Smart Plastic Container Recycling System

ECE 445 Senior Design

Final Report

12/6/2023

Team 7
Jason Wright, Jennifer Chen, Smruthi Srinivasan
Jeff Chang

Abstract

Climate change is one of the greatest problems of our time and it is our responsibility as a society to do our part to limit the implications of global warming. One of the easiest ways to tackle this problem is to recycle. The United States lacks education regarding proper recycling compared to other countries such as Sweden. Our project aims to tackle a portion of all recycling: plastics. This paper explains the design process and overall results of our Smart Plastic Container Recycling System along with the motivation behind it.

Table of Contents

1. Introduction **1**
1.1 Problem 1
1.2 Solution 1
1.3 Visual Aid 2
1.4 High Level Requirements 2
2. Design **3**
2.1 Block Diagram 3
2.2 Physical Design 4
2.3 Control Subsystem 4
2.3.1 Overview 4
2.3.2 Design Decisions 5
2.3.3 Results 5
2.4 Power Subsystem 6
2.4.1 Overview 6
2.4.2 Design Decisions 6
2.4.3 Results 6
2.5 Sensor Subsystem 7
2.5.1 Overview 7
2.5.2 Design Decisions 7
2.5.3 Results 7
2.6 User Interface Subsystem 8
2.6.1 Overview 8
2.6.2 Design Decisions 8
2.6.3 Results 9
2.7 Software Design 9
2.7.1 Overview 9
2.7.2 Design Decisions 10
2.7.3 Results 11
3. Cost and Schedule **12**
3.1 Cost Analysis 12
3.1.1 Cost of Materials 12
3.1.2 Cost of Labor 12
3.2 Schedule 13
4. Conclusion **15**
4.1 Accomplishments 15

4.2 Uncertainties 15
4.3 Ethics and Safety 16
4.4 Future Work 16
References **17**
Appendix A: PCB Schematics **19**
A.1 Microcontroller Schematic 19
A.2 Stepper Motor Driver Schematic 20
A.3 Buck Converter and LDO Schematic 20
Appendix B: Requirements and Verification Tables **21**
B.1 Control Subsystem Requirements and Verification Table 21
B.2 Power Subsystem Requirements and Verification Table 21
B.3 Sensor Subsystem Requirements and Verification Table 22
B.4 User Interface Subsystem Requirements and Verification Table 22

1. Introduction

1.1 Problem

Recycling maintains a lot of benefits for the community around us, especially as we aim to tackle the effects of climate change. The benefits of recycling are countless, but recycling works to reduce waste and pollution, conserve energy and natural resources, and create and support jobs domestically. Unfortunately, a lot of people struggle with determining which materials can be recycled and where they can effectively recycle them due to the recycling infrastructure in the United States being outdated [1].

While other countries have effectively taught their population how to correctly recycle their items from a young age, the United States lacks education on proper recycling. This leads to contamination of other recyclables, ultimately preventing them from being recycled. In fact, estimates show that over 50% of waste ends up in landfills instead of being recycled [2]. We usually think of plastics as recyclable, but depending on the jurisdiction, some plastics may not be able to be recycled. If they are accidentally recycled, they run the risk of contaminating all of the other recyclables, which is a mistake we can no longer afford as the potential effects of climate change loom ahead.

1.2 Solution

Our solution to this problem is a device with an imaging system that reads the symbols printed on plastic containers. This device will be mounted on a user's trash system. We will have a camera sensor that works with a machine learning model (VGG16) to read the numbers printed on the plastic container and a GPS sensor that determines the location of the user using latitude and longitude coordinates. That information will be utilized to determine if that specific plastic container can be recycled in the user's location using RecycleNation API. Once the determination has been made, we will have a sorting actuator that places the plastic in the proper bin and a web application explaining more about the type of plastic and display the recycling centers nearby (if any).

1.3 Visual Aid

Figure 1: Visual Aid

1.4 High Level Requirements

1. Camera detects the plastic being positioned in front of it 95 ± 1% of the time and system is able to correctly identify the symbol listed on the plastic container 95 ± 1% of the time
2. GPS location sensor determines the user's location within a 10 meter radius and pulls the data regarding recycling in that area
3. System correctly determines 95 ± 1% of the time if the container is recyclable or not and places the container in the proper bin, web application displays specific information about plastic being recycled/specific location centers that accept this type of plastic.

