration - Converting the webapp into an android studio project and downloading it on a phone, integrating mobile phone notifications through FCM



Camera Trash Detection Testing



First Attempt: Failed Connection to Camera



Second Attempt: Connection secured, less accurate detection



Final Program: Our current best detection system with the camera

Using YOLOv8, trained using 400 images of trash, accuracy 50-60%

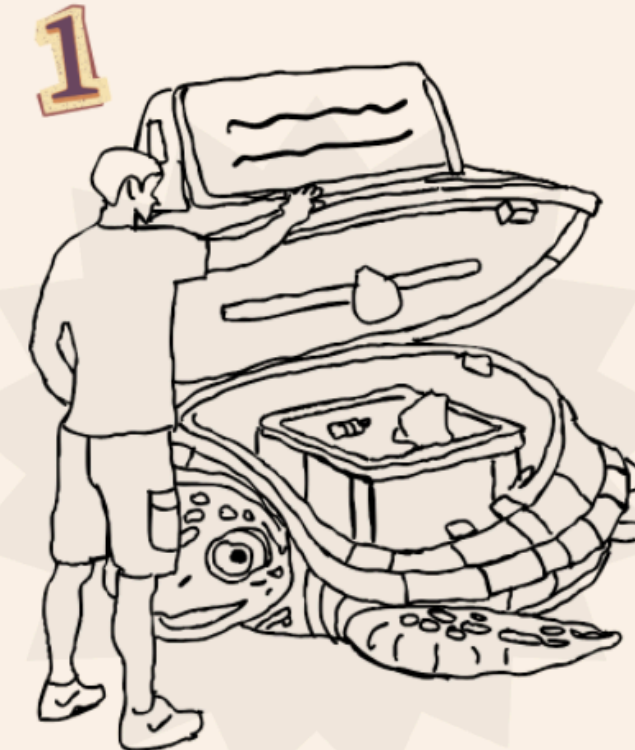
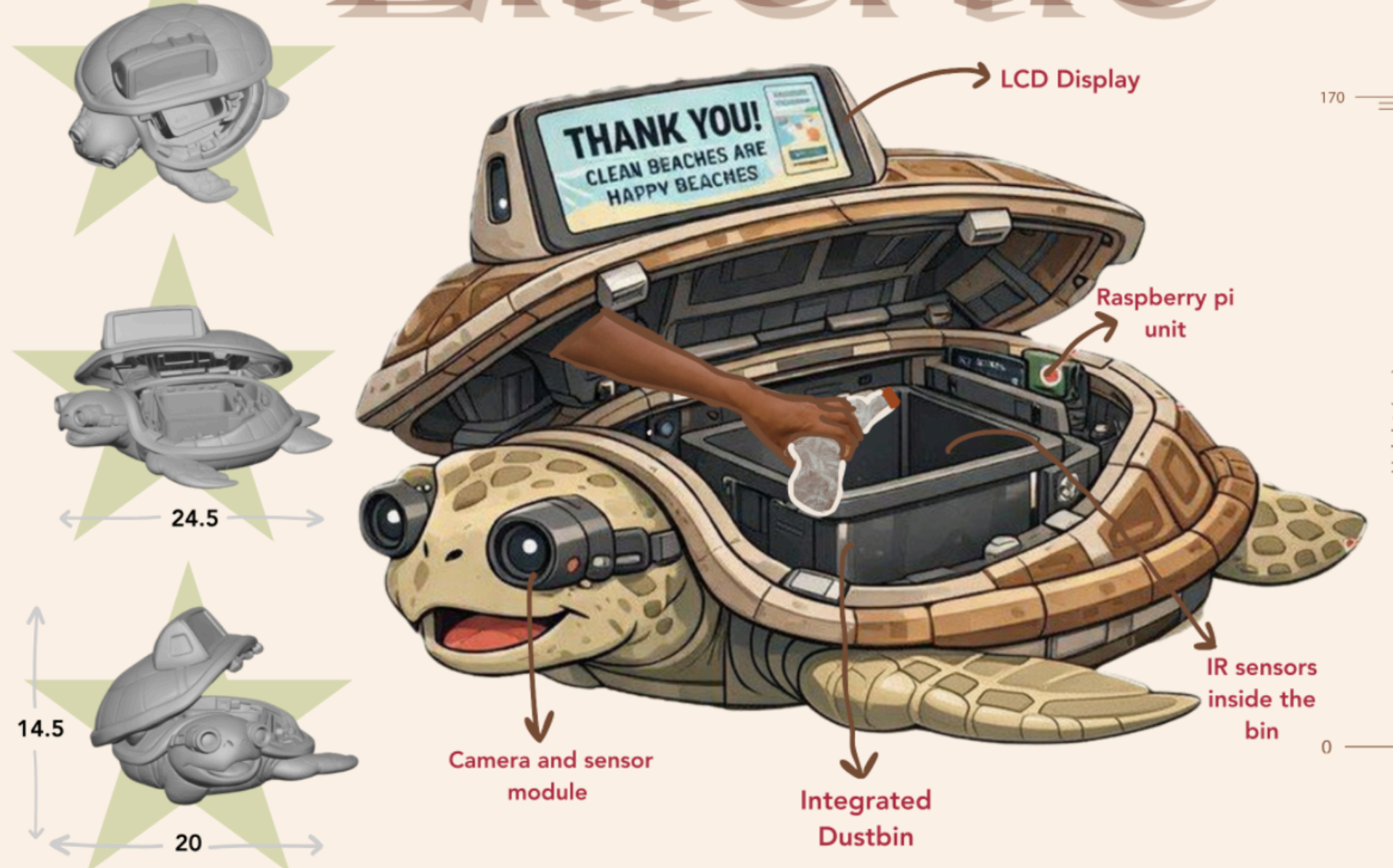
Utilizing YOLOv11 nano, accuracy increased to 80-85%

Littertle

No litter, happy turtle

Problem Statement

How Might We improve real time monitoring of waste accumulation happening at the coastline in ECP so that N-Park conservation staff can respond to threats.



When the camera detects clutters of trash, it will send out a signal to the N-park workers, with the locations of the trash highlighted inside the map

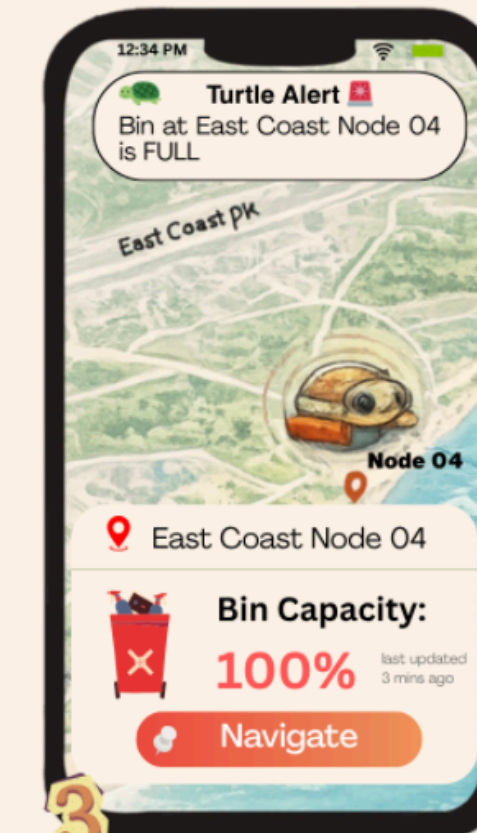
Littertle overlooking the beach



Camera Modules monitors the surroundings and utilises an AI recognition system to detect unwarranted objects.

Key Features

- **LCD Display:** Shows thank-you messages and live stats (e.g., litter collected today)
- **Raspberry Pi:** Controls sensors, display, and logs camera data
- **IR Sensors:** Detect litter deposition and bin-full status
- **Integrated Bin:** Stores collected waste under the shell
- **Front Camera ("Eye"):** Tracks usage and enables litter hotspot analysis



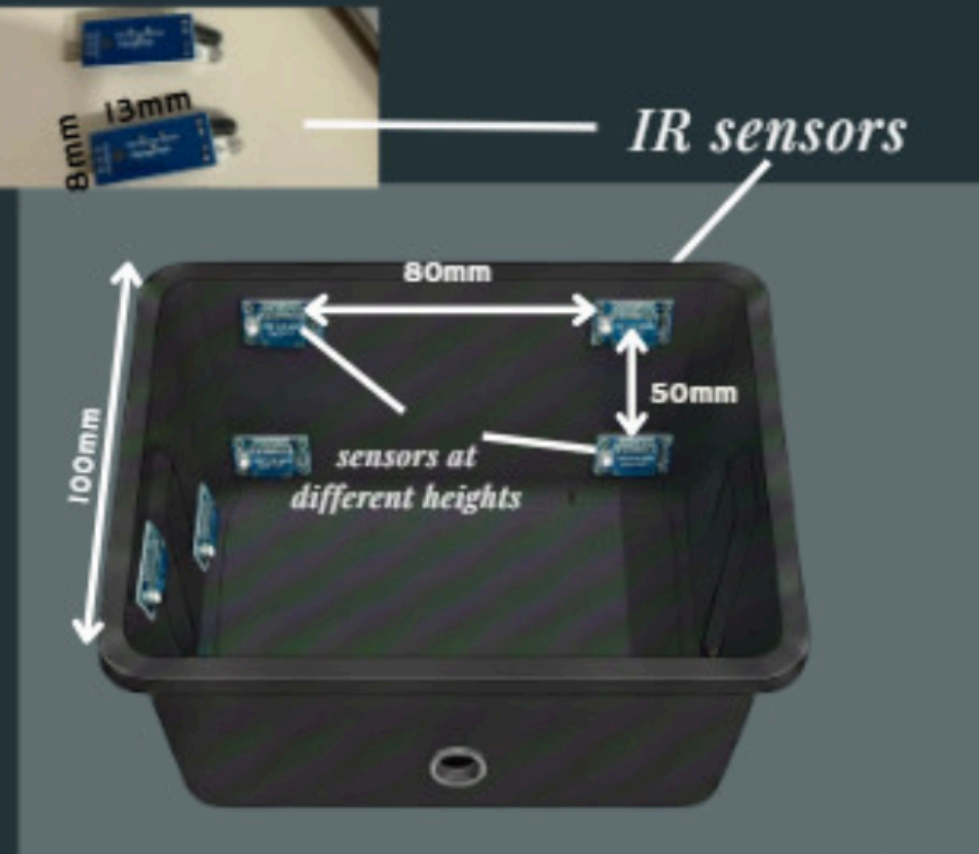
Authorities receive **real-time alerts** when bins are full, enabling immediate dispatch and preventing overflow.



Aggregated node data highlights trash **hotspots** via **heatmap**, allowing NParks officers to pursue action before conditions worsen.

