Boston University
Electrical & Computer Engineering
EC464 Senior Design Project

Second Prototype 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 the second prototype testing, Ecobin successfully captured images in an ideal testing environment inside an enclosed box which was constructed by our Mechanical Engineering counterpart. This prototype aims to detect simple objects, such as plastic bottles, and classify it as recyclable or trash. The team designed and implemented an embedded system with Raspberry Pi (henceforth RasPi or Pi) which took photos using the PiCamera. In terms of software, the front-end of this prototype includes a Tkinter interface which displays the type of disposal along with classification accuracy; while the back-end employed a Tensorflow object detection model running on Boston University's Shared Computing Cluster (henceforth SCC) integrated with MongoDB (MLab) 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
- Camera Lens
- PIR Motion Sensor
- Ultrasonic Sensor HC-SR04
- LCD Screen - 3.5 inch
- LED Strips (12V, 0.9W)
- 12W Power Adapter (Pi3)
- 12DC Power Supply (LED Strips)
- Transistor (BJT PN2222A)

Software

- Object Classification (Machine Learning)
- Web Scraper (Collect datasets from Google Images)
- MongoDB

- Driver scripts for sensors and overall Ecobin functionality
- Dependencies
    - Pymongo, base64, numpy, PIL, Keras/ Tensorflow,Tkinter

Mechanical

- Object Detection Platform
- Ecobin Housing/Box
- Camera Mount/Stand
- 3D printed Camera Lens Mount

Overall Architecture

Figure 1 describes how each of Ecobin's module interact 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 platform for the object detection, as well as a housing(box) to keep the environment enclosed. The semicircle is the platform where the object is placed. The camera, RasPi, and LED strip will be placed on the left side of this platform. The motion sensor and ultrasonic sensor will be prepared outside of this housing.

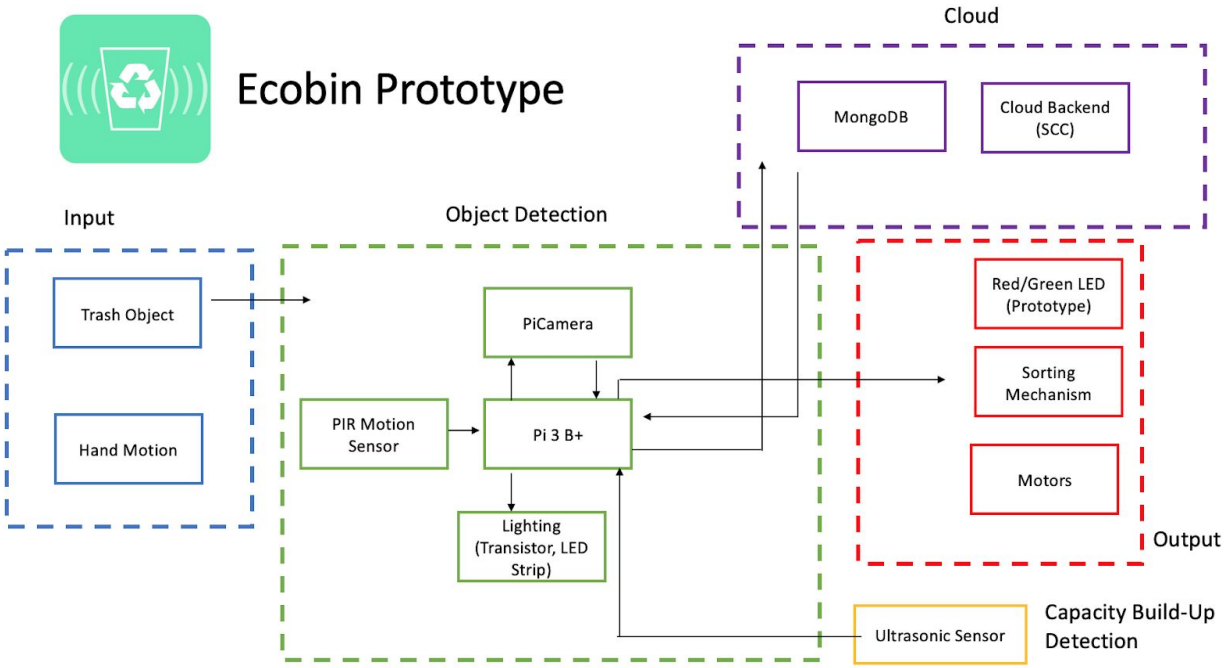

Figure 1: Block diagram of the Ecobin prototype. For this prototype, the RasPi is interfaced with MongoDB

