Boston University
Electrical & Computer Engineering
EC464 Senior Design Project

Full Functional Test Report

Ecobin

by

Team 9
Ecobin

Team Members

Aditya Wikara adwikara@bu.edu
Hayato Nakamura hayaton@bu.edu
Kevin Sinaga kfsinaga@bu.edu
Shreenidhi Jayaram shreej@bu.edu
Charles Thao cthao19@bu.edu

Ecobin

Table of Contents

1. Objective

For final functional testing, Ecobin successfully identified images, which were in the housing and platform constructed by the Mechanical Engineering counterpart, and mechanically sorted the objects using a stepper motor. The team designed and implemented an embedded system with Raspberry Pi (henceforth RasPi or Pi) which took photos using the Raspberry Pi Camera. In terms of software, the front-end of this prototype includes an iOS application which displays the type of disposal along with user statistics. The back-end employed a Tensorflow object detection model, trained on Boston University's Shared Computing Cluster (henceforth SCC) and running on an OpenStack cloud. Ecobin also used MongoDB Atlas as its database.

2. Equipment & Setup

The Ecobin system includes three major components: hardware, software, and mechanical.

Hardware

- Raspberry Pi 3 B+ (with 32GB SanDisk SDHC Class 10 card)
- Raspberry Pi Camera Module v2
- Wide Angle Camera Lens
- PIR Motion Sensor
- Ultrasonic Sensor HC-SR04
- LED Strips (12V, 0.9W)
- 5V/12V DC Power Supply Unit
- Transistor (PN2222A BJT)
- TB6612 Stepper Motor Driver
- Phidget Motor Controller
- PCB Breadboards/Protoboards

Software

- Object Classification (machine learning)
- Web Scraper (to collect datasets from Google Images)
- MongoDB Atlas (database)
- Python scripts for sensors and overall Ecobin functionality

- iOS Application (10.2.0+, Swift 4.0+)
- REST API deployed on cloud instance
- Dependencies
    - Pymongo, base64, numpy, PIL, keras/tensorflow,httpauth,requests

Mechanical

- Object detection platform
- Ecobin housing
- Camera lens mount (3D printed)
- 2 stepper motor (NEMA-17, and NEMA-17 + Gearbox)

Overall Architecture

Figure 1 describes how each of Ecobin's module interacts with each other. So far, these modules interact with each other seamlessly. *Figure 2* is a 2D representation of the mechanical design that the ME team made for this prototype. It consists of a semicircular platform for object detection, as well as a housing (box) to keep the environment enclosed. The semicircle is the platform on which the object is placed. The camera, RasPi, and LED strip are mounted on the left side of this platform. The motion sensor is positioned on top of the housing, right beside the lid. The sorting mechanism employed is a sweeper comprising the rear wall of the imaging stage, which turns clockwise, as seen from the top, if the object is general waste, and counterclockwise if recyclable. A stepper motor triggers this sweeping motion.