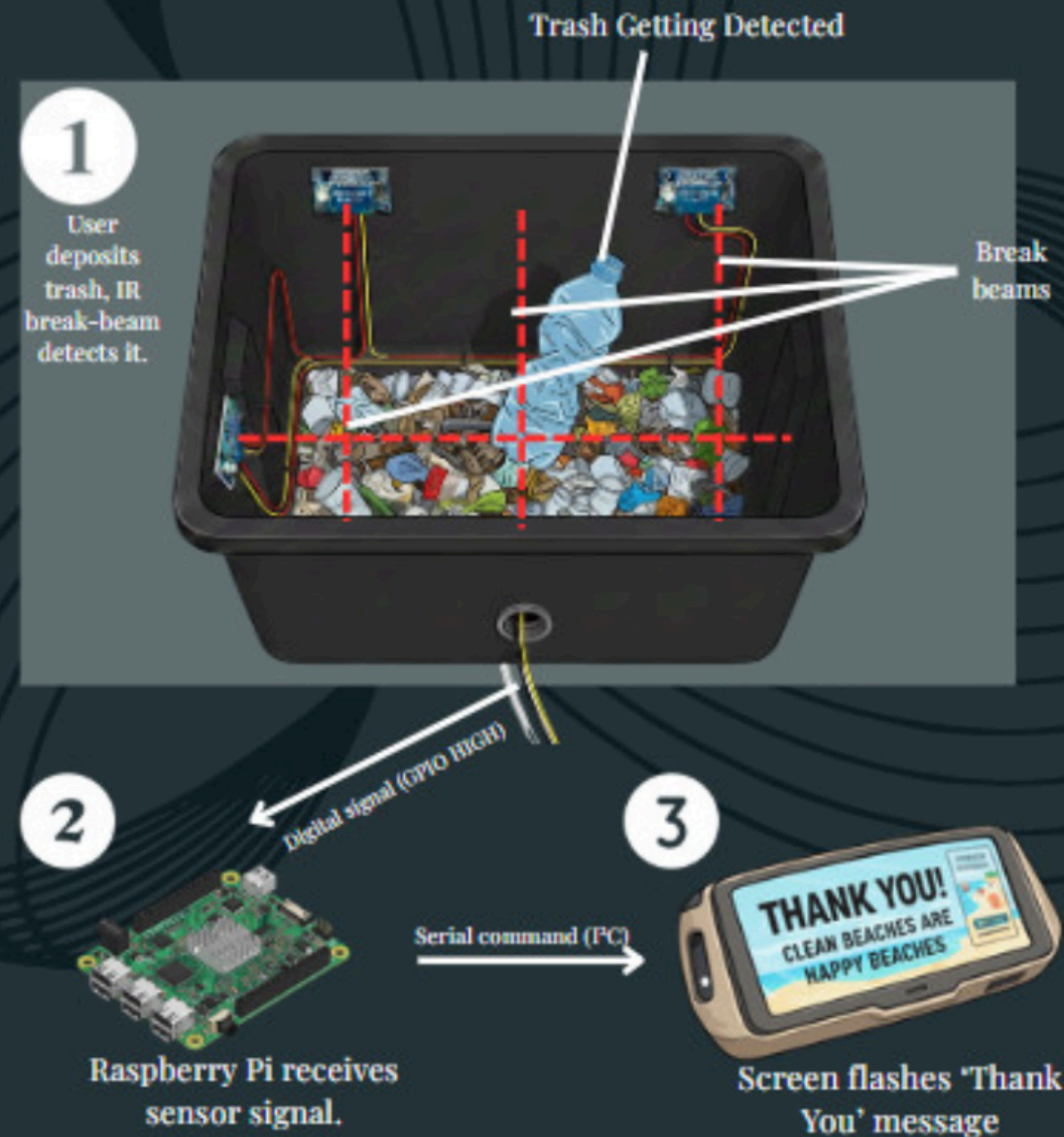
SUBSYSTEM: IR SENSORS

Base Specifications



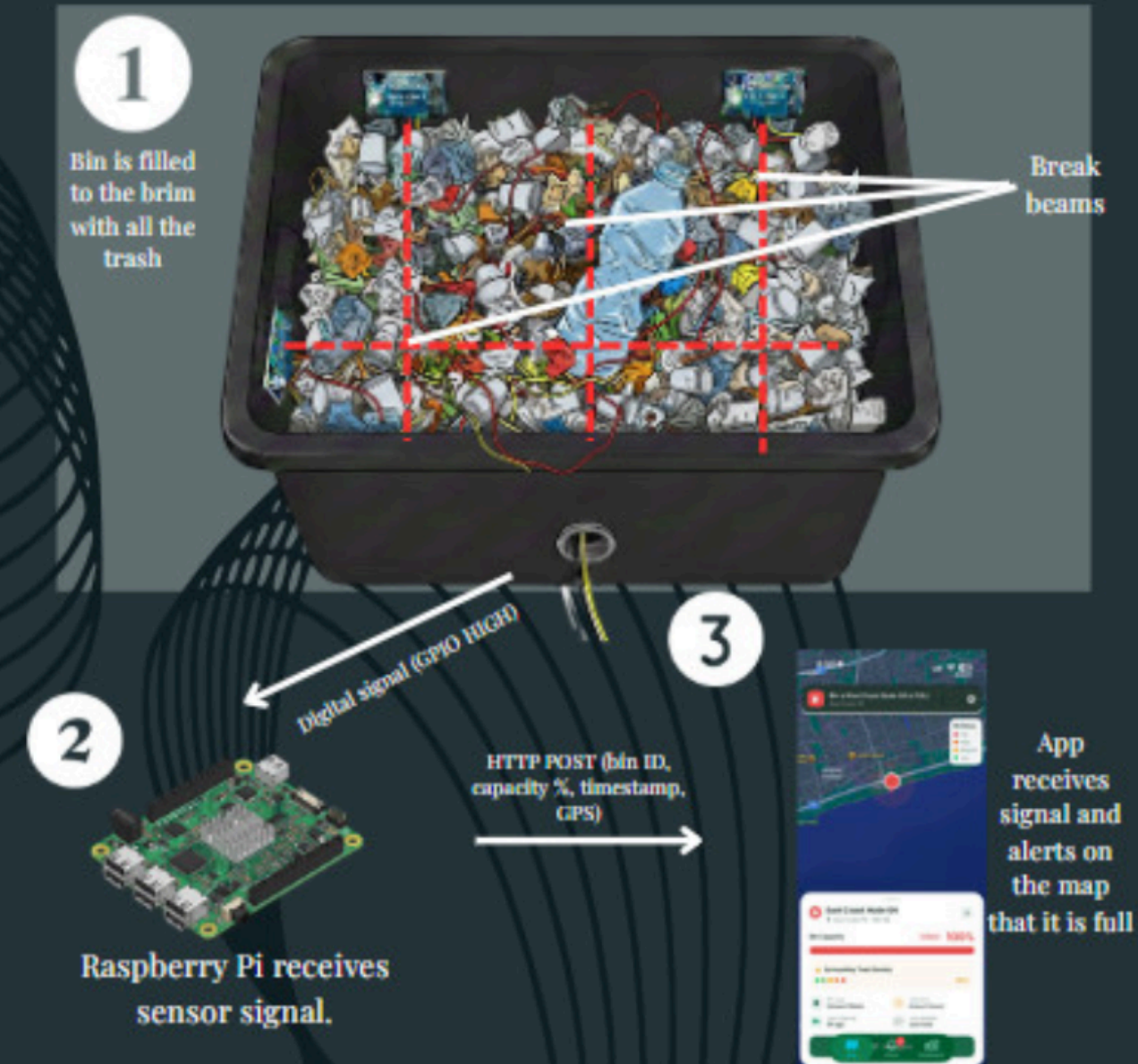
- Sensor model: MH Flying-Fish IR obstacle sensor (reflective type)
- Operating range: 3–80 cm
- Detection logic: Beam break >50ms = valid deposit (filters hand hovering)
- Fill-level thresholds: 50% (2/4 blocked), 75% (3/4)
- Alert trigger: ≥90% capacity sustained for 5 seconds
- Environmental: Indoor mounting, protected from direct rain; ambient IR rejection via modulated detection
- Sensors will be cased in a transparent box to mitigate direct contact with trash thrown

Scenario 1: Trash Thrown



- 1 IR sensors detect object breaking the light beam as trash enters the bin.
- 2 Raspberry Pi receives digital signal, validates deposit, and increments trash thrown in.
- 3 LCD instantly shows 'Thank You' message, confirming successful deposit to the user.

Scenario 2: Bin Filled



- 1 IR sensors continuously monitor fill-level; when trash blocks multiple beams, the system registers high capacity.
- 2 Pi detects threshold exceeded, processes data, and triggers alert protocol.
- 3 Mobile app displays bin location, capacity status (75%), and urgency level, enabling immediate dispatch for collection.

Detailing a sub-system

~ Chassis

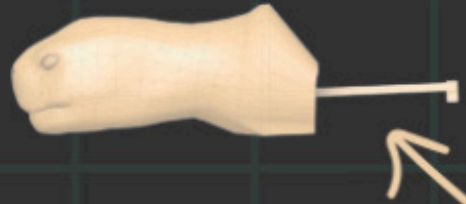
Role in the System

This subsystem forms the mechanical backbone of the turtle model. It provides structural integrity, houses electronic components (Camera, wiring, microcontroller), and enables modular assembly of the shell, legs, neck, and base platform. It ensures mechanical stability while maintaining aesthetic accuracy.

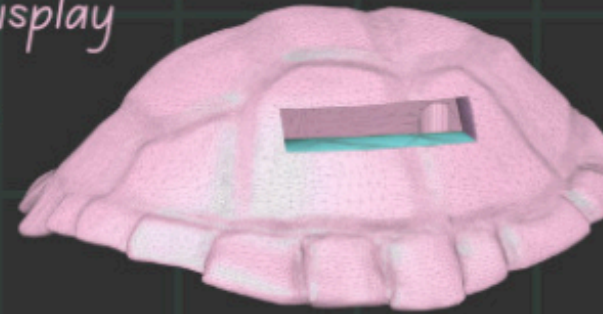
Key Requirements

- Provide rigid structural support for all mechanical and electronic components.
- Enable modular assembly (head, legs, shell top, shell bottom detachable).
- Integrate internal wire-routing channels for clean cable management.
- Maintain sufficient internal volume for electronics housing.
- Ensure printability (optimized for 3D printing tolerances).

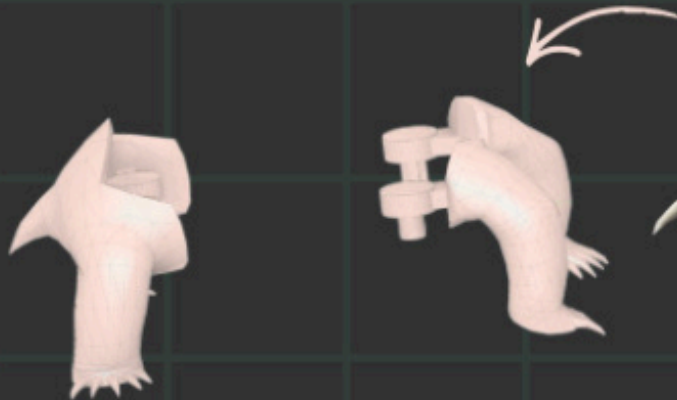
The head housing wires and camera module.



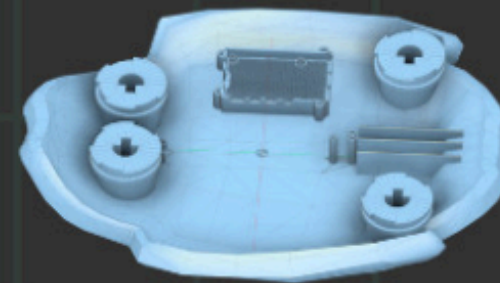
The Upper shell housing trash flap and overhead display



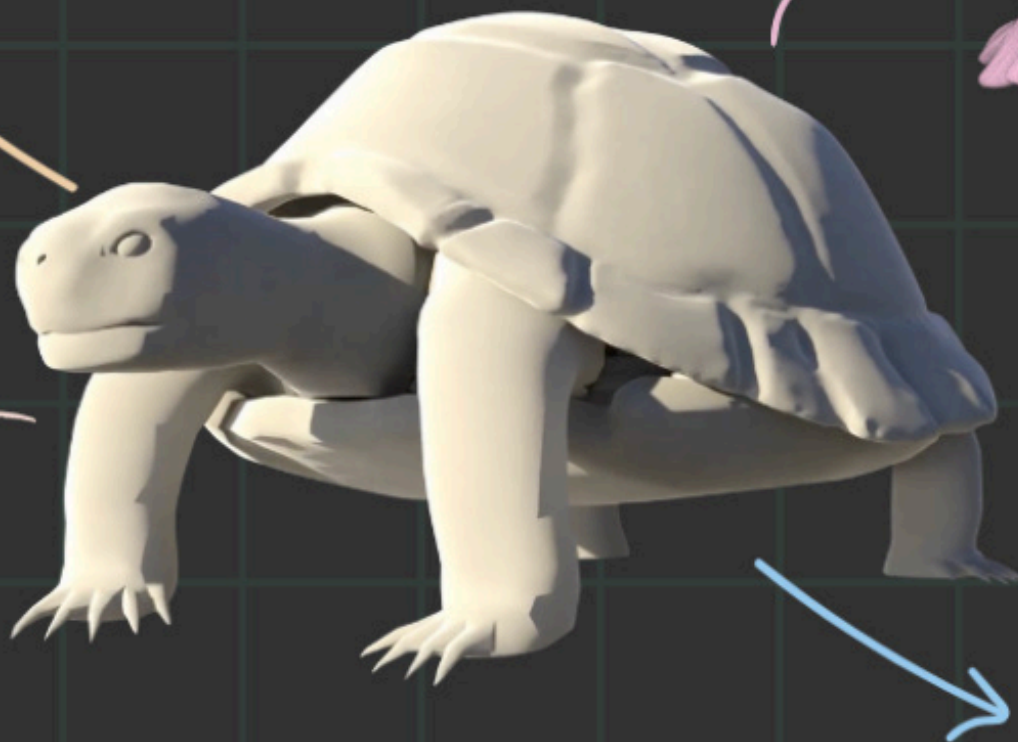
The limbs with screws for attachment into the main frame



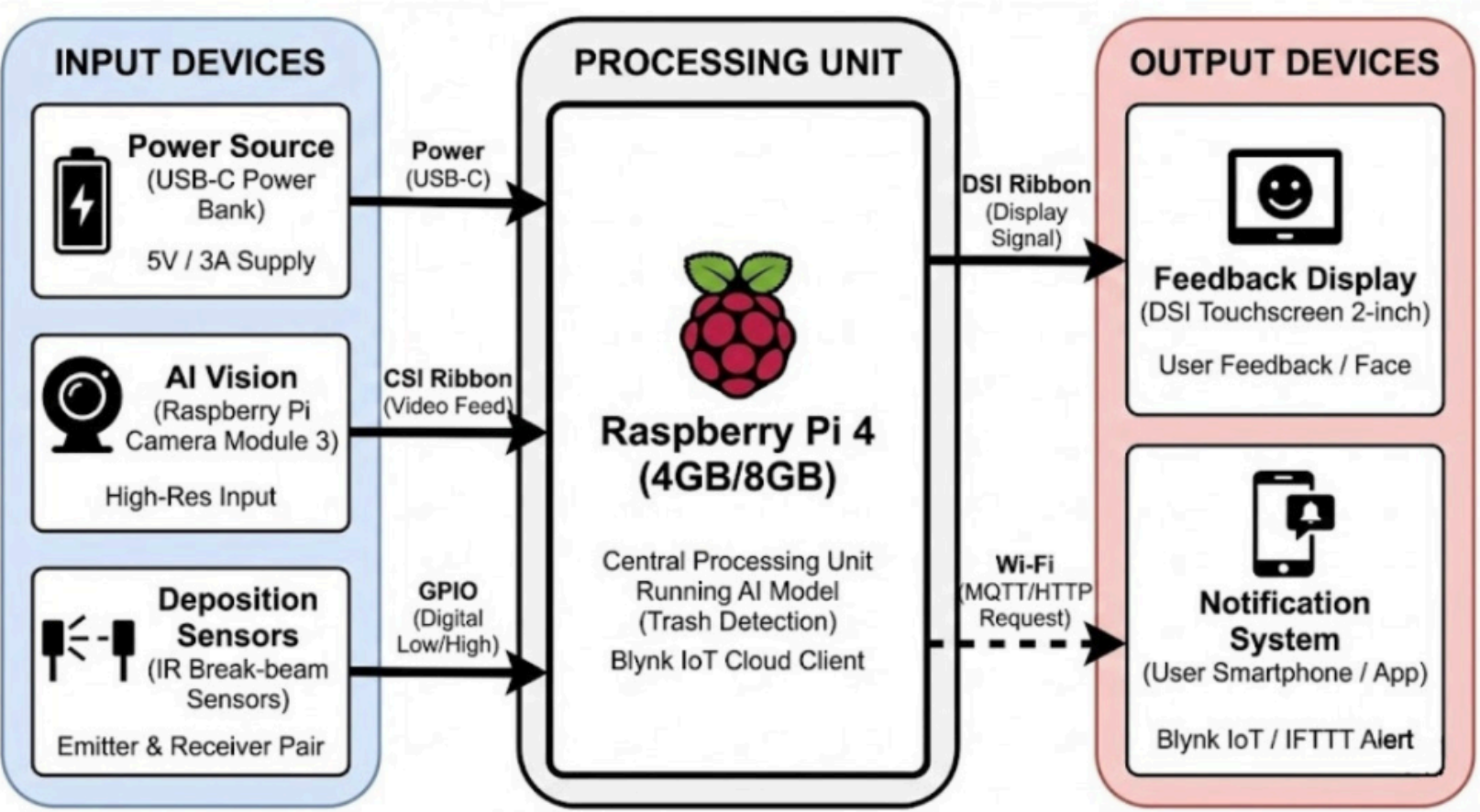
The lower shell housing RPI and the trash can.



1:4 Scale down ratio



FUNCTIONAL BLOCK DIAGRAM



SUBSYSTEM DESCRIPTION

The core of the system is a Raspberry Pi 4 Model B (4GB), chosen for its ability to handle multi-threaded OpenCV processes while simultaneously managing IoT connectivity.

Integration: The Pi interfaces with the Vision Subsystem via the high-bandwidth CSI port to minimize CPU overhead, while IR Break-beam sensors trigger interrupt-based events via GPIO to log trash collection.

Feedback Loop: User feedback is handled locally via a 2-inch DSI screen (for lag-free GUI rendering) and remotely via Wi-Fi to the Blynk IoT Cloud, enabling real-time status alerts on the user's smartphone.