2. Design

2.1 Block Diagram

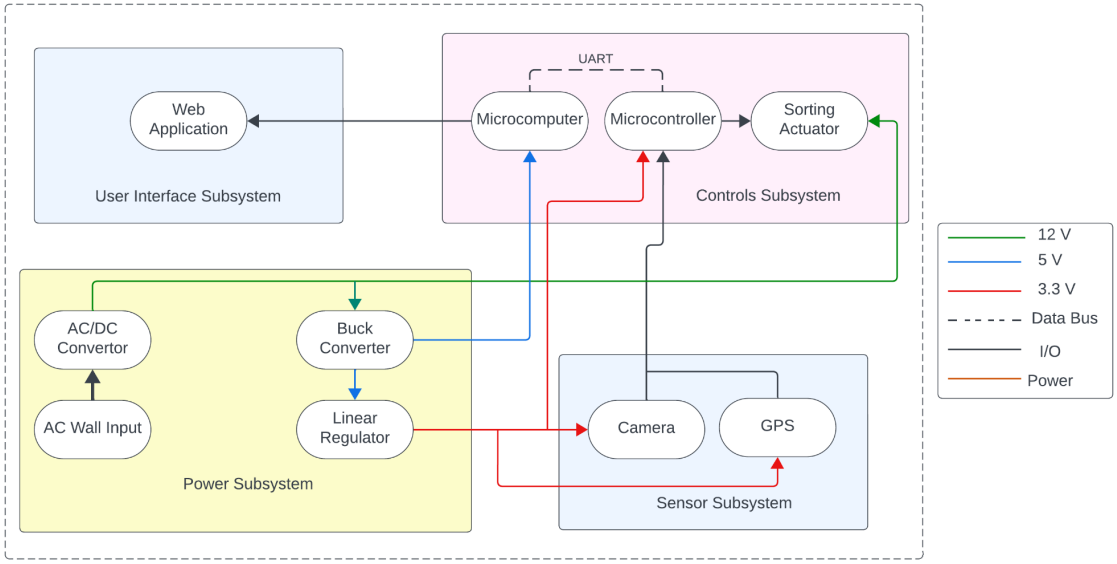

Figure 2: Initial Block Diagram

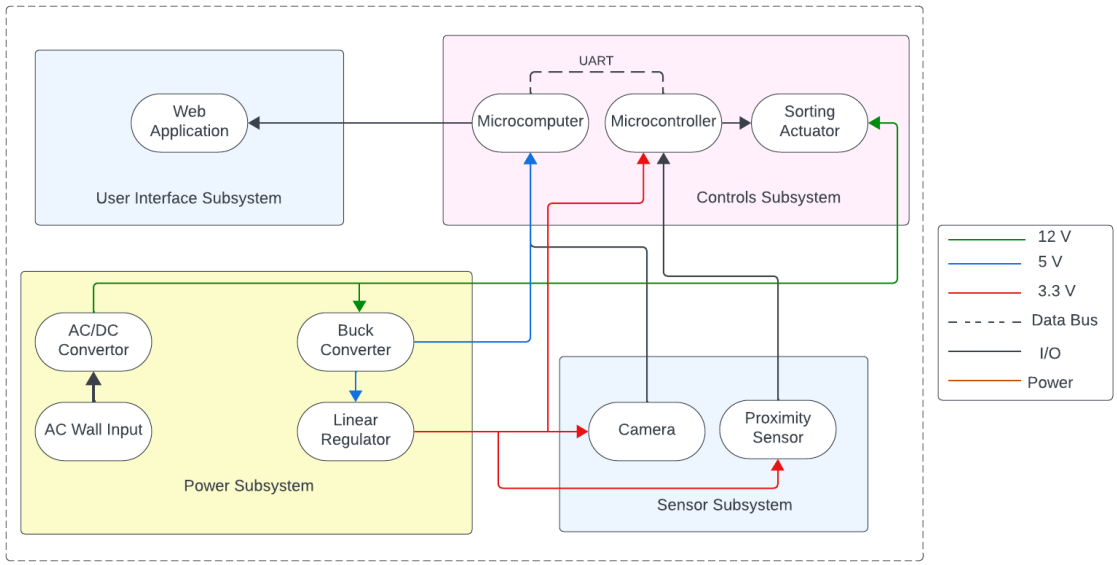

Figure 3: Final Block Diagram Design

Based on the feedback we received from our Design Review session, we made modifications from our initial block diagram in figure 2 to the final block diagram shown in figure 3. The following sections discuss these design changes in depth.