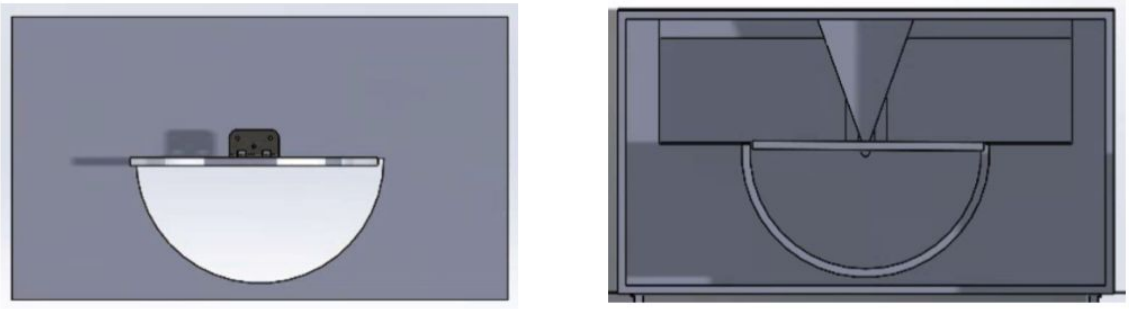

Figure 2: 2D representation of the Ecobin design (the classifier part). This is a top view. The actual design will have the same dimensions with the prototype but with different material.

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 an ultrasonic sensor. The PiCamera captures an image of the object in the bin which is then processed from 16:9 aspect ratio into the 3:2 aspect ratio. Additionally, it is resized into 224x224 pixels to ease the processing time in the cloud server. Ecobin's camera was positioned to avoid any blind spots from the platform and any unnecessary noise, which could disrupt the machine learning backend. To address poor lighting conditions, Ecobin is equipped with a 12V LED strip with high luminance, powered from an external source and operated 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. 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.

In addition to the sensors, the second prototype displays an Ecobin graphical user interface(GUI) on the desktop monitor. For the final product, it will be either be displayed on an onboard LCD screen and/or on the iOS application. The test plan mentioned the use of a wide angle lens on the Pi Camera to help minimize the distance required to capture the image of the object. However, this lens was removed shortly before prototype testing as it malfunctioned and caused the Pi camera to take blurry images.