Power: A dedicated 5V/3A rail ensures stable operation during peak processing loads.

Hardware Specifications:

- 1. Controller: Raspberry Pi 4 Model B (4GB)
- 2. Vision Input: Raspberry Pi Camera Module 3 (Wide-Angle, 12MP Sony IMX708 sensor)
- 3. Sensing Interface: 2x IR Break-Beam Sensors (3.3V Logic, Digital Input)
- 4. Display: 2-inch DSI Touchscreen (DSI-1 Interface, 24-bit color)
- 5. Power Management: 10,000mAh Lithium-Polymer Power Bank (Output: 5V/3A via USB-C)

Software Specifications:

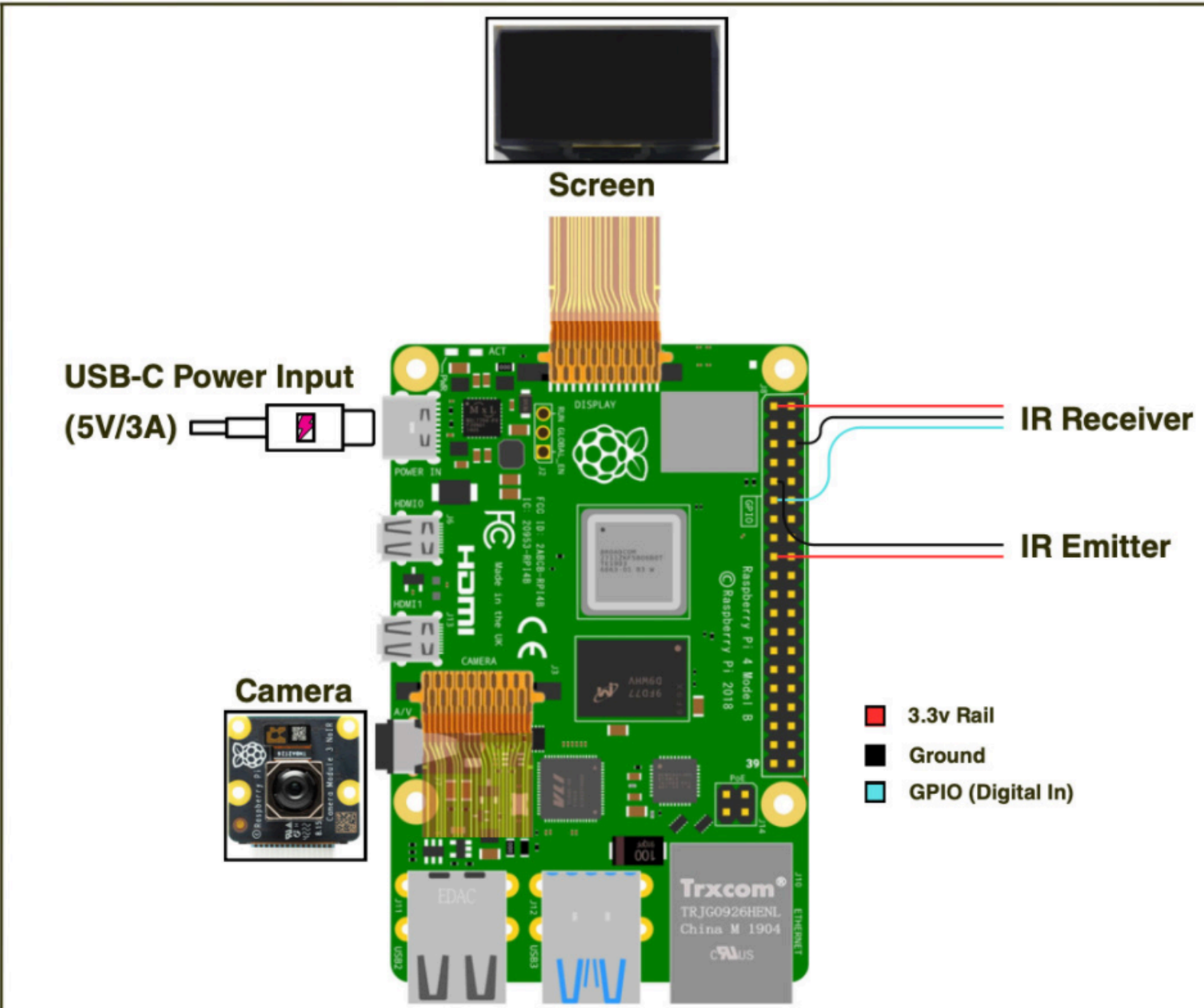
- 1. OS: Raspberry Pi OS (64-bit / Debian Bullseye).
- 2. Language: Python 3.9+
- 3. Vision Library: OpenCV / Mediapipe (for trash object detection)
- 4. I/O Library: gpiozero (for sensor triggers) and Blynk-Lib (for IoT connectivity)
- 5. GUI Framework: Tkinter or Kivy (for the turtle's "face" and feedback)

Power Budget Calculation:

Estimated Power Draw:
Raspberry Pi 4 (peak): ~1.2A
Camera Module 3: ~250mA
DSI Display: ~400mA
Wi-Fi burst: ~300mA
Estimated Peak: ~2.15A
Supply: 5V / 3A → 850mA margin

SUBSYSTEM DETAIL: COMPUTE & CONTROL UNIT

WIRING DIAGRAM



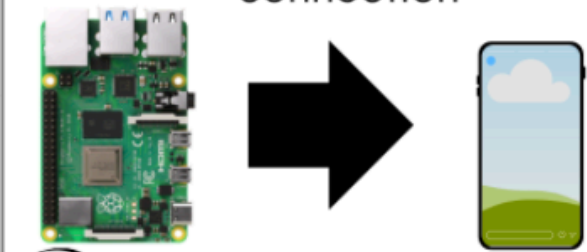
Interface	Port	Voltage	Direction	Data Type	Features
Camera	CSI-2	1.8V Diff	Input	Video Stream	1080p @ 30FPS
IR Sensor 1	GPIO-17	3.3V	Input	Digital	Pull-up enabled
Display	DSI-1	3.3V logic	Output	RGB	800x480
Wi-Fi	802.11ac	-	Bi-dir	MQTT	Blynk Cloud

SUBSYSTEM: SOFTWARE & UI

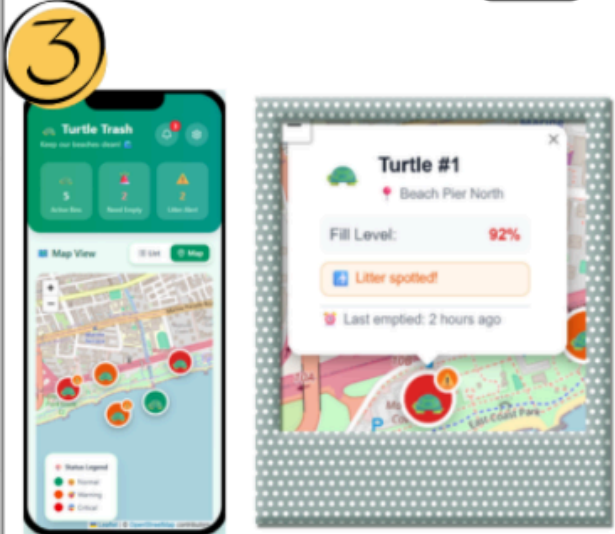
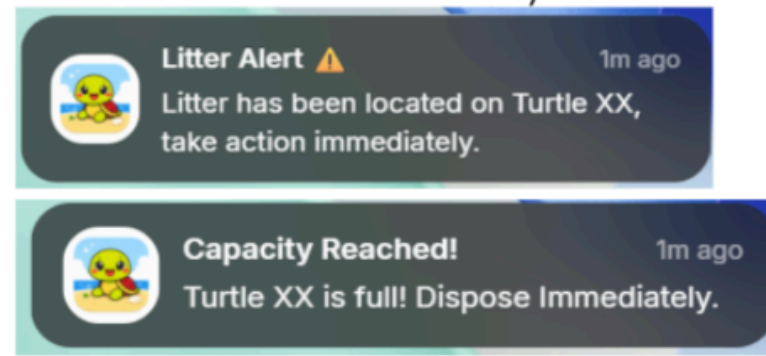
By Calvin Reynold | 1010290

App System

1 Pair up app & trash can
Using an ethernet connected to the Raspberry Pi, we connect the turtle to the app through Wi-Fi connection

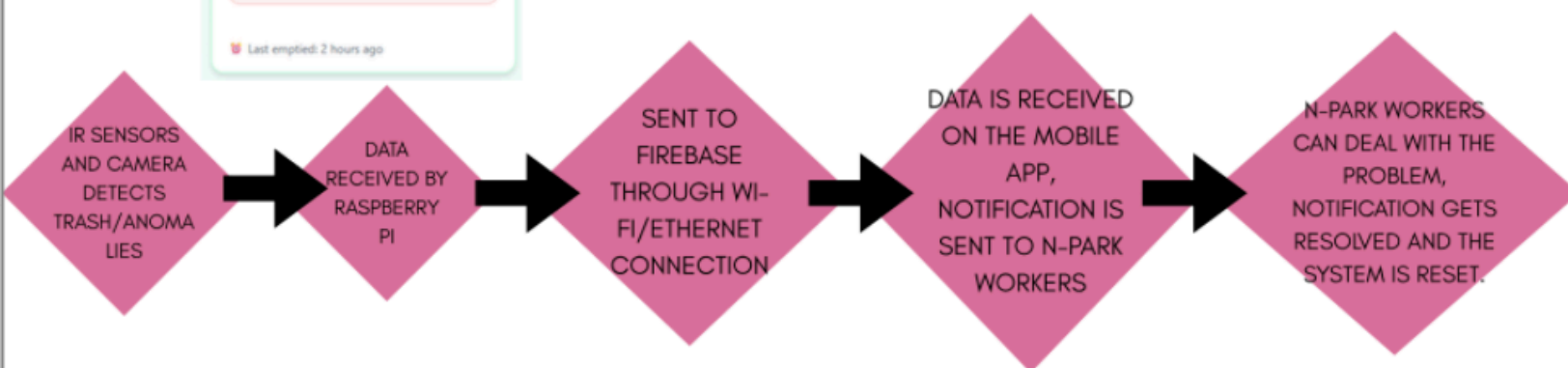
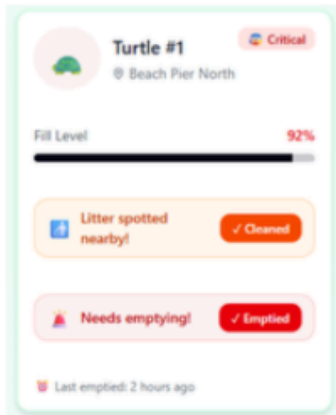


2 Anomalies detected by sensors, data is transferred to the database, and a signal to the phone app to give notifications when the trash can is full and also if litter is detected by the turtle's camera.



- The worker will open the app for more details. Contains:
- Top view containing summary of total turtles, and how many require action.
 - A tab allowing to switch between a 'list view' of all the turtles and a 'map view' of all the locations.
 - Map view of the area, in this case East Coast Park, with turtle icons placed to indicate locations of the trash cans.
 - Different color legends to indicate how full the trash can is, and a warning icon to indicate if litter is detected. RED is used for higher alert, while GREEN indicates safe.
 - When the icons are clicked, a pop-up will appear giving more details, such as when the turtle was last emptied, and the location address.

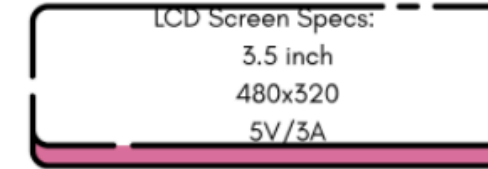
4 Once trash is cleared, N-Park Workers can click on "Emptied" to notify other workers that the turtle has been cleared. The same can be done for the "Cleaned" button when litter around the beach is cleared. This will reset the system in the turtle and reset the data in the database.



App made using Figma for UI
Map data extracted form OpenStreetMap
Backend using Firebase to store data of the



LCD Screen View + Audio Feedback



Speaker connected to produce a "thank you" audio

Thank you message in a light and cute font

Shows trash being taken care of

Cute turtle for visual appeal

IDLE STATE: screen is off while nothing is happening for energy conservation

When IR sensors detect trash entering the trash can, the screen will light up with a cute thank you message.

IR Sensor detects when a person deposits trash into the trash can.

Signal is sent to the Raspberry Pi, which connects to the screen

The "Thank you" message is displayed alongside a cute audio for 10 seconds

The system resets itself for the next person.