2.2 Physical Design

Figure 4: Physical Design

The sorting mechanism of our system is a simple platform that the user places the plastic object onto. Our camera sits several inches back from where the object is placed in order to view the entire item. The PCB and other components are encased and mounted behind the user side of the system next to the camera. All of this is mounted above two small trash cans, one on either side of the sorting mechanism. Once the system has decided if an object is recyclable, our 12V stepper motor will tilt the platform through the use of a chain attached to an axel. The use of a stepper motor allows us more precise position control, allowing it to return to the level position after each sort. Additionally, the chain system allows us to customize the sensitivity of our rotation through the use of simple gears. Our motor contains 200 steps per revolution, giving a base precision of 1.8° ± 5%, which we can increase as needed.

2.3 Control Subsystem

2.3.1 Overview

The control subsystem consists of our microcontroller (STM32F103C8T6), a microcomputer (Raspberry Pi 2), and our 12V stepper motor and driver. The motor and driver directly take 12V input from the wall converter, the microcomputer takes 5V input from the buck converter [3] [4], and the MCU will use 3.3V from our LDO [5] [6]. The MCU has 3 functions: data collection from the sensor subsystem, communication with the microcomputer, and control of the stepper motor. The stepper motor physically sorts items by tilting the platform where items are placed once they have been identified as recyclable or not from the other subsystems. The Raspberry Pi will be loaded with the trained image recognition software and be able to connect to a web

application for the purpose of retrieving location specific recycling data and providing information to the user.

2.3.2 Design Decisions

The control subsystem consists of the microcontroller, Raspberry Pi, and the stepper motor with its driver. For our design, the microcontroller did not have to have strong computing power, but did need to interface with the Raspberry Pi through UART, the programmer through JTAG, the stepper motor driver through GPIO (General Purpose Input/Output) pins, and have one more GPIO pin for the proximity sensor. To meet these requirements, the STM32F103C8T6 MCU was chosen. The A4988 driver was chosen to allow more precise control of the stepper motor. Integrating it into our system greatly simplified the programming process, allowing motor control to be completed through 8 logic signals driven by the MCU. These signals allow us to enable sleep mode, control direction, step using simple pulse signals, and have the ability to enable microstepping for more precise positional control if desired.

The physical circuit for the motor driver was designed based off of an example circuit in the A4988 datasheet. To map this circuit to our design, only a few components needed modification. First, the sense resistors needed to be selected based on the maximum current draw of our motor. Equation 1 describes how this resistance can be calculated. This is derived from the driver internals available on the datasheet.

$$R = \frac{V_{ref}}{8 * I_{max}} \quad (1)$$

Using a reference voltage of 1V with a 350 mA max current draw, the necessary sense resistance is 0.36 ohms. 0.35 ohm resistors were actually used in practice due to availability. The off-time of the driver can also be customized with a pull-down resistor on the ROSC pin, but we determined the default off-time of 30 microseconds was more than adequate for our application, so this pin was set to ground (this enables the chip's internal off-time). The capacitor values for this circuit did not need to be modified.

2.3.3 Results

Testing of the control subsystem began with running a simple program to blink an LED by toggling a logic signal being sent to a single GPIO pin. Through STM32CubeIDE, we were able to successfully connect to and run the code on our custom circuit board. We next wrote a program to enable our motor driver, select a direction, step 25 times (45 degrees), switch directions and move back to the level position. After some experimentation with pulse lengths and delay, we were able to successfully operate our motor while using the proximity sensor as a trigger.

To complete our system integration, the Raspberry Pi would serve as the middle ground between our user interface and microcontroller, allowing the MCU to control the motor according to the

determination made by the machine learning model and UI. Unfortunately, the Raspberry Pi that our team acquired appeared to be faulty out of the box, unable to be loaded with any operating system, resulting in the loss of an integral connection for our system. Without additional time to obtain a new board, this piece of the control subsystem had to be omitted for demonstration purposes.

2.4 Power Subsystem

2.4.1 Overview

The power subsystem consists of three major components. First, 120 VAC power is rectified to a 12V DC value by a 30W converter (Qualtek QFWB-30-12-US01). The 12V power is passed directly to the stepper motor. It is also sent to a buck converter that steps the power down to 5V. The 5V power is sent directly to the raspberry pi, and to a linear, low-dropout regulator (LDO). The LDO steps the voltage down one more time to 3.3V which can be passed to the microcontroller, camera, and motor driver. Both DC-DC converters need exterior resistance, capacitance, and inductor values according to their individual datasheets.