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

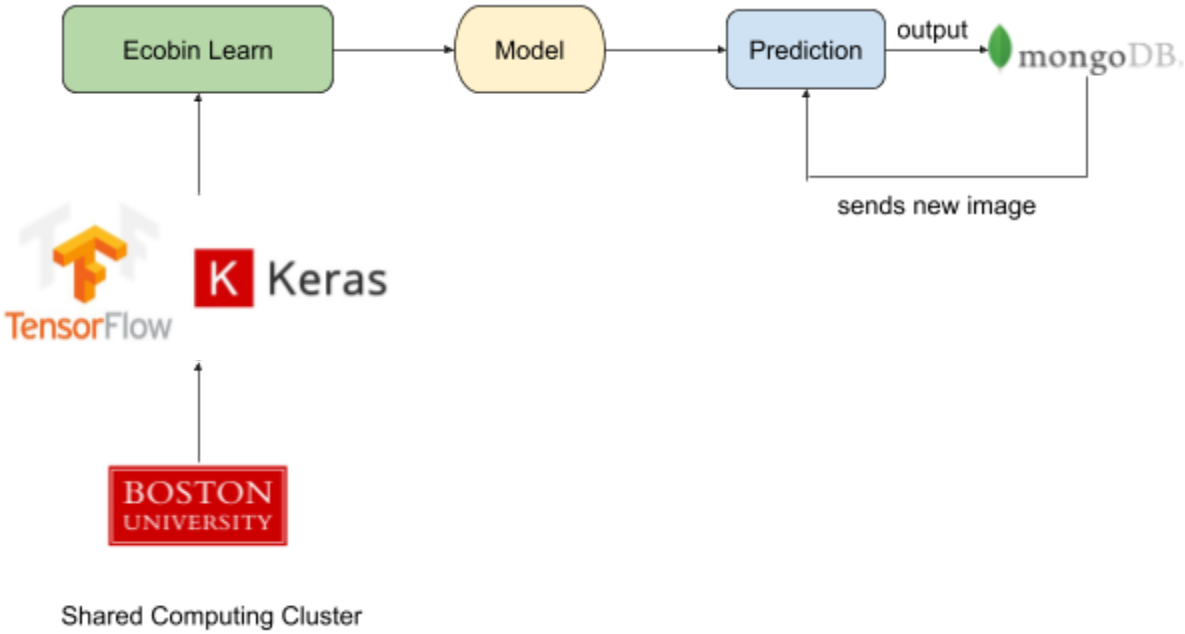

Figure 4.1 Ecobin Back End Topology

a. Database

Using mLab, a cloud MongoDB database enables the Ecobin system to access both user data and the classification results from virtually anywhere. 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. Figure 4.2 illustrates the document in the MongoDB that is used primarily for storing the encoded image taken by the Raspberry Pi, as well as the result classified from the backend script. The flag field is used to let the backend/frontend know whether or not the data has been successfully updated. After the necessary updates are made, the backend will then flip the flag back to 1, and will return the two key variables, *result* and *percentage*. The variable *result* will return a specific object that the

```
{
    "_id": {
        "$oid": "5c760183118d262d045c35e2"
    },
    "user_id": "temp",
    "img": "",
    "flag": "0",
    "result": "",
    "percentage": ""
}
```

Figure 4.2 MongoDB Architecture I

backend classification processes, such as ‘paper’. In order for the user as well as Ecobin to understand if this specific object is recyclable or not, the post processing of this keyword is performed in the Raspberry Pi, which then returns an additional variable, “Recyclable” or “Waste”.

Figure 4.3 MongoDB Architecture II

Similar to the *temp* document in *Figure 4.2*, the weekly trash data is stored in the same MongoDB network, labeled as ‘User1’. The latest trash data is stored as a field corresponding to the week it was recorded, which is then added to the end of the field with the special character separator → *. In the long term, this would be a useful field to keep, as users may learn about the following:

1. Ratio of recyclable trash vs general waste they have produced in a given time frame
2. Most consumed object (ie. Plastic Bottles)
3. Local trash data (ie. What users are throwing away in Boston)

Analysis in fields such as the user’s most consumed object may help the user understand their overconsumption on the particular product, helping them reduce consumption and/or switch to a more eco-friendly alternative. Such data will be analyzed and provided in the iOS application that is currently under development.

b. 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, based on the VGG16 pre-trained model, but differs from VGG16 in the last four layers:

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

This deep learning neural network, which embraces transfer learning and fine tuning, was trained with 12 classes with 450 images each:

1. Recyclable: Plastic bottles, plastic containers, paper, metal cans, glass, forks, spoons, cups
2. Trash: Apples, oranges, bananas, leafy greens

For training, we utilized two Nvidia P100 Graphical Processing Units (GPUs), with 16GB of memory each. We trained the model over 50 epochs and achieved positive cross-validation results. In the graphs below, it can be noticed that by the end of the 50th epoch, the training accuracy (96%) and validation accuracy (93%) varied by less than 5%, and likewise for the loss. Ecobin achieved this result by experimenting with multiple values for the neural network. The model's weights are then saved to ecobin.h5 and ready for detecting objects from new images.