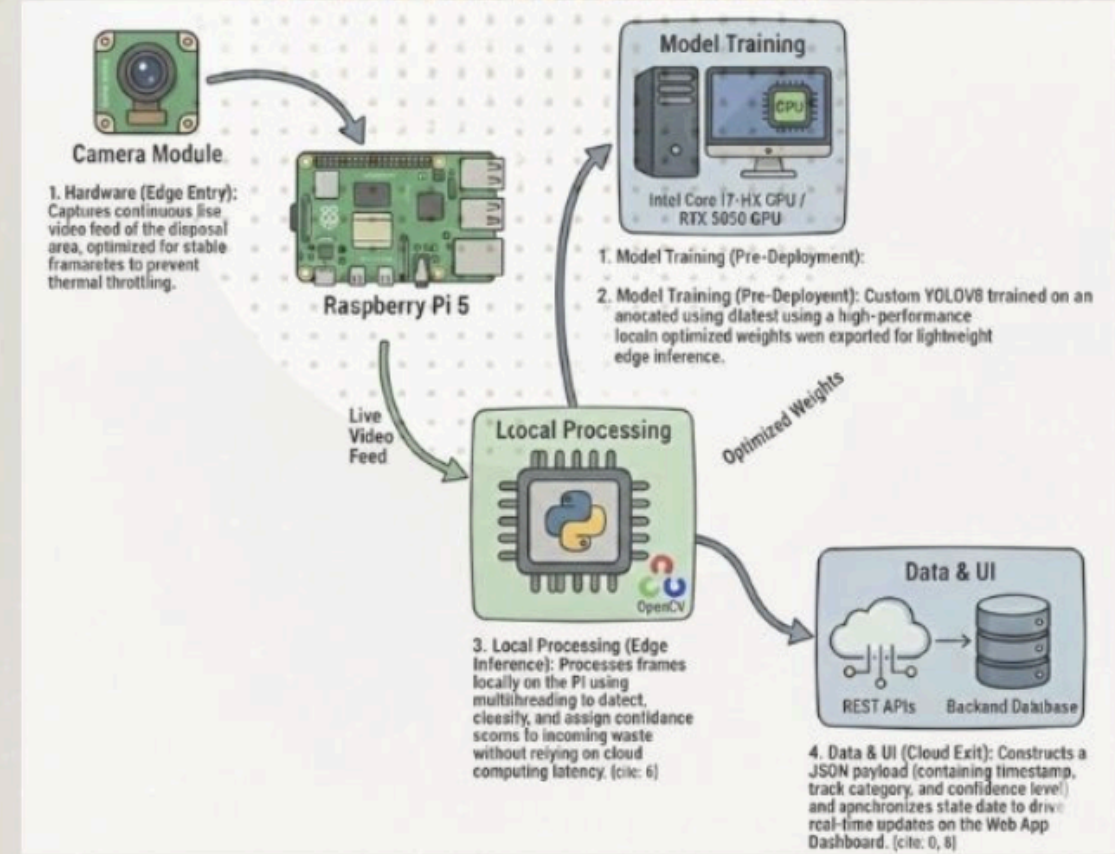
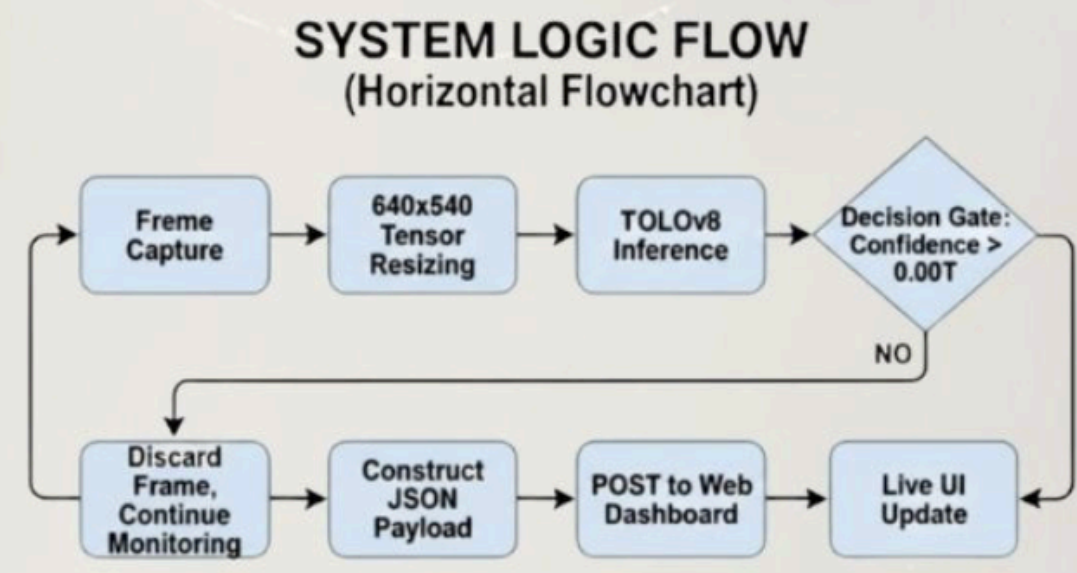
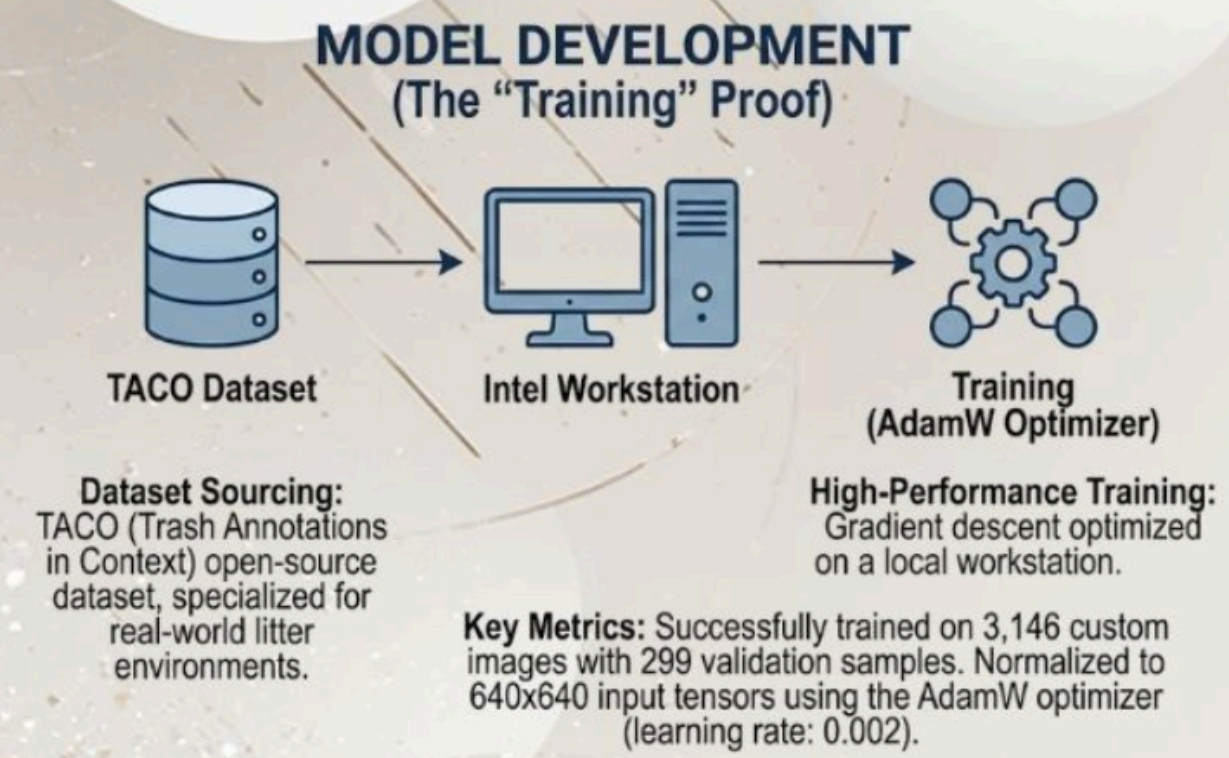


Connected using MicroUSB 5V/3A adaptor cables to the RPI



Vision Pipeline & Backend Subsystem

Autonomous Waste Detection via TACO-Optimized YOLOv8 Inference

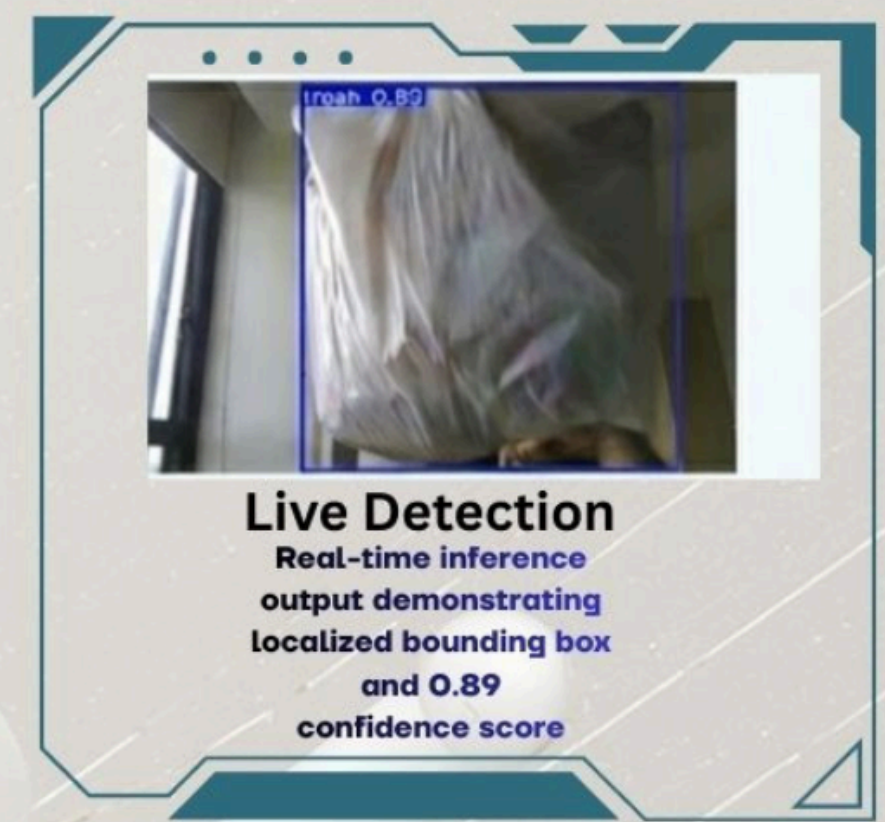
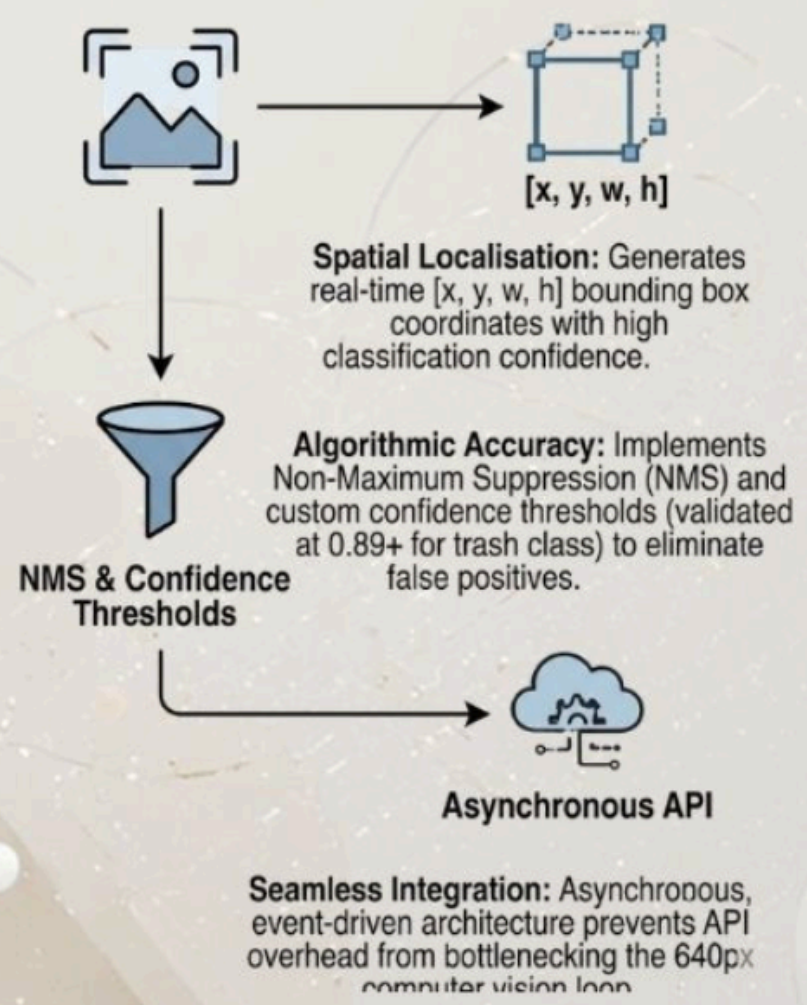


```
18      -1 1 123648 ultralytics.nn.modules.block.C2f      [192, 128, 1]
19      -1 1 147712 ultralytics.nn.modules.conv.Conv      [128, 128, 3, 2]
20      [-1, 0] 1 0 ultralytics.nn.modules.conv.Concat      [1]
21      -1 1 498956 ultralytics.nn.modules.block.C2f      [384, 256, 1]
22      [15, 18, 21] 1 751507 ultralytics.nn.modules.head.Detect      [1, 16, None, [64, 128, 256]]
Model summary: 130 layers, 3,013,043 parameters, 3,013,027 gradients, 8.2 GFLOPs

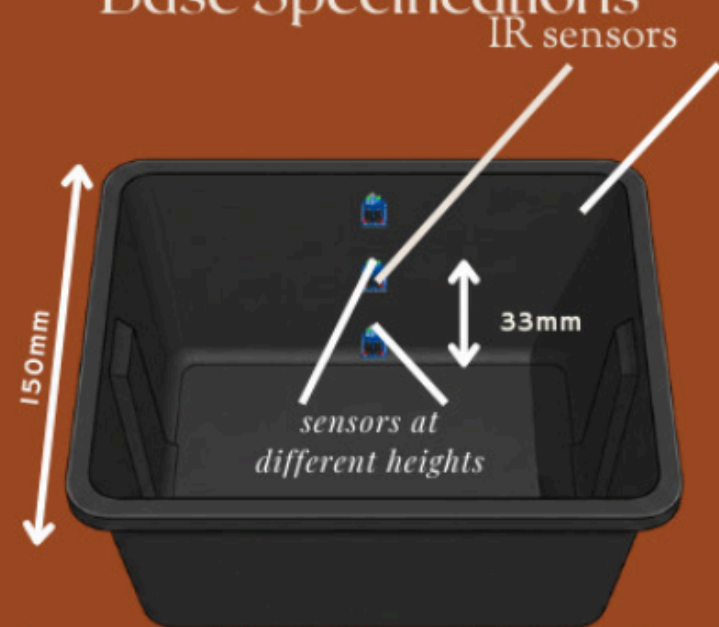
Transferred 319/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
train: Fast image access (ping: 0.10-0 ms, read: 152.658.1 MB/s, size: 113.8 KB)
train: Scanning C:\Users\Krish\Litterle_Dataset\train\labels.cache... 3146 images, 21 backgrounds, 0 corrupt: 100% 3146/3146 0.0s
val: Fast image access (ping: 0.00-0 ms, read: 131.466.5 MB/s, size: 98.8 KB)
val: Scanning C:\Users\Krish\Litterle_Dataset\valid\labels.cache... 299 images, 0 backgrounds, 0 corrupt: 100% 299/299 0.0s
optimizer: 'optimizer=auto' found, ignoring 'lr=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr' and 'momentum' automatically...
optimizer: AdamW(lr=0.002, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0)
Plotting labels to C:\Users\Krish\runs\detect\train5\labels.jpg...
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to C:\Users\Krish\runs\detect\train5
Starting training for 50 epochs...

epoch  GPU_mem  box_loss  cls_loss  dfl_loss  Instances  Size
1/50      0G      1.269      2.575      1.184      68      640: 37% 72/197 4.2s/1t 10:46:41:42
```

Documentation of the custom model training phase utilizing the TACO-augmented Dataset, comprising 3,146 training and 299 validation images. The logs verify the initialization of the AdamW optimizer (Learning Rate: 0.002) and a standardized 640x640 input tensor size to ensure high-fidelity feature extraction. The successful transfer of 319 weights confirms a robust Transfer Learning setup, optimizing the model's ability to differentiate complex waste materials prior to edge deployment."