2.4.2 Design Decisions

The stationary nature of our project naturally led to the decision to power it through standard AC wall outlets. To keep our custom power converters as pure step-down converters, we started with a pre-purchased 12V wall adapter. A simple linear regulator could not be used to step the 12 V down to 5 V due to the high current draw under load of the Raspberry Pi (1.5 A maximum). Instead, a customizable buck converter was selected. By following the guidelines set by the SIC402 datasheet, resistor dividers, capacitors, and an inductor were selected to operate at 400 kHz switching frequency, keep ripple voltage under 100 mV, and have current ripple under 40% of maximum current. The 3.3V components are all much lower power (expected total draw under 200 mA), so a 5V to 3.3V LDO could be used without many other considerations. Additional space for capacitors was added to the design to ensure that voltage ripple could be reduced after testing if necessary.

2.4.3 Results

Testing began by sending 12 volts to the circuit board and using a multimeter to read the 5V and 3.3V outputs through test pins. At first, the 5V pin was actually reading only 3.24 volts, and the 3.3V pin was reading 2.2 V. While attempting to gather more information to debug, the buck converter stopped responding at all, no longer giving any output at all or drawing any current. This was the most difficult component to solder including 3 underside pads, so we believe a soldering issue underneath the chip caused part of it to burn out during testing. To work around this component failure, a second wall adapter was added to our design to power our 5 volt components. Because no current was being drawn through the failed buck converter, this second voltage source could be connected to the same 5V test pin we were previously reading from in

order to power part of our system. Once constructed, a voltmeter was then used to confirm that our linear regulator was outputting a clean 3.3 volts. Ultimately, this solution was sufficient and the MCU and stepper motor were both successfully operated through our custom circuit board.

2.5 Sensor Subsystem

2.5.1 Overview

The sensor subsystem contains the hardware necessary to signal a camera to capture an image of a recycling symbol on the container. The camera is positioned to take an image of the user's plastic container that is placed on the tilting platform and the image data is transmitted to the Raspberry Pi. A proximity sensor is located on the tilting platform to signal the camera that an item has been placed and an image should be captured for further classification.

2.5.2 Design Decisions

As shown in the original block diagram and information presented above, the sensor subsystem was supposed to consist of a camera and GPS, both of which communicated directly to the microcontroller. This design was modified to have the camera directly communicate with the microcomputer in order to streamline the design and cut down on the amount of data transfers necessary. The GPS was also removed from the final design because it was redundant to constantly ping a user's location when the system would typically remain stationary. A proximity sensor was also added to this subsystem so that the picture would be captured only if there was a container placed on the platform. The goal of this addition was to reduce energy consumption required for the camera to constantly be on and capturing images.

2.5.3 Results

Our final design of the sensor subsystem differed from our initial design. Unfortunately, we were unable to test and verify our camera sensor because it was supposed to be directly connected to the Raspberry Pi. Since that was not working, we were not able to run our camera. A potential idea for future work would be to use a computer's web camera, eliminating the need for a Raspberry Pi. Instead of fulfilling the GPS requirement using a GPS sensor, we retrieved a user's location using their IP address, which we will discuss more in depth in the next section. We were able to verify the proximity sensor's functionality by programming our microcontroller to light up an LED when an item was placed on the platform in front of it. We were able to further confirm that the proximity sensor worked by programming our microcontroller to move the stepper motor to one side upon placement of an object on the platform.

2.6 User Interface Subsystem

2.6.1 Overview

The user interface consists of a full stack web application where information about the specific plastic the user is attempting to recycle is displayed, in addition to the locations of specific recycling centers that accept the type of plastic inputted. The user interface receives this information from the machine learning model, which was intended to be housed on the microcomputer, and uses the classification to display the necessary information. The left side of the page displays the plastic type that is pulled from our microcomputer software and information about it and the right side of the page displays a map that pinpoints the locations using the Geoapify API [8] and Google Maps API [9]. The web application was built using React JS.

2.6.2 Design Decisions

The user interface consists of the plastic symbol classification, the description of the plastic type and a map that includes markers of nearby recycling locations based on the user's location. We used the Google Maps API for the map, and the Geoapify API for the recycling locations. For the UI design, we used the React framework for the front-end and Django on the back-end of the web application. We chose to use React for the UI design as it has a wide range of component libraries, allowing for a customizable and responsive interface. Originally, we were going to receive information regarding image classification from the Raspberry Pi. Since we were not able to use the Raspberry Pi and load our model onto it, we integrated Django on the back-end of the web application. Django is a Python based framework, which made it compatible with the machine learning model we implemented in PyTorch and could be integrated with the React user interface.