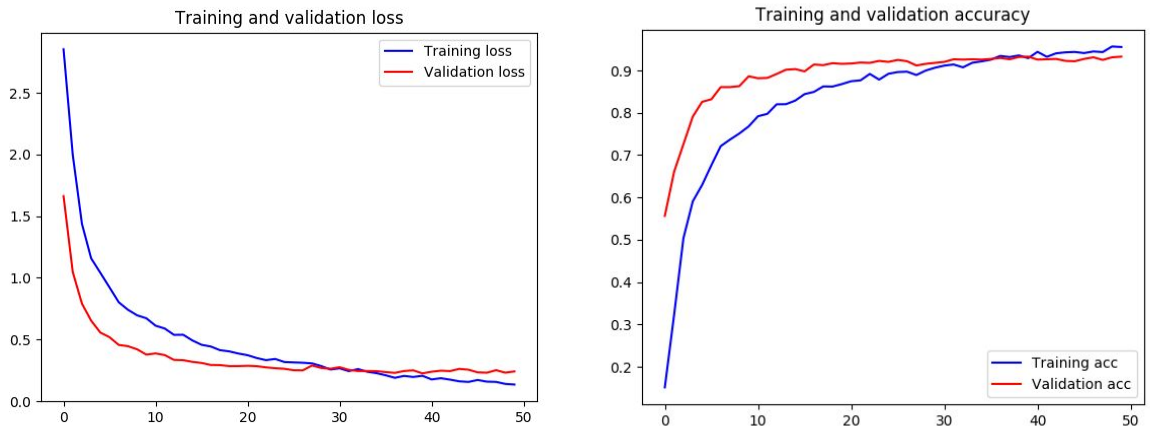

Figure 5. Loss and Accuracy over 50 epochs

After loading the model, a prediction script constantly checks for a flag in MongoDB to retrieve new images. When the signal is positive, meaning that the binary image data has been successfully updated in the database, it converts the image from binary format to JPEG format and predicts the object and inserts into MongoDB the object type as well as the confidence value. These values in MongoDB will later be used by our user interface on the Raspberry Pi and iOS application.

2.2.2. Front-end

The stored values in MongoDB are then passed into the Raspberry Pi for final processing and output into the GUI Display. The machine learning model returns the name of the object identified as well as the accuracy probability of prediction of the given object. The Raspberry Pi receives this information from Mongodb and feeds the two inputs into a script which classifies the identified object into either “Recyclable” or “Trash” based on predefined lists.

The identified classification and the classification accuracy are passed into the python GUI script. The GUI script is made using the Tkinter library in Python along with tk_tools for animation. The current GUI developed for the computer monitor screen also works with our 3.5’’ LCD display.

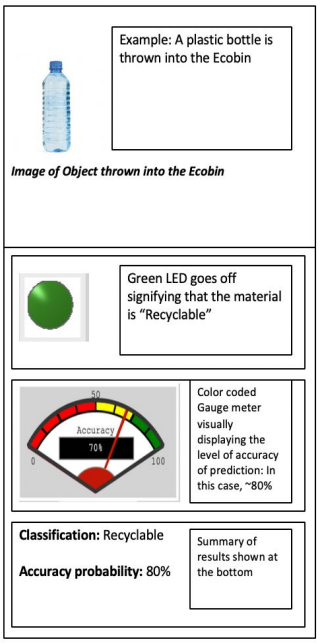

Figure 6: Ecobin GUI Output

In addition, the team is also developing an iOS application to output user data and trash recycling statistics directly onto the user’s smartphone. The current design of the GUI display is similar to Figure 6 on the right.

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 (resistors, LEDs, ground, power supplies, transistor)
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.