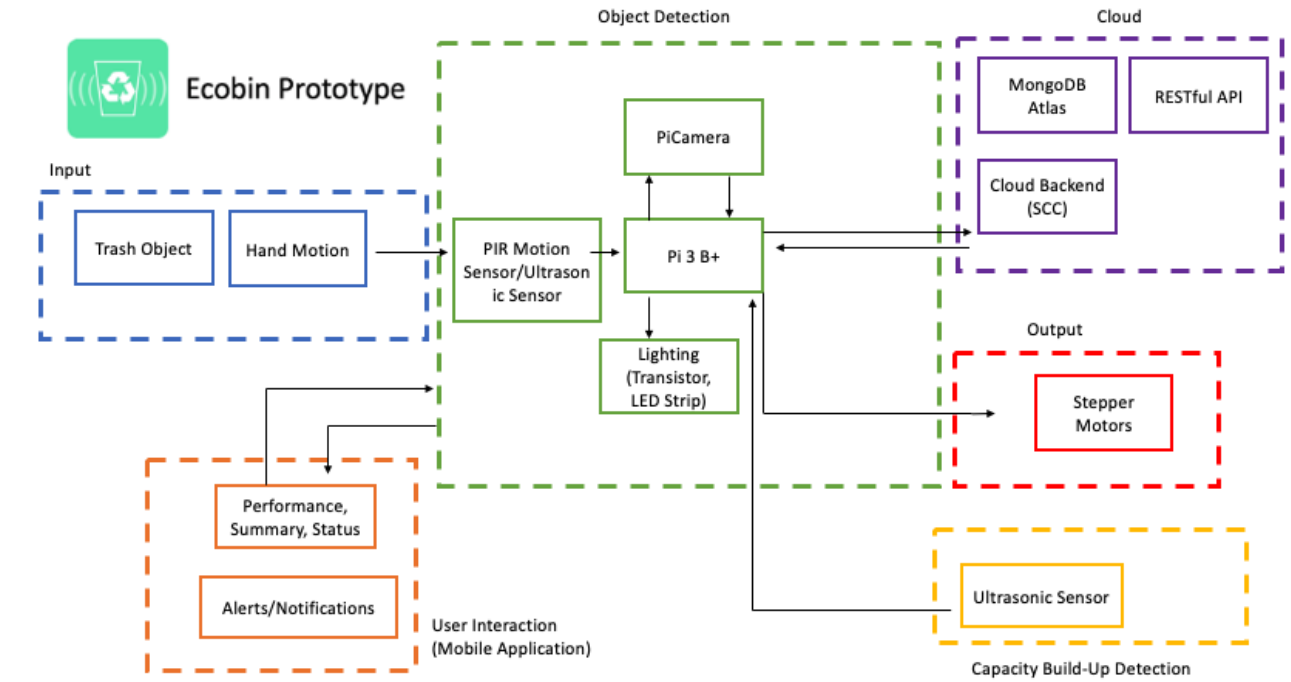


Figure 1: Block diagram of the Ecobin prototype.

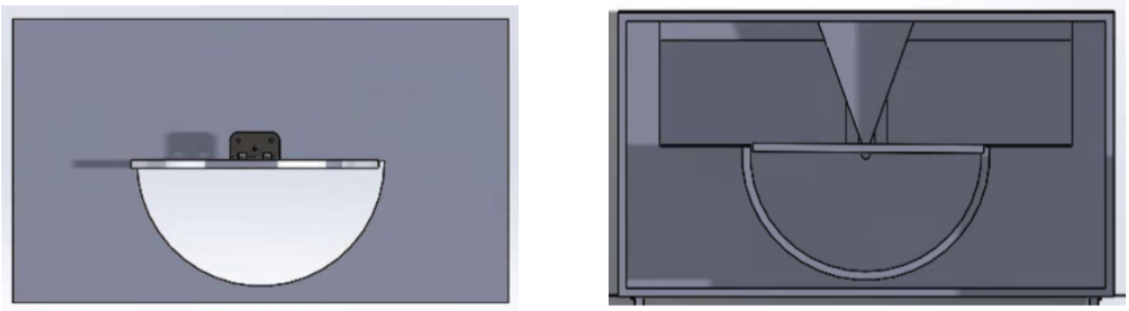


Figure 2: 2D representation of the Ecobin design (the classifier part). This is a top view. The actual design will have the same dimensions with the full functional prototype but with different material.

2.1 Hardware Setup:

Ecobin uses a Raspberry Pi 3 B+ microcontroller to relay signals to the camera, the LED strip, a motion sensor, and stepper motors. The Pi Camera captures an image of the object in the bin which is then processed into 224x224 pixels to ease the processing time in the cloud server. Ecobin's camera is positioned in such a way to minimize blind spots on the platform and noise

on the image, which could disrupt the machine learning backend. To address poor lighting conditions, Ecobin is equipped with a 12V LED strip with high luminance, operated by the Pi via a transistor switch. The team carefully placed the LED strip to avoid glare. The prototype also features a platform and an outer case designed by the Mechanical Engineering team.

As an improvement to the last prototype, the team supplied this prototype with several new sensors and a power supply unit, allowing the Ecobin to be powered using a single wall outlet connection. A new passive infrared (PIR) motion sensor now detects motion and triggers the Pi to take a picture of the object. It gives more consistent results compared to the one used in the first prototype and operates on 5V, driven directly from the RasPi. Another sensor incorporated into this prototype is an HC-SR04 ultrasonic distance sensor, which simulates waste detection buildup of Ecobin. By detecting distance from a point, it will measure the level of trash build-up within receptacles of the Ecobin. Similar to the PIR, it also operates on 5V, allowing it to be powered directly from the Pi board. Both sensors are mounted on protoboards to allow for more permanent connections than what can be achieved using solderless breadboards.