2.6.3 Results

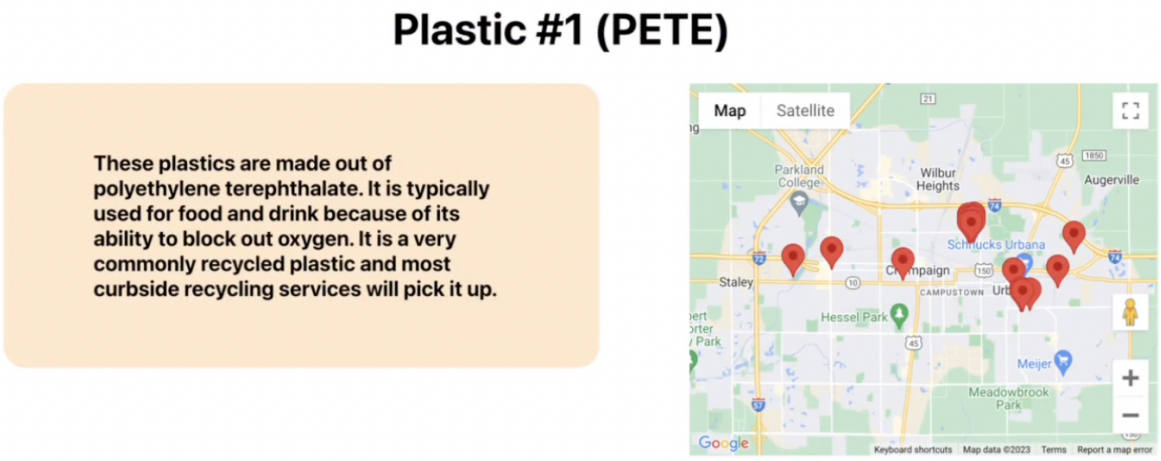

Figure 5: Web Application Front End

Figure 5 above shows our final web application design. We have a header that displays the plastic type and the text box on the left provides more information about the plastic type that the container was classified as [7]. The right displays a map centered in the Champaign-Urbana area and the red markers denote recycling centers in the area. We were originally going to use the RecycleNation API to find these locations, but we were not able to receive access to it so we have to pivot to using the Geoapify Places API. The map rendering was done using the Google Maps API. The map's location is dependent on the user's IP address, which we use to extract the user's location. We also fulfill the requirement that we display centers within a 30 mile radius (not necessarily shown above) since radius is a parameter in the API call. On the backend of the application, we were able to integrate the machine learning model using Django, as shown in this video. This is especially useful if we want to continue improving the system because we could completely remove the Raspberry Pi and instead connect the model from the backend of the application to the microcontroller.

2.7 Software Design

2.7.1 Overview

The machine learning model we selected is the VGG16 Model [10]. This model is a deep Convolutional Neural Network that consists of 16 layers of convolutional layers, max pooling layers and dense layers. Our design uses a model that is pre-trained on ImageNet [11] and we finetuned the model on the Plastic Identification Symbol dataset [12]. This model is then loaded onto the Raspberry Pi and fine tuned once connected to the controls subsystem.

The image that the camera sensor takes will be our input image into the network. As a requirement of our control subsystem, we will verify that the image is 244x244 pixels beforehand. This image then passes through a series of convolutional layers of 3x3 filter size that learn specific features of the image and max pooling layers that select the maximum value of our image feature map. This works to increase efficiency and reduce computational complexity, which is important since a downside of using the VGG model is that since it has 16 layers it can take a long time to train. It then goes through 3 fully connected layers that work on classifying the image and outputting the category it most likely belongs to.

An example classification for our use case is first, the camera captures an image of a water bottle's recycling symbol (Plastic #1). The image will then be sent to the microcontroller and then the microcomputer where it will be inputted into the VGG model network. The image will pass layer by layer as the network learns its features and performs a classification. The final output will be a list of class names and the corresponding probabilities that this image belongs to the specific class, with Plastic #1 being at the top with the highest probability.

2.7.2 Design Decisions

We selected the VGG 16 model for this project because it can reach a test accuracy of 92.7% for the ImageNet dataset (consisting of 14 million training images with 1000 classes) and its architecture allows for a higher accuracy for image classification [14]. We opted to use PyTorch to implement this since it had a pre-trained model. Initially, we were going to train the model only on the Plastic Identification Symbol dataset (contains 685 data points), but later realized that this dataset was not very suitable for our purpose because the symbols in it were all close up and the images our camera would be taking were not as close up. We considered a R-CNN model or even using unsupervised learning but it would have been difficult to implement due to limited time and machine learning knowledge. Ultimately, we chose to train on the Plastic Identification Symbol dataset first, take images that would comprise our own dataset, and then fine tune the model on the dataset we created. Our dataset contained approximately 60 images spread across 8 different classifications. However, a limitation of this dataset as well as the Plastic Identification dataset was that some plastics were harder to find so those plastics had