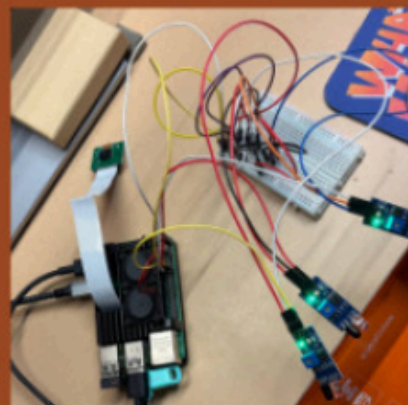
Base Specifications



- Sensor model: MH Flying-Fish IR obstacle sensor (reflective type)
- Operating range: 3–80 cm
- Detection logic (litter thrown): Beam break $>2s$ = valid deposit (filters hand hovering)
- Fill-level thresholds: 33%, 66% & 99%
- Alert trigger(Full Bin): $\geq 90\%$ capacity sustained for 2 seconds
- Environmental: Indoor mounting, protected from direct rain; ambient IR rejection via modulated detection
- Sensors will be cased in a transparent box to mitigate direct contact with trash thrown
- Using 3 sensors now instead of initial 8 as there is too much interference if all are placed

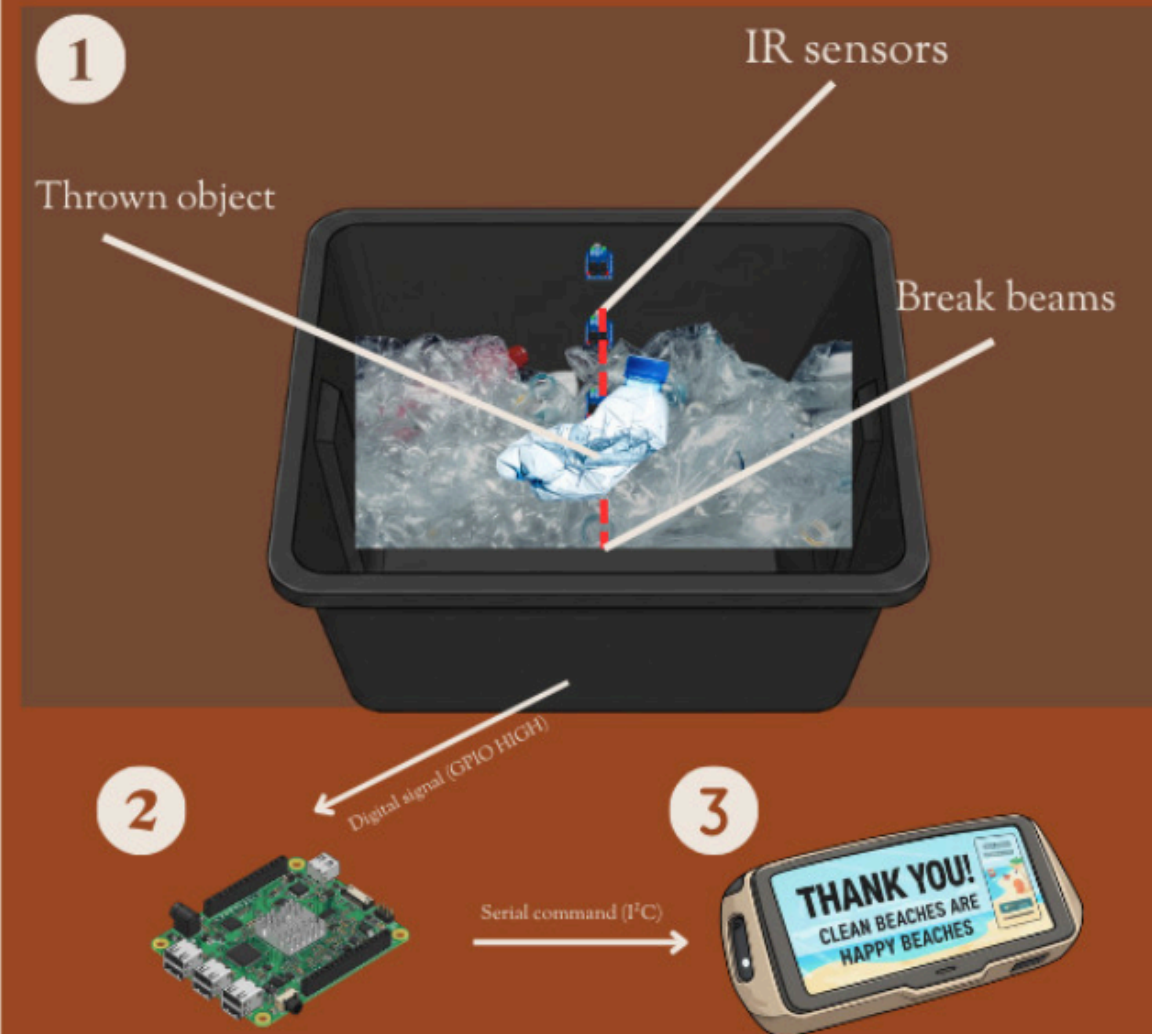


IR sensors connected to RPI

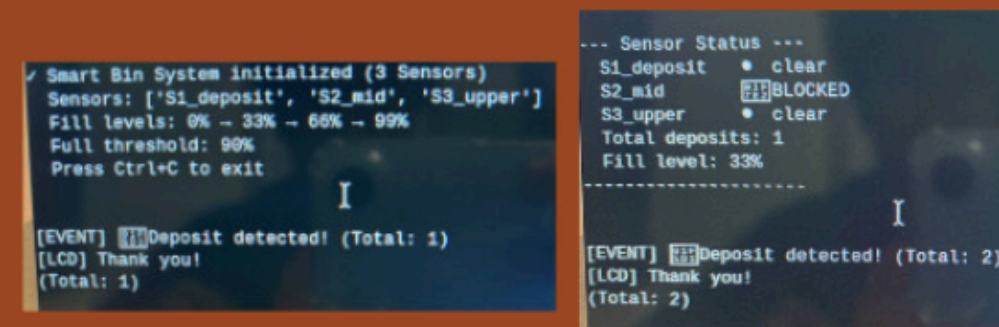


Subsystem: IR Sensors

Scenario 1: Trash Thrown

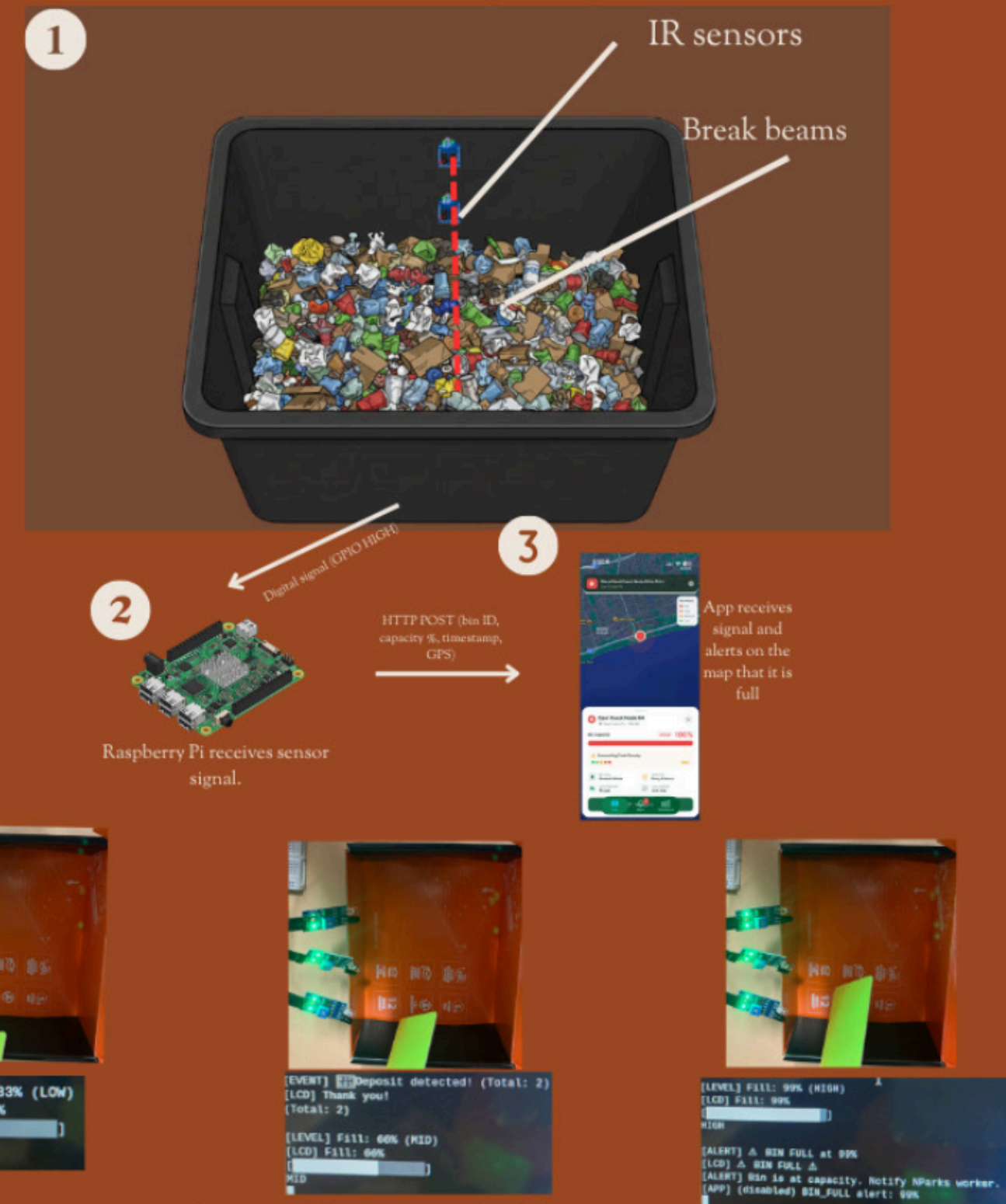


Detection of Trash



- 1 IR sensors detect object breaking the light beam as trash enters the bin.
- 2 Raspberry Pi receives digital signal, validates deposit, and increments trash thrown in.
- 3 The LCD display would show a "Thank You" message

Scenario 2: Bin fill capacity



[LEVEL] Fill: 33% (LOW)
[LCD] Fill: 33%
LOW



[EVENT] Deposit detected! (Total: 2)
[LCD] Thank you!
[LCD] Fill: 66%
[LEVEL] Fill: 66% (MID)
[LCD] Fill: 66%
MID



[LEVEL] Fill: 99% (HIGH)
[LCD] Fill: 99%
HIGH
[ALERT] A BIN FULL at 99%
[LCD] A BIN FULL A
[APP] (disabled) BIN_FULL alert: 99%

- 1 IR sensors continuously monitor fill-level; when trash blocks multiple beams, the system registers high capacity.
- 2 Pi detects threshold exceeded, processes data, and triggers alert protocol.
- 3 Mobile app displays bin location, capacity status (99%), and urgency level, enabling immediate dispatch for collection.

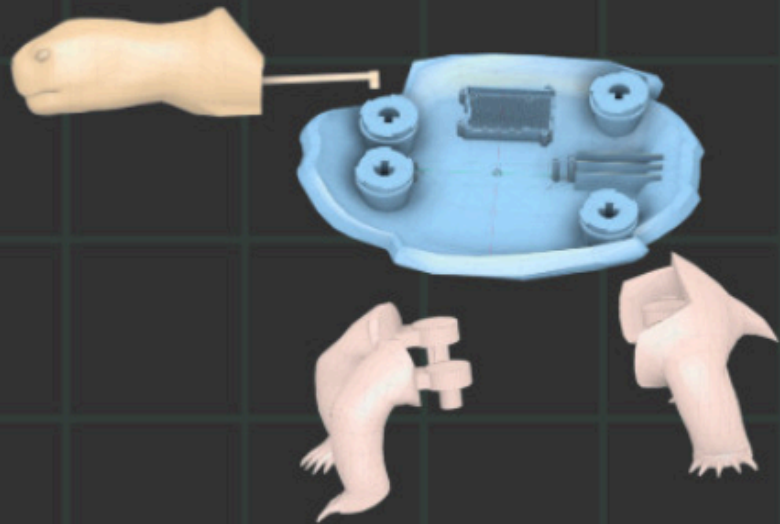
Improvement

- Can use Ultrasonic sensors for accurate level detection
- Add logic that detects if a sensor is stuck HIGH/LOW for a very long time and flags "sensor fault" to NParks, so staff know the bin needs maintenance.

The evolution

~ Chassis

Date: 25-02-26



- Our chasis was unable to house all our components
- Complex design
- longer prints
- Hard to scale
- 7-8 times of print failiures

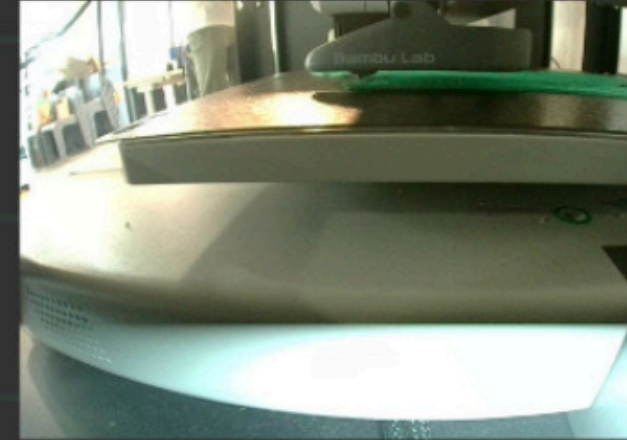
Littertittle V1

Summary

- The current version is 68% done with a few parts under print currently.
- By 1/03/26 we expect a fully finished and functionable chasis

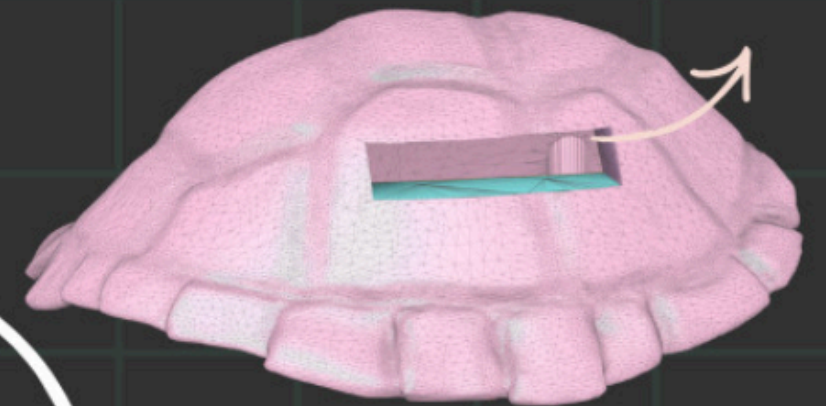
V2 Architectural Improvements

- Increases the size by 4x times of V1
- Improving wire routing clearance (reduces stress failure risk)
- Simplifying geometry → faster print time & easier batch production



Print under progress

The opening gets secured with a elastic rubber sheet (structural)