In addition to the sensors, this prototype utilizes two mechanical functionalities: automatic lid and sweeper. The automatic lid mechanically rotates based on a stepper motor controlled by an Arduino Uno. The Arduino Uno constantly checks for a signal from the motion sensor. Hence, after a hand motion is detected, the lid will open. The sweeper is supported by a NEMA-17 stepper motor which is attached with a gearbox. This motor is controlled by a Phidget motor controller, which the RasPi can control using Python. The RasPi commands the sweeper to rotate clockwise for trash, and anti-clockwise for recyclables, as seen from the top.

Lastly, the device no longer features an LCD screen mounted on the unit that displays the graphical user interface (GUI). Instead, the Ecobin GUI is now only accessible through the iOS application. In terms of power supply and wall outlet connections, a dual output (5VDC and 12VDC) power supply unit is now in use, allowing for all the components to be powered from a single wall outlet. The 12V motors and LED, along with the 5V Raspi can be powered using this new power supply unit; the Arduino Uno is powered by the RasPi.

Raspberry Pi pin numbers	Usage/Description
2	V_{CC} = 5V

6	GND
7	GPIO 04: PIR Out
11	GPIO 17: LED strips
12	GPIO 18: Ultrasonic Trigger
16	GPIO 23: Ultrasonic Echo

Table 1: Hardware Pinout

2.2 Software Setup:

2.2.1. Back-end

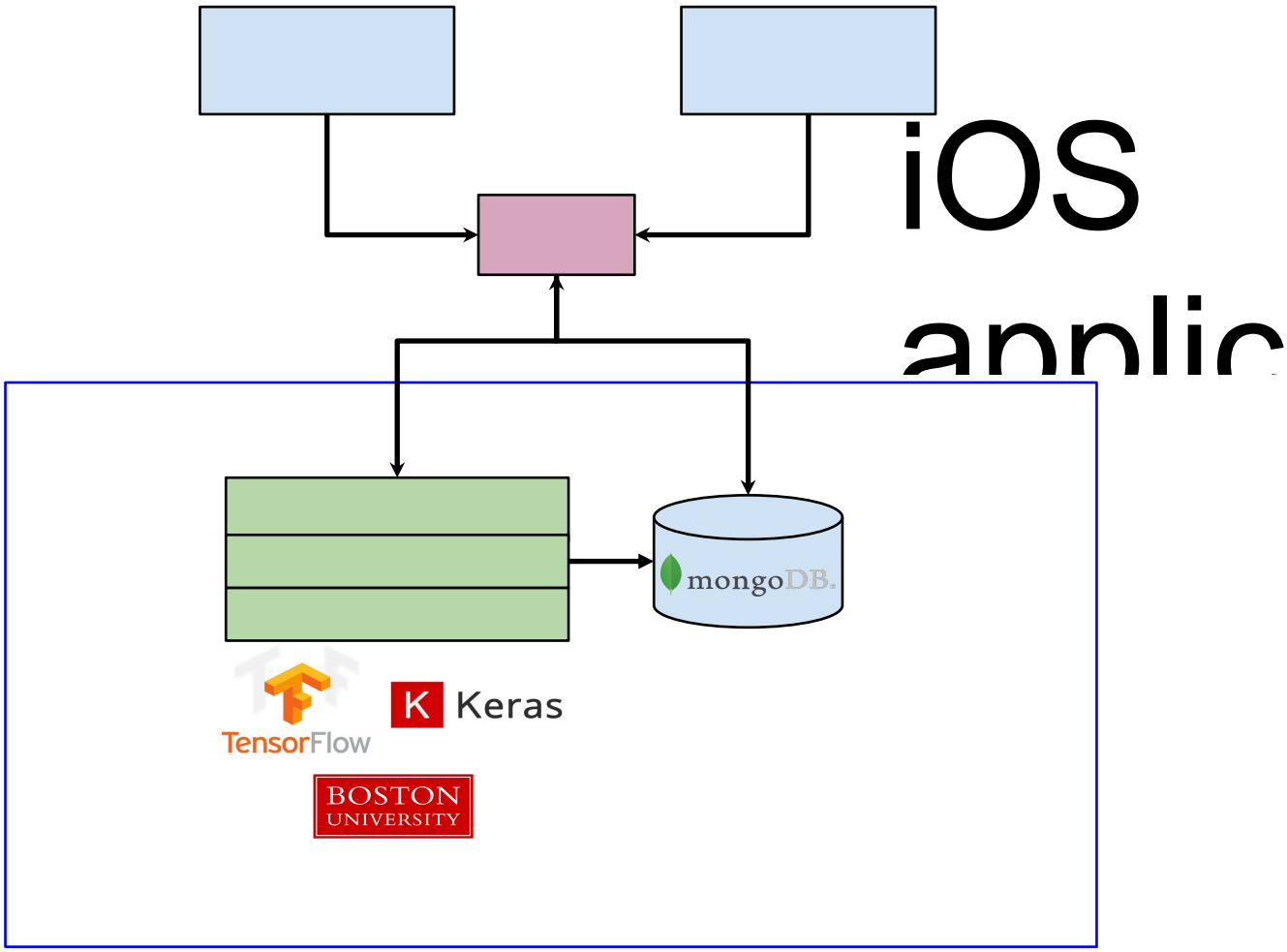


Figure 4.1 Ecobin Back End Topology

A. API

a. API integration:

Ecobin API is the primary tool for Raspberry Pi, iOS application and the backend to communicate. A architectural diagram of the API is as follows:

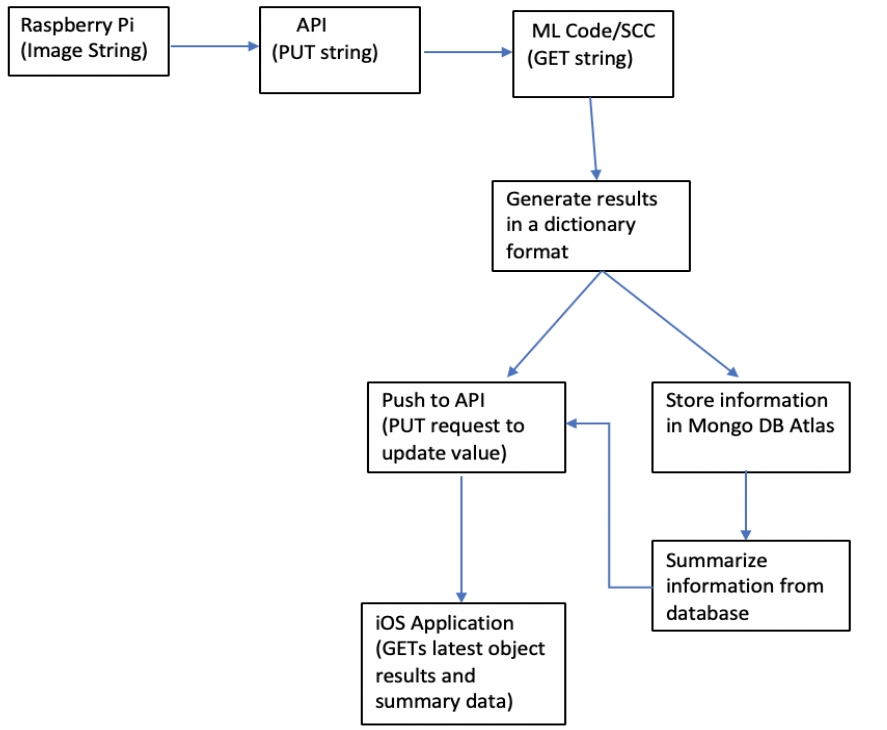


Figure 4.2 : Software block process diagram to demonstrate communication between the raspberry Pi , PyMongo Atlas Database, Shared Computing Cluster and iOS application using the API

The API functions in this order:

1. The Raspberry Pi takes a picture of an object placed in the bin and converts the .jpg into a base64 encoded string that can be stored in the API using a PUT request.
2. The API features an authentication method implemented with 'HTTPAUTH'. It requires a username and password in order to be able to access and store information from and to the API.
3. The machine learning code then pulls the encoded string information from the API using a GET request.