2.3.2 Software:

Backend:

1. Maintain MongoDB's stability
2. [On SCC] Run backend.py to activate Tensorflow and deep learning model
    a. Ensure that the prediction value and confidence are corrected put into MongoDB
    b. Ensure 99th percentile latency

Raspberry Pi and front end: Run ecobin.py

3. Metrics of Prototype 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 Ecobin GUI, which pops up on the monitor.
2. The Raspberry Pi communicates with the Ecobin API and the 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

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 higher score than the expected score, based on testing of 12 images. The error was from overly reflective white background surface of the Ecobin interiors. When the objects were tested without such an interior, the classifications returned results with absolute accuracy, as the deep learning network is proficient in object detection. Regardless, the overall performance of the prototype has been significantly improved from the previous prototype, especially with a newly built machine learning model.

Object	**Category**	**Correct? (Y/N)**
Metal Can	**Recyclable**	Y
Red Cup	**Recyclable**	Y
Green Cup	**Recyclable**	Y
Metal Spoon	**Recyclable**	Y
Metal Fork	**Recyclable**	Y
Pair of Scissors	**Recyclable**	Y
Transparent Cup	**Recyclable**	Y

Banana	**Trash**	N
Orange	**Trash**	N
Apple	**Trash**	Y
Leafy Green	**Trash**	Y
Crumpled	**Recyclable**	Y
Result → 10/12		83 %

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 terminal screen. For example, when a pair of scissors was passed in for detection, it said "Recyclable" with an probability accuracy of 70%. When a plate of leafy green kale was passed in for identification, it correctly identified as "Trash" with an accuracy of 84%.
3. Following the example of the scissors, the algorithm tried to match the identified classes to the recyclable or trash list and given its material. It identified them as "Recyclable" and output the probability accuracy on to the output GUI with a gauge slider screen.
4. From trained twelve tests, it gave an accurate result for over 10/12 independent objects. When the failing objects were tested with a higher resolution and appropriately positioned camera, they were also correctly identified according to their actual classifications (i.e. "Trash" for Banana and "Trash" for Orange).

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 background of the Ecobin was white and mistakenly classified as "paper".
  Potential Solution: Applying bit mask to color images can help to isolate the image itself from its background.

- Problem 2: The Raspi Camera was installed at an angle which captured flash glare from the LEDs.
  Potential Solution: Working with the mechanical engineering team to build this support system for optimal positioning. Combining this with the above bit mask technique would yield better pictures for detection.

- Problem 3: Limited classes (twelve classes) in training dataset. At the moment, it works well for over twelve distinct classes of categorization.
  Potential Solution: Training with a wider range of classes.

- Problem 4: LCD Screen output does not turn on when the computer screen HDMI is on display. With one Raspberry Pi 3B+ model, it is either only possible to output it to the HDMI or the 3.5 inch TFT LCD Display.
  Potential Solution: To display results onto LCD screen, we'd either need another Raspberry Pi operating it or VMC into the screen from the desktop.

6. Conclusion

This second prototype has made successful and significant progress in terms of core software functioning and hardware capability from the first prototype. While the first prototype was able to detect images and do a basic classification of either recyclable or trash using pre-trained VGG16 models while using the desktop computer as a pseudo "cloud" environment, the second prototype uses fine-tuned deep learning neural network and transfer learning to classify the images supplied. The new model achieved much higher accuracy in an ideal testing environment. Additionally, image detection has been migrated to Boston University's SCC with

GPUs enabled for ultra-fast processing. MongoDB database supports the communication between the Pi and cloud and outputs information to the Python GUI display. Furthermore, compared to previous prototypes, to achieve a modular software design, this version of Ecobin utilizes one script “Ecobin.py”, which integrates multiple modules including MongoLab Database, object detection on the cloud and the output classification GUI

Subsequent steps for Ecobin include eliminating all errors encountered in the Evaluation section and implementing new features. An iOS application will be developed for robust and efficient user interaction. It will include user statistics as well as analysis so that the user can keep track of their recycling performance. If needed, the team will build a simple RESTful API. The database will be more diversified as project development carries on. In doing so, Ecobin hopes to maintain positive feedbacks and improve satisfaction from its client.