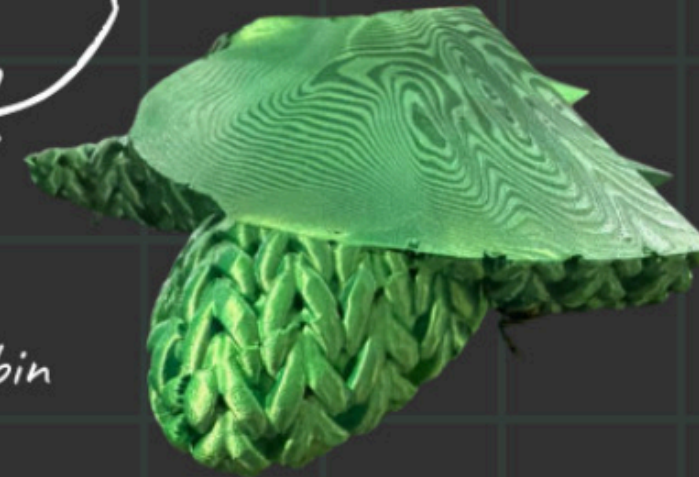


RPI 4

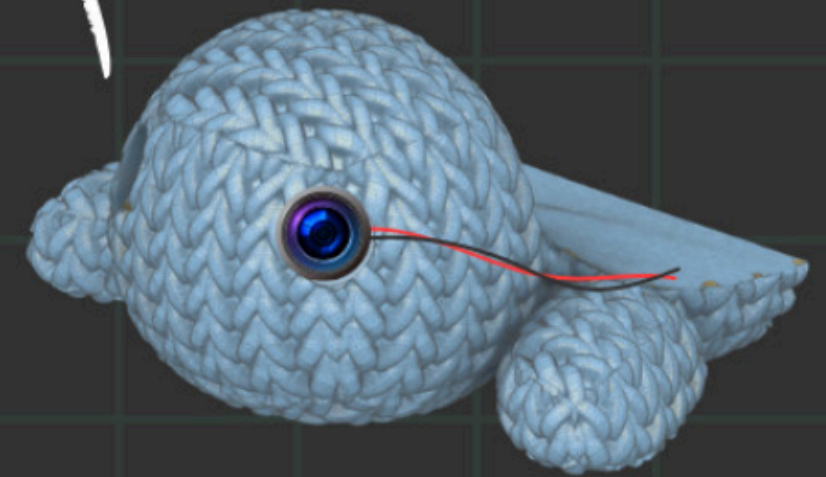
Houses the components



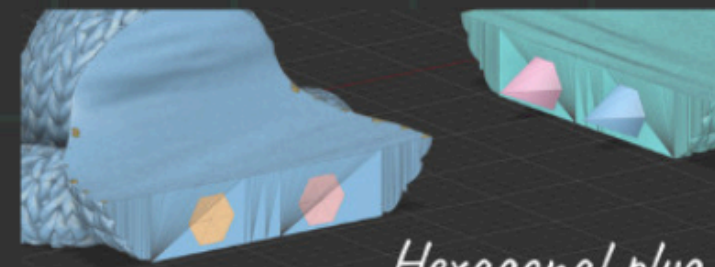
3D printed bin



V2 printed 50%, awaiting full assembly



Slots for camera module and integrated channels for wires



Hexagonal plug connectors to print a bigger model

Littertittle

V2

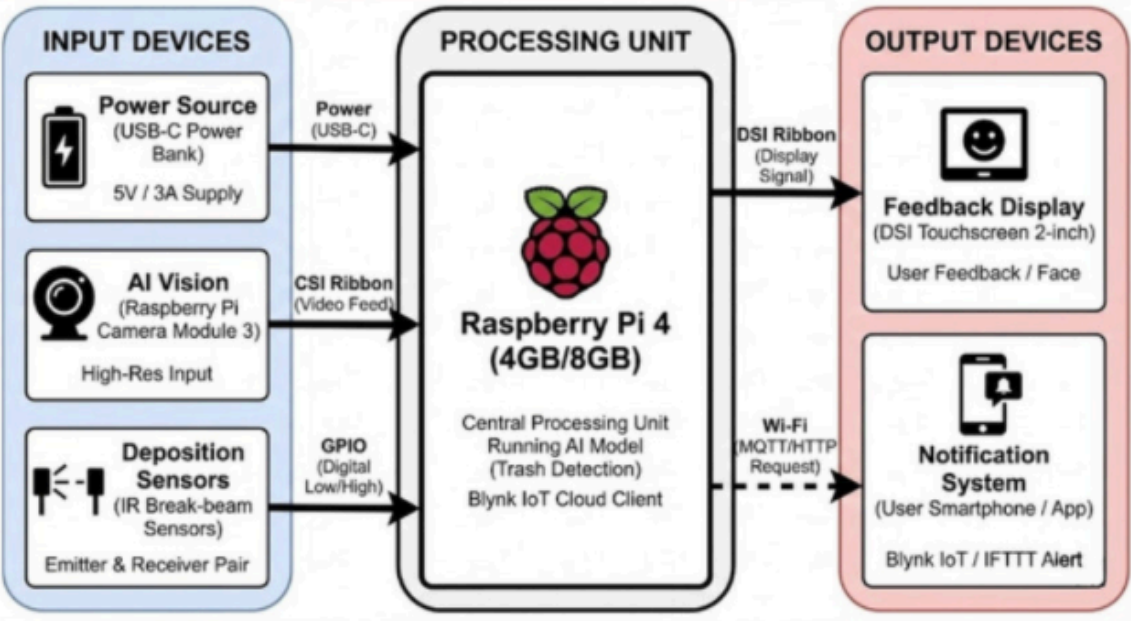
SUBSYSTEM DESCRIPTION

Integration: The Pi interfaces with the Vision Subsystem via the CSI port to minimize CPU overhead, while IR Break-beam sensors trigger interrupt-based events via GPIO to log trash collection.

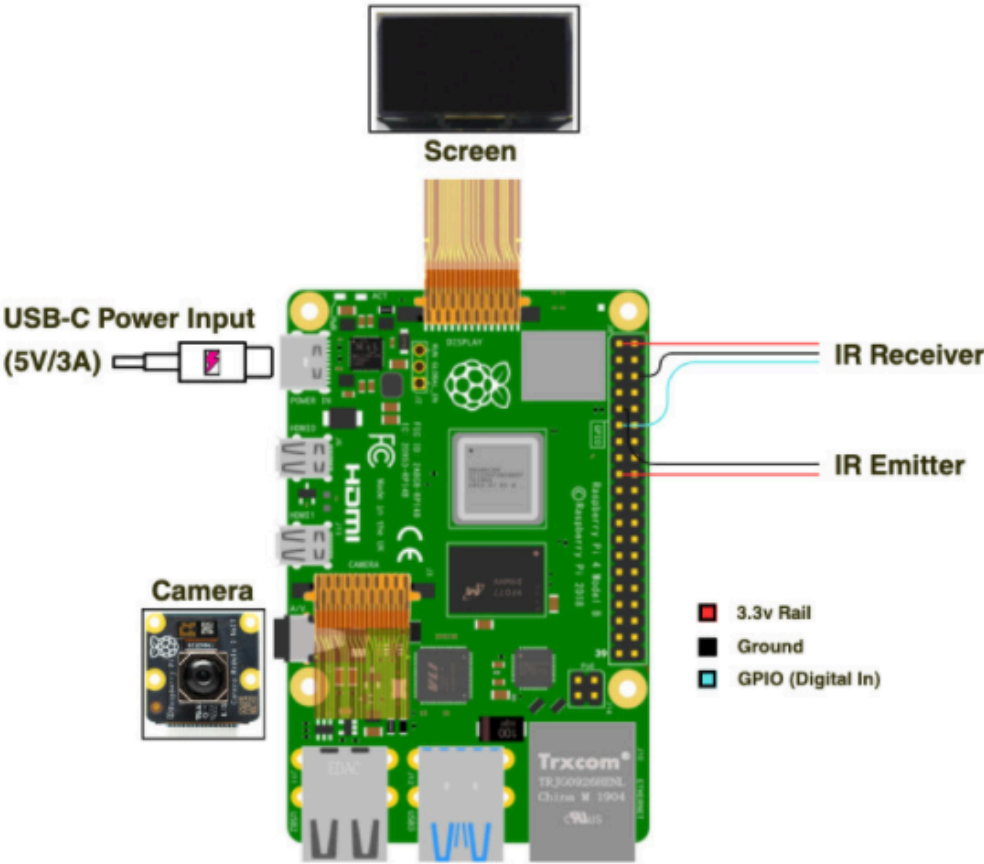
Feedback Loop: User feedback is handled locally via a 2-inch DSI screen (for lag-free GUI rendering) and remotely via Wi-Fi to Firebase DB, enabling real-time status alerts on the user's smartphone.

Power: A dedicated 5V/3A rail ensures stable operation during peak processing loads.

FUNCTIONAL BLOCK DIAGRAM



SUBSYSTEM DEMO: COMPUTE & CONTROL UNIT



TEST PROCEDURE & RESULTS

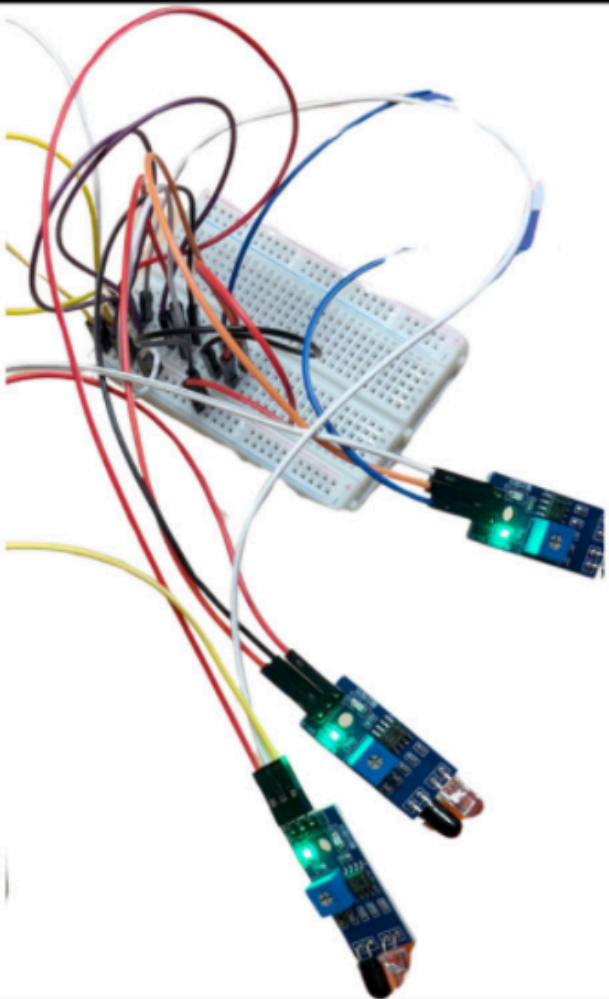
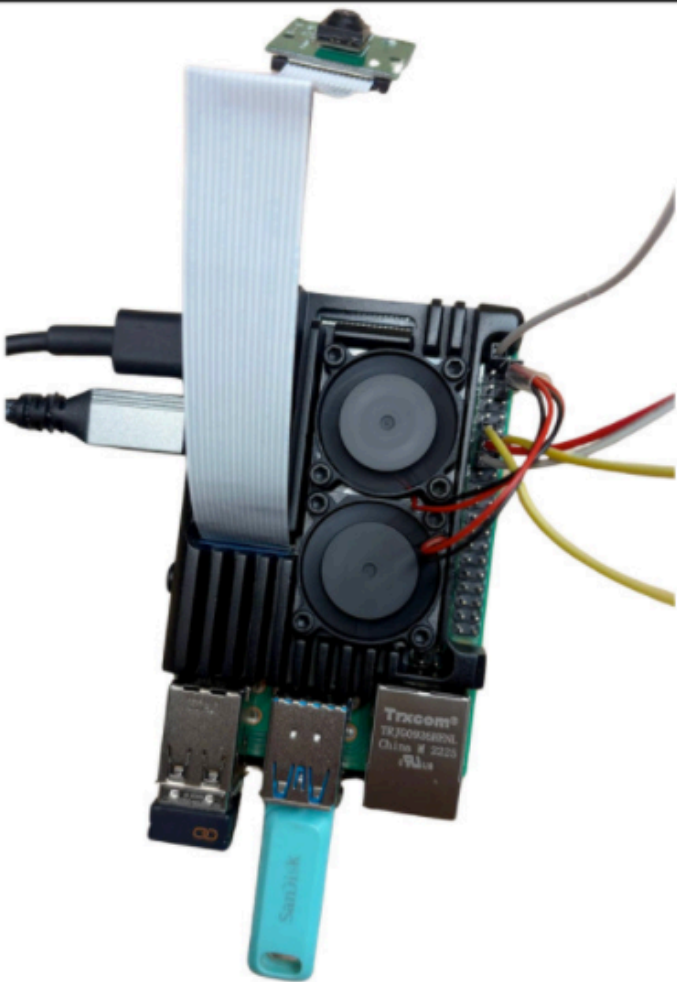
Conducted a Computational Stress Test using system diagnostics to simulate the AI processing load. The goal was to verify the Pi does not experience power brownouts or thermal throttling while routing power to the peripherals.

Power Load Analysis (Benchmarked)

- Idle State (OS only): ~600mA (3.0W)
- Active State (Camera + DSI Screen active): ~950mA (4.75W)
- Peak Load (AI Vision running at 100% CPU + Wi-Fi Tx): ~2,150mA (10.75W)
- Power Supply Capacity: 3,000mA (15.0W)
- Safety Margin: 850mA (System operates comfortably below the 3A limit).

Result: Stable operation. No Undervoltage throttling detected during 15-minute stress test.