4. In the backend, the machine learning code then processes the image and returns a dictionary with key value pairs holding attributes: object name, accuracy, its classification type. The backend also updates the API with the processed information using a PUT request.
5. The backend file connects to the Mongo Atlas database and stores the classification and accuracy information processed from the Machine Learning code.
6. The RasPi receives information from the API, whether the object is 'trash' or 'recyclable'.
7. The RasPi outputs this signal to the stepper motor, which sorts the object.
8. The API then sends the information to the iOS application which presents interactive results to the user.

b. <u>API architecture:</u>

The API needs to have an architecture that supports the multiple POST and GET requests executed from the Raspberry Pi, SCC , MongoDB Atlas database and the iOS application.

```
classify = [
    {
        'id': 1,
        'name': 'nameholder',
        'string': 'stringplaceholder',
        'type': 'Trash',
        'accuracy': 80,
        'recyclable': 0,
        'trash': 0,
        'accuracyavg': 0,
    }
]
```

Figure 4.3 : Initialized key-value pairs for API in dictionary

From figure 4.3, it can be observed that the fields hold these values:

'name' : Name of object identified

'string' : Base64 encoded image string

'type' : whether it's Trash or Recyclable

'accuracy': Probability percentage of prediction accuracy

'recyclable' : Total count of Recyclables

'trash' : Total count of trash

‘accuracyavg’ : Average of “Accuracy”

‘CapacityRecycle’ : Capacity limit of Recyclable Bin

‘CapacityTrash’ : Capacity limit of Trash Bin

B. Database:

MongoDB Atlas is used at the data storage for the Ecobin. Atlas serves two main purposes:

A. To create a “document” in a “collection called “test” in a database called “ecobin-results” to store the classified attributes of each object tested in the ecobin

B. To perform database ‘aggregate’ functions using pipeline in order to be able summarize information stored in the database and automatically update the summary with each tested object. This information is queried by the API using the PUT request so that the iOS application can access it. The aggregate function performs two functions:

   i. Counts and number of “Trash” and “Recyclable” objects store their frequencies in a list

   ii. Perform the average of accuracies of the probability percentages of images identified and classified to provide an estimation for the rate of success of the classification.

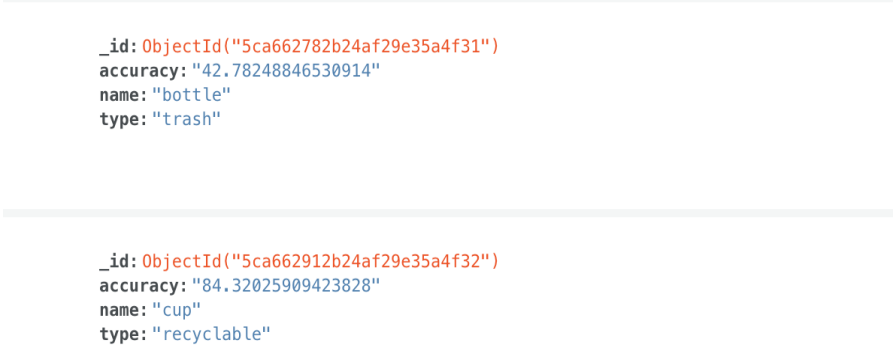


Figure 4.4 : Results stored in Mongodb Atlas

Figure 4.4 shows how the classification results are made into “documents” in a “collection” and stored into a database into MongoDB Atlas. By staying consistent with the

cloud network, MongoDB offers flexibility and reliability without sacrificing speed. The handshaking between the frontend and backend with the database takes an average of 0.03 seconds, which satisfies our prototype requirement for a fast (~2 seconds) transaction.

C. *Object Detection:*

Ecobin's object detection algorithm, which uses Keras and Tensorflow, is trained and run on the SCC. The algorithm includes a neural network designed based on the VGG16 pre-trained model. However, the last four layers were detached and modified into the following:

1. Convoluted (512 nodes)
2. Flatten
3. Dense (4096 nodes)
4. Dropout (80%)
5. Dense (12 layers)

This deep learning neural network employs concepts from transfer learning and fine tuning. Different from the latest test, it includes an additional convoluted layers to better learn the features in the images. This deep learning neural network was trained to recognize 13 classes of objects. Improving from our last test, each class now has approximately 1000 images for training and validation.

1. Recyclable: bottles, cups, metal cans, glass, general plastic containers, utensils, carton/cardboard, general recyclable items.
2. Trash: food, fruit, compostable containers, leafy greens, general trash items

The general recycling and trash items include items that belong to each perspective class but do not have enough images to be trained into their own classes. For example: general recyclable items include envelopes, wrinkled paper and egg cartons while general trash has used tissues, napkins, and etc.

For training, we utilized two Nvidia P100 Graphical Processing Units (GPUs), with 16GB of memory each. We trained the model over 25 epochs and achieved fairly sufficient cross-validation results. Ecobin also uses SGD optimizer with learning rate decay and Nesterov for better results. In the graphs below, it can be noticed that by the end of the 25th epoch, the training accuracy (95%) and validation accuracy (94%) varied by less than 1%. The loss also

converges well. Ecobin achieved this result by experimenting with multiple values for the neural networks and chose the final model. The model's weights are then saved to ecobin.h5 and ready for detecting objects from new images.

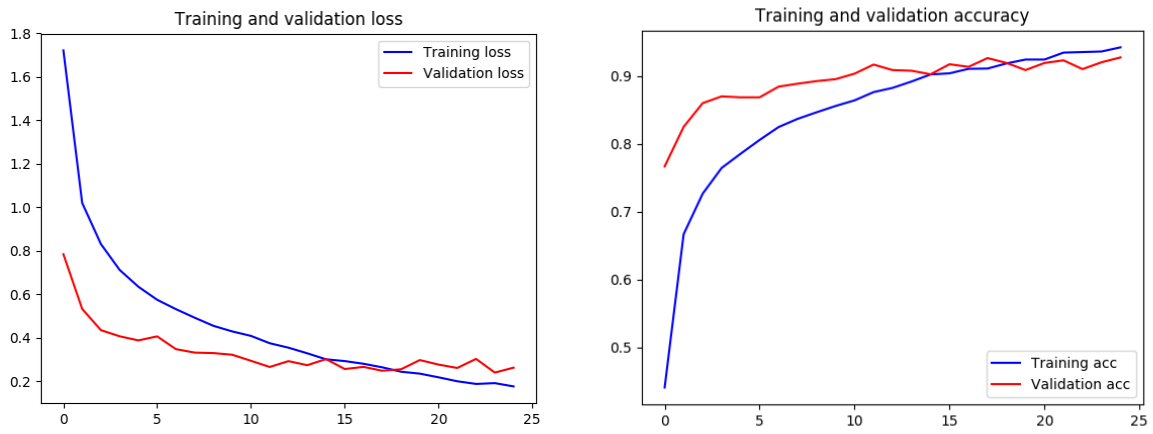
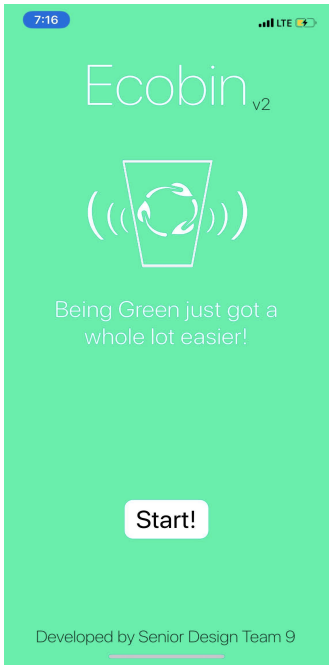


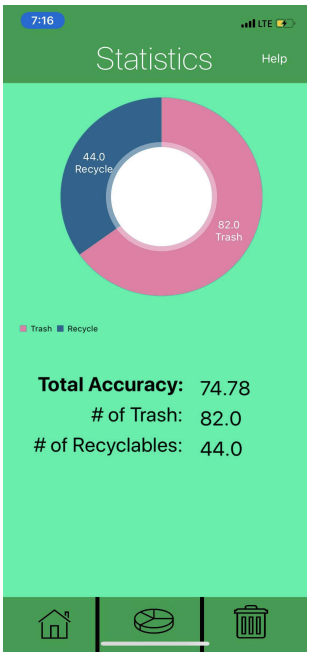
Figure 5. Loss and Accuracy over 50 epochs

The backend.py script preloads the model and every time there is a new image from the Raspberry Pi, signaled by a 'PUT' request to the API, backend.py will predict the object. If the accuracy is too low (<70%), the item is automatically classified as trash in order to reduce true negatives.