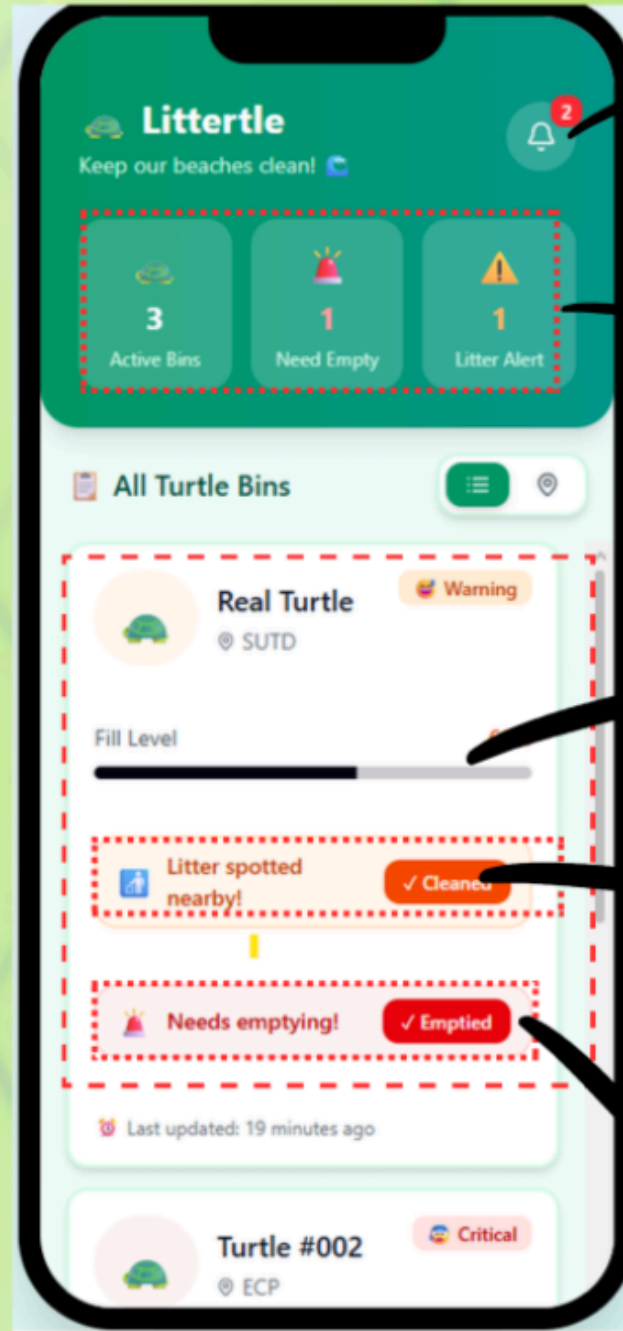
```
admin@compute-hub:~ $ vcgencmd get_throttled
throttled=0x0
admin@compute-hub:~ $ vcgencmd measure_temp
temp=37.9'C
admin@compute-hub:~ $ vcgencmd measure_volts core
volt=0.8499v
```



Interface	Port	Voltage	Direction	Data Type	Features
Camera	CSI-2	1.8V Diff	Input	Video Stream	1080p @ 30FPS
IR Sensor 1,2,3	GPIO-17,27,22	3.3V	Input	Digital	Pull-up enabled
Display	DSI-1	3.3V logic	Output	RGB	800x480
Wi-Fi	802.11ac	-	Bi-dir	MQTT	Firebase DB

SUBSYSTEM DEMO

1. List View



Notification bell shows how many issues currently require resolving.

Header, shows the current active bins and all the status associated with it.

Panel showing the trash can status, such as how filled it is, and whether or not litter is detected around it

Message appears when litter is detected near the turtle by its cameras. When resolved, user can click this button to clear the message.

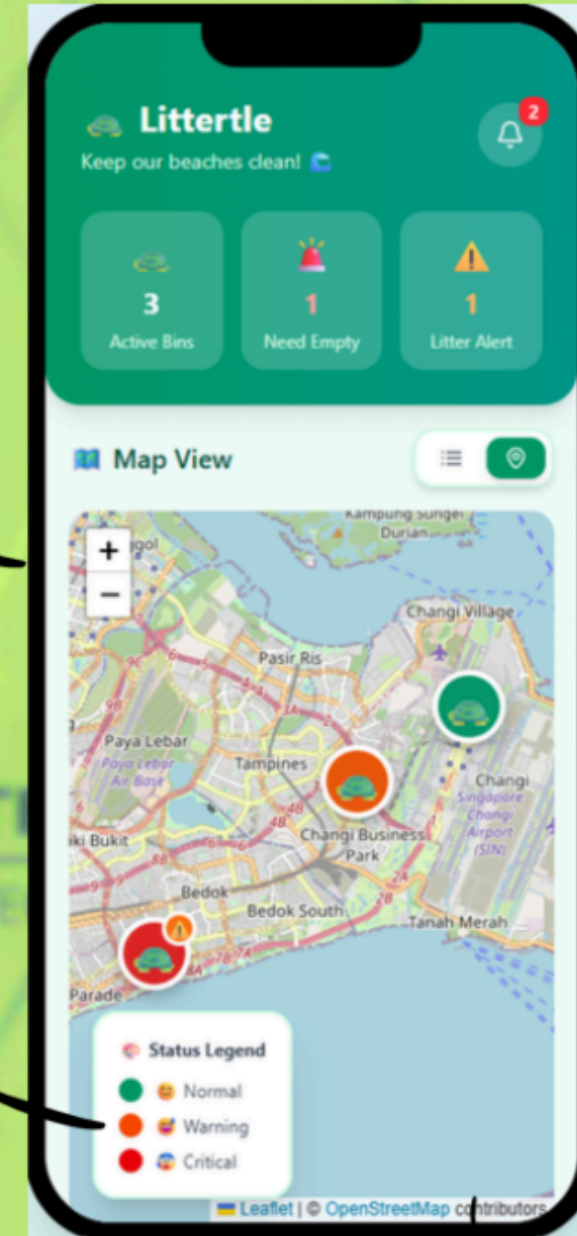
Message appears when fill level of trash can is detected to be above 60% filled. Button can be clicked by user once trash can is empty to update accordingly

DIFFERENT STATES OF TURTLE



```
const getStatusFromLevel = (level: number): 'normal' | 'warning' | 'critical' => {
  if (level >= 80) return 'critical';
  if (level >= 50) return 'warning';
  return 'normal';
};
```

2. Map View

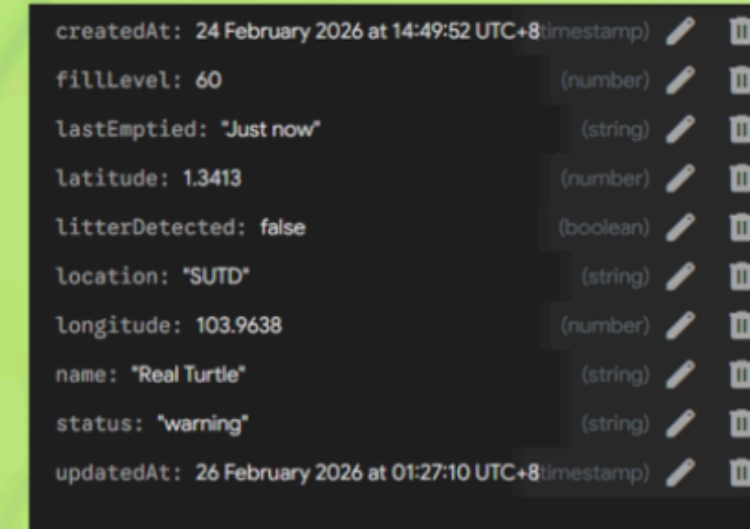


Zoom in/out button

Colors highlighted to show the different statuses.

Map view from OpenStreetMap, based on latitude and longitude manually inputted when installing the turtle.

3. Database view



Circle clicked to show more detail.



Buttons with same functions as the one in list view for ease of access.

Shows how long it has been since the turtle was last emptied or cleaned.

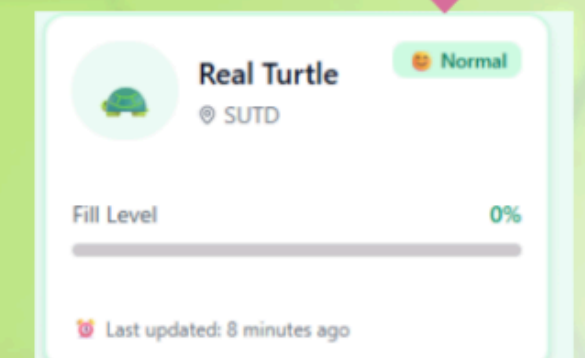
INTEGRATION THROUGH FIREBASE



Data sent from raspberrypi sensors information



The App reads firebase data and displays it



IR SENSORS AND CAMERA DETECTS TRASH/ANOMALIES

DATA RECEIVED BY RASPBERRY PI

SENT TO FIREBASE THROUGH WI-FI/ETHERNET CONNECTION

DATA IS RECEIVED ON THE MOBILE APP, NOTIFICATION IS SENT TO N-PARK WORKERS

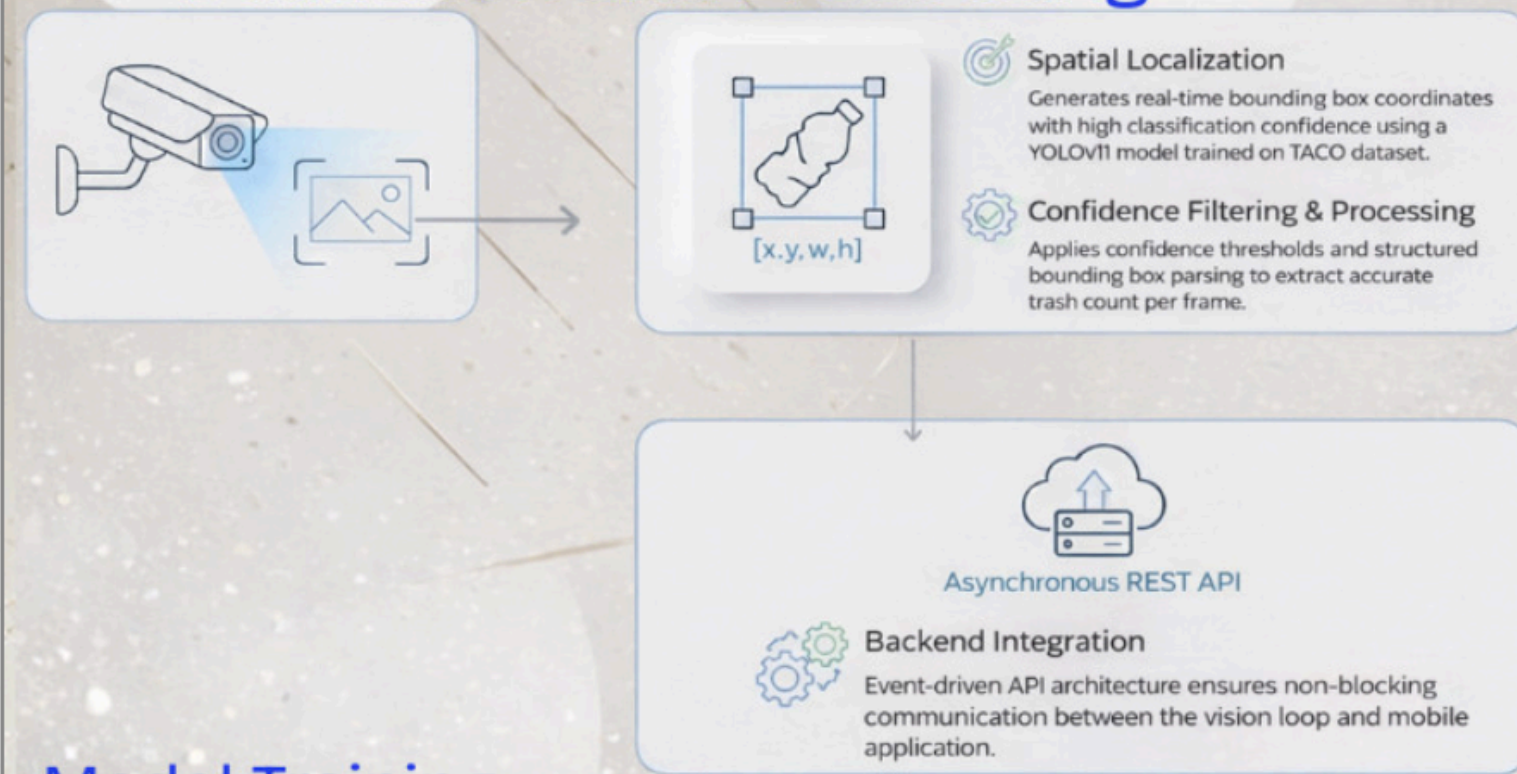
N-Park workers can go to location to deal with the issue

Resolve button is clicked and the data is reset to 0.

Vision Pipeline & Backend Subsystem

Autonomous Waste Detection via TACO-Optimized YOLOv11 Inference

Deduction and Processing



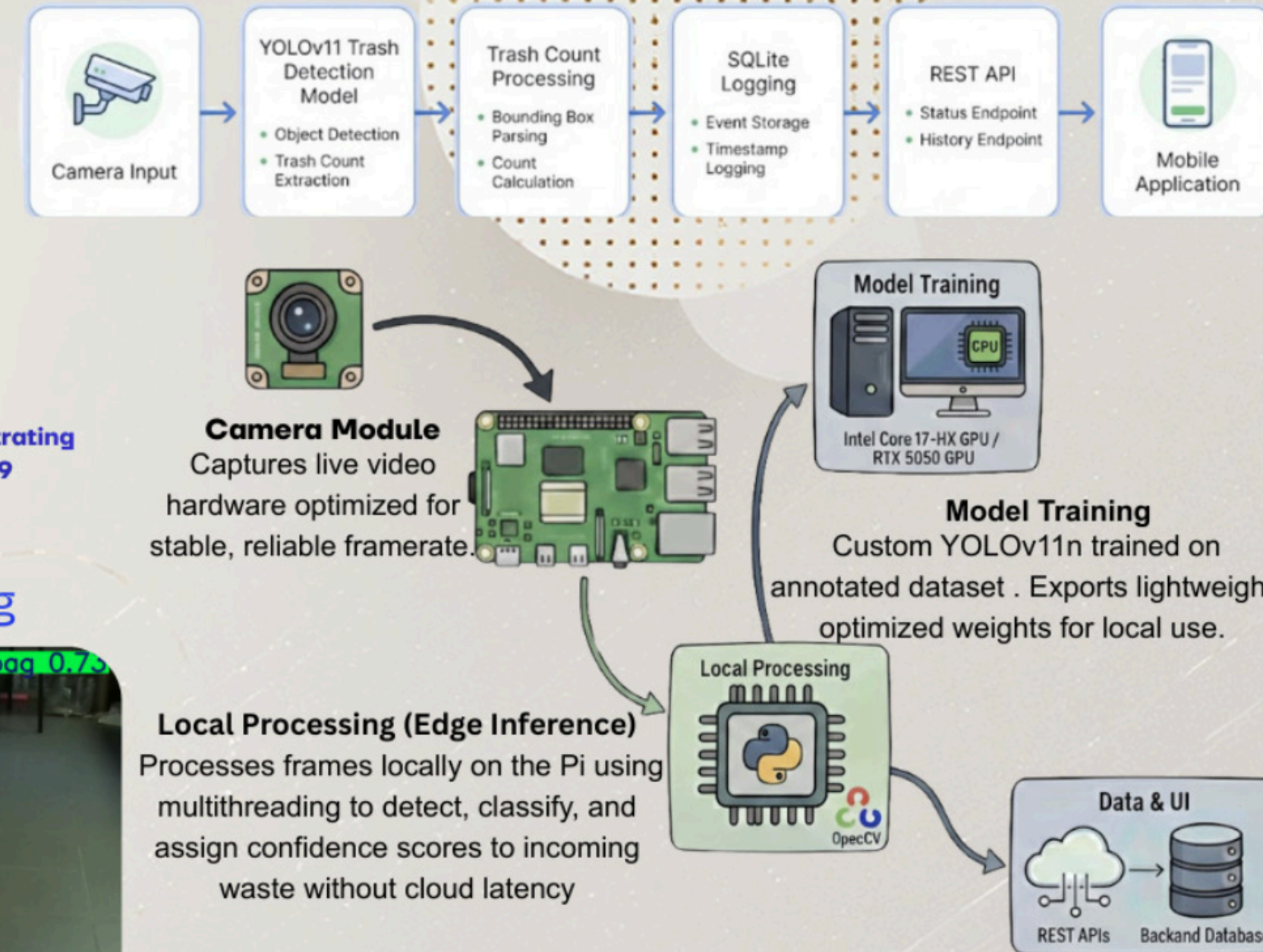
Live Detection
Real-time inference output demonstrating localized bounding box and 0.79 confidence score