2.2.2. iOS application

The iOS application is designed for IOS 10.1.0+, using swift 4. It queries the API using URLSession with Authentication. The authentication password and username is encoded through base64 to ensure security. There are also error checks in place when the URLSession connectivity with API fails, displaying an error message and trying again. Additionally, another feature is asynchronous processing, which enables parsing and updating of the UI to be done efficiently without having to wait for serial computing.





Step 1 : Landing Page when application is launched

Step 2 : User is then directed to the information page after the application is launched on their phone

Step 3 : After user throws an item into ecobin, it gets classified and returns information on the type of object that was throw(in this case, ‘fruit’) , the classification (‘trash’) and the probability accuracy of prediction (‘81.62%’). The user can tap the refresh button at the top left corner of the page to automatically fetch information if any new object is thrown into the Ecobin.

Step 4: After the user taps on the summary button, statistics are displayed showing the user the average of the total prediction probability of the items thrown into ecobin and the number of

trash and recyclable items thrown in so far. There is also an interactive pie chart which helps the user visualise the information presented.

2.3 Pre-Testing Procedure:

2.3.1 Hardware:

1. Make sure the RasPi and sensor connections are correct based on Table 1.
2. Check the electric circuit (microcontrollers, LEDs, motors)
3. Check all the wiring, make sure none are loose. Tape the wires if needed.
4. Mount the camera on the wall next to the platform that the ME team made. The camera is supposed to be fixed in the upper right corner of the left wall.
5. The RasPi, transistor, and LED strip is to be placed on the side of the platform, which will be covered by the Ecobin housing (box).
6. The PIR motion sensor and the ultrasonic sensor will be outside the Ecobin housing.
7. Make sure the sweeper is stabilized (horizontally straight)

2.3.2 Software:

Backend: (Can someone check if this is correct?)

1. The RasPi has internet connection.
2. Make sure the computer/SCC has all the dependencies.
3. MongoDB Atlas database is set up.
4. The machine learning backend is live.
5. The API server 128.31.22.22 is up and running .
6. The iOS application is able to communicate with the API.

3. Metrics of Functional Testing

3.1 Method of Measurement

The overall performance of the prototype is measured by qualitative and quantitative metrics. The qualitative analysis, based on a set a predefined criteria, examines how well the hardware and software functionalities are integrated. The quantitative analysis measures the degree of accuracy of the object detection algorithm, an imperative feature of the Ecobin, through a score sheet.

3.2 Prototype Criteria

The criteria for a successful prototype are as follows:

1. The Raspberry Pi successfully captures an image of the object on the platform in response to a signal from the motion sensor, then displays whether the object is trash or recyclable and a confidence level on the iOS application.
2. The Raspberry Pi communicates with the API and an image of the object on the platform is uploaded to the cloud for processing in the backend and the results of recognition and classification are received back in the RasPi in < 2 seconds.
3. The recognition and classification algorithm should be able to classify whether an object is recyclable or trash with at least 75% accuracy.
4. The RasPi turns on the LED strip and captures the image of the object in response to motion detected by the PIR sensor. The LED strip turns off after the object detection.
5. Both encoding and decoding the 5MP image from the Pi Camera in base64 binary format is successful, with the resulting image being identical to the original image
6. The Ecobin has a sweeper, which mechanically pushes the object to the desired compartment. This sweeper is actuated by a stepper motor.

3.3 Score Sheet

The score sheet measures the quantitative success of the prototype. The following table was for reference during the prototype and marked by the appropriate instructors and GTAs. The prototype is successful if it has a score of at least 75%.

4. Result of Prototype Testing

The result of the prototype testing is evaluated based on the score sheet. The prototype got most of the values right as expected from this extensively trained model. The error from over reflection due to the excessively white background from the second prototype has been eliminated by changing the color of the interiors to black. The overall performance of the full functional version has significantly improved from the previous prototype, especially with an improvised machine learning model trained an expanded dataset.

Score Sheet

Test Object	**ObjectType**	**Category**	**Correct? (Y/N)**
Plastic Bottle	Bottle	**Recyclable**	Y
Granola Box	Carton	**Recyclable**	Y
Shampoo bottle	Bottle	**Recyclable**	Y
Drinking cup	Cup	**Recyclable**	Y
NyQuil Bottle	Bottle	**Recyclable**	Y
Apple	Fruit	**Trash**	Y
Soda Can	Can	**Recyclable**	Y
Body wash container	General_Plastic	**Recyclable**	Y
Kiwi	Fruit	**Trash**	Y
Lemon	Fruit	**Trash**	Y
Fork	Utensils	**Recyclable**	N
Drink Can	Metal_can	**Recyclable**	Y
Tea Box	Carton	**Recyclable**	Y
	Result → 12/13		92 %

Reflecting on the aforementioned prototype criteria, the following observations were made:

1. The system was effectively activated by passing a hand over the PIR motion sensor
2. The white LED strip lit up and the captured image was successfully identified and output onto the iOS application. For example, when a kiwi was thrown in, it said “Trash” with an probability accuracy of 85%. When a plastic bottle was passed in for identification, it correctly identified as “Recyclable” with an accuracy of 82%.

3. The Ecobin sweeper is able to mechanically rotate clockwise for “Trash” objects, and anti-clockwise for “Recyclable” objects. This allows the object to be sorted in the desired compartment.
4. From the thirteen tests run , it gave an accurate result for over 12/13 independent objects. They were accurately identified under their classified object type as well as their category. The only anomaly was when testing for the fork/spoon because it did not mechanically stay in place in the bin due to the gap between the sweeper and the ecobin base.

5. Evaluation

After running the above tests, the team was able to clearly see and identify the specific problems of Ecobin. Concurrently, members have been researching into potential solutions to these problems. The following is an evaluation of the existing problems and potential solutions.

- Problem 1: The automatic opening and closing of lid

Potential Solution: Need to adjust and secure the wiring connections between the RasPi, stepper motor, and the Arduino Uno.

- Problem 2: The gap between the spinning lid and the Ecobin base is too narrowly, allowing only small objects such as utensils fall through

Potential Solution: Working with the mechanical engineering team to request for a flap with more precise dimensions

- Problem 3: The sweeper tends to be lose, which makes the rotation inconsistent.

Potential Solution: Work with the mechanical engineers to tighten the sweeper

- Problem 4: The sweeper is positioned at a certain height. It is not leveled with the base of the platform. This prevents it from fully sorting small objects like spoons and forks.

Potential Solution: Work with the mechanical engineers to lower the height of the sweeper.

- Problem 5: The structure of the Ecobin needs to be completed and reinforced. The team needs to secure the placement of the hardware.

Potential Solution: Need to carefully position the hardware in the classifier. After that, the classifier needs to be integrated with the "bin" component of the Ecobin.

From the second prototype testing, a number of improvements and additions have been made including the training of new classes for the machine learning model, implementing an API , MongoDB Atlas as well as iOS application for display. All of these new additions work seamlessly with the integrated hardware. In the second prototype, one of the identified problems was the difficulty in adjusting raspberry camera focus when taking pictures of the object. This problem was effectively solved by positioning the camera at an angle which reduces background noise (the sweeper, the ceiling, and etc).

6. Conclusion

The full functional testing has made significant progress from the second prototype testing in terms of software function, hardware and mechanical capability. While the second prototype was able to take images using the RasPi,run images through a limited number of classes and store information in a temporary mLABS database and output information onto a Python GUI, the full functional unit is a far superior improvement. The latest unit has a better trained machine learning model in terms of number of items in dataset and number of classes of classification. It also features an API to communicate between all the devices while simultaneously integrating with a Mongo Atlas Database. Instead of the Python GUI, there is a well developed and API integrated iOS application personalized to the user to show usage statistics. The RasPi and the hardware setup have been well integrated with the software to allow for complete user functionality by just running one command(ecobin.py) for a fast, friendly and seamless user experience. The hardware wiring connections have been migrated from the conventional breadboard, into a breadboard PCB. The Ecobin now has a power supply, which is currently just powering the LED strip, but will later also power the RasPi and motors. Lastly, the Ecobin is finally able to utilize two stepper motors, which is used for the automatic lid and the sweeper.

Subsequent steps for final demo day for the Ecobin will include eliminating all errors encountered in the Evaluation section and packaging the Ecobin for customer installation. In

doing so, Ecobin hopes to maintain positive feedback from all parties involved and cater to the purpose that it first set out to achieve - to engineer a hassle-free way to go green.