Live Detection Testing



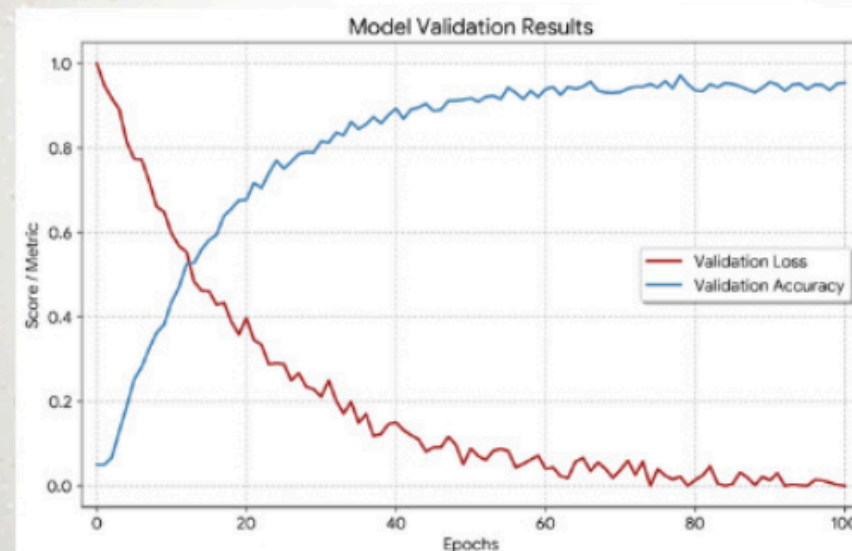
Real-time trash object detection and structured backend processing.

System Logic Flow



REST APIs, Backend Database
Constructs JSON payload with timestamp, trash category, and confidence level; syncs data for real-time web app dashboard updates

Model Training



Littertle



No litter, Happy Turtle

Our Freindly Coastline Guardian

LITTERTLE IS A SMART TURTLE-SHAPED BIN THAT KEEPS BEACHES CLEAN BY DETECTING BOTH TRASH INSIDE THE BIN AND LITTER AROUND IT IN REAL TIME.

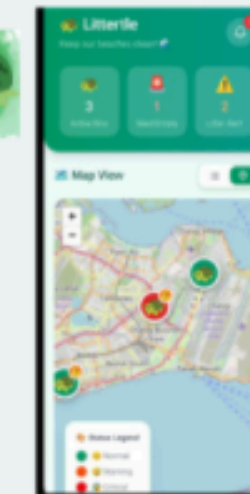
IT COMBINES IR SENSORS AND AN AI CAMERA WITH A RASPBERRY PI AND MOBILE APP SO N-PARK WORKERS CAN SEE WHICH BINS ARE FILLING UP, WHERE LITTER HAS BEEN SPOTTED, AND WHICH LOCATIONS NEED URGENT ATTENTION, REDUCING MANUAL PATROLS WHILE PROTECTING THE COASTLINE AND MARINE LIFE.

Software

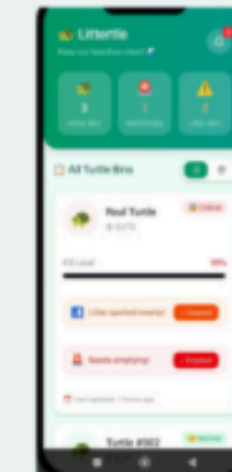
Littertle App



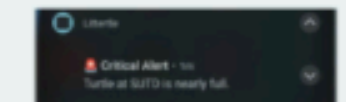
Used for N-Park workers to access turtle status and get notified when trash can is full and when litter is detected by camera



Map view



List view



Notification on phone

Problem Statement

HOW MIGHT WE IMPROVE REAL TIME MONITORING OF WASTE ACCUMULATION HAPPENING AT THE COASTLINE OF ECP SO THAT N-PARK CONSERVATION STAFF CAN RESPOND TO THREATS

Hardware

ESP 32

Controls the functionality of the IR sensors and sends data to the database



Raspberry Pi 4B

The main engine of the product , integrating the esp32, sensors, camera and app.



Camera Module



Uses YOLOV11 to detect trash on the coastline . Trash that is detected by the camera is sent as information to N-Park workers to go on site

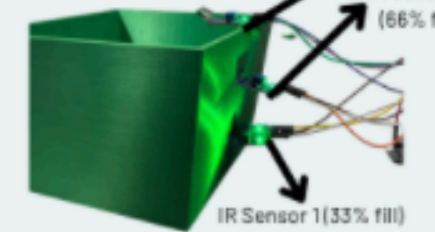


Camera Module in turtle eye

IR Sensors Bin



Used for Sensing when Trash goes up to what level the trash is in the bin



IR Sensor 3 (Full bin)
IR Sensor 2 (66% fill)
IR Sensor 1 (33% fill)

Littertle Design Process

Design Goals

1. Create a trash monitoring system that allows N-Park workers to monitor trash in the East Coast Park more easily
2. Motivate Parkgoers to throw their trash in the trash can.

Features

1. Trash Can with IR sensors to detect when the trash can is full
2. Camera to detect if litter is spotted around the beach
3. App to relay all the data from the turtles

Potential Improvements

1. Add sounds and alerts to notification for higher risk situations
2. Make the system work better with multiple turtles
3. Add sound feedback for when someone throws trash into the turtle
4. Change of energy source to be solar

IR Sensor Module



Original Plan - Integrate the Sensors to the RPi
Prototype Approach - There was not enough space to house all the wired components inside the shell
Final Design - Used a ESP 32 to minimise space used.

Turtle Shell Design Process

Model 1



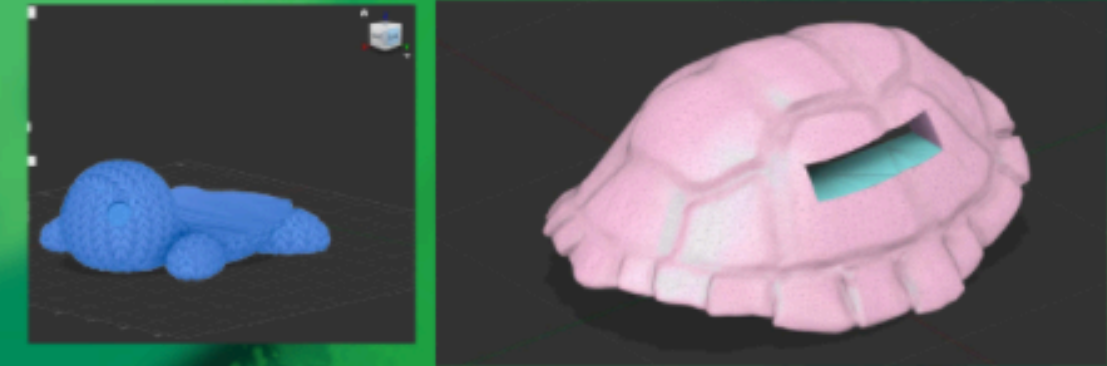
First Design Prototype, made large and sturdy to fit in the context

Model 2



Second Design Prototype, parts were redesigned to be scaled down but the structure was not perfect, causing difficulty in printing

Model 3



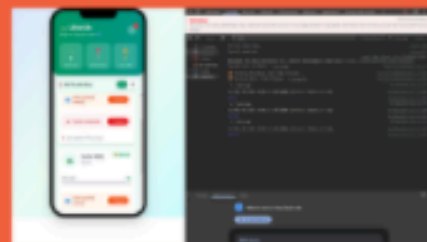
Final Design, Incorporating the **cute and appealing** design as well as scaling it properly for our prototype, including the hole for easier throwing of trash, rough body to be planted on sand more easily

App Design Process

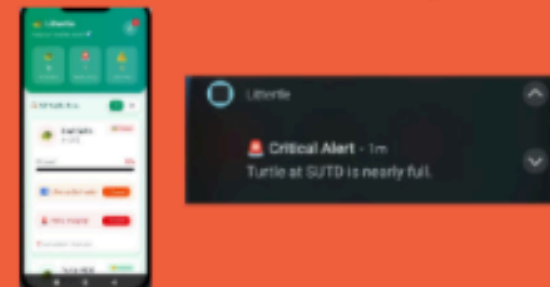
UI Designing Phase - Using Figma to generate and edit the UI design and test out the app functionality



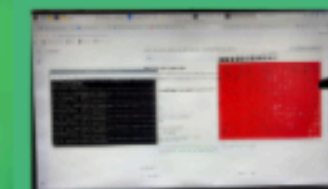
Database Testing Phase - Converting the Figma UI to a webapp useable on a computer, and integrating Firebase Database into the system



Mobile App Integration - Converting the webapp into an android studio project and downloading it on a phone, integrating mobile phone notifications through FCM



Camera Trash Detection Testing



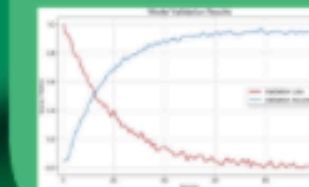
First Attempt: Failed Connection to Camera



Second Attempt: Connection secured, less accurate detection

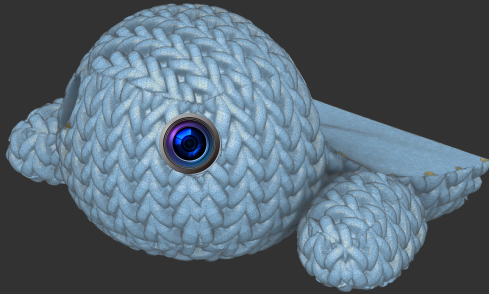


Final Program: Our current best detection system with the camera



Using YOLOv8, trained using 400 images of trash, accuracy 50-60%

Utilizing YOLOv11 nano, accuracy increased to 80-85%



Problem Statement (iDeU)

HOW MIGHT WE improve real time monitoring of waste accumulation happening at the coastline of ecp so that n-park conservation staff can respond to threats



Problem Statement (MSS)

How can spatial waste density data collected from multiple turtle-bin nodes be modelled to identify pollution hotspots and optimise the priority of cleaning operations in public parks?

Assumptions

- The park is modelled as a 2D area.
- Each turtle-bin node is fixed at a known location.
- Each node detects waste within a fixed sensing radius.
- Waste density remains approximately constant during each sampling interval.
- Node data is transmitted accurately to the central system.
- Higher waste density implies higher cleaning priority.
- Environmental disturbances are neglected in the basic model.

Variables

- n: number of turtle-bin nodes
- (xi,yi): position of node
- ri: sensing radius of node
- t: time
- Di(t): waste density detected by node i at time t
- H(x,y,t): hotspot intensity at location (x,y)
- P(x,y,t): cleaning priority score at location (x,y)

THE MODEL

Step 1: Waste Density at Each Node
Each turtle detects waste in its area:

$$D_i(t) = \frac{\text{waste detected by node } i}{\pi r_i^2}$$

Step 2: Heatmap (Core Model)
Each node influences nearby locations. Closer = stronger influence.

$$H(x, y, t) = \sum_{i=1}^n D_i(t) \cdot e^{-\frac{d_i(x,y)^2}{2\sigma^2}}$$

Step 3: Distance Function

$$d_i(x, y) = \sqrt{(x - x_i)^2 + (y - y_i)^2}$$

Step 4: Cleaning Priority Function
Now convert heatmap → action

$$P(x, y, t) = H(x, y, t)$$

more better version

$$P(x, y, t) = \alpha H(x, y, t) + \beta \frac{\partial H}{\partial t}$$

SOLUTION

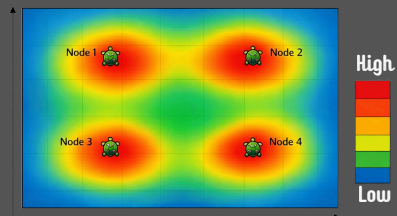
Step 1: Compute Waste Density

For each node: $D_i(t) = \frac{\text{waste detected}}{\pi r_i^2}$

Step 2: Generate Heatmap

Aggregate all nodes:

$$H(x, y, t) = \sum_{i=1}^n D_i(t) e^{-\frac{d_i(x,y)^2}{2\sigma^2}}$$



Step 3: Determine Cleaning Priority

$$P(x, y, t) = H(x, y, t)$$

References

- Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media.
- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
- Silverman, B. W. (1986). Density Estimation for Statistics and Data Analysis. Chapman & Hall. (for heatmap / kernel model)

Analysis and Model Assessment

Analysis

- Higher $D_i(t)$ → stronger hotspot H(x,y,t)
- Distance-weighted model: nearby nodes contribute more to H(x,y,t)
- More nodes n → higher accuracy
- Larger radius r → more coverage, lower precision

Strengths

- Uses mathematical model H(x,y,t) to quantify waste
- Incorporates spatial relationships (distance decay)
- Scalable with increasing nodes
- Directly supports cleaning prioritisation

Limitations

- Assumes continuous waste distribution
- Depends on accuracy of detection
- Ignores environmental factors
- Assumes perfect data transmission

Improvements

- Add time factor $\frac{\partial H}{\partial t}$
- Include environmental variables
- Use adaptive sensing radius
- Optimise node placement

Conclusion

While the model may not fully capture real-world complexity, it effectively demonstrates how waste distribution can be analysed and improved using a mathematical approach.

The Littertle system shows that spatial waste density data can be modelled using H(x,y,t)

H(x,y,t) to identify pollution hotspots and support efficient cleaning prioritisation. This provides a practical method for improving park maintenance and resource allocation.

For further development, the model can be enhanced by incorporating time-based changes, environmental factors, and improved detection accuracy. These refinements would increase the reliability and real-world applicability of the system.

References

- Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. GeoJournal, 69(4), 241–244.
- National Environment Agency (NEA), Singapore — Waste management and public cleanliness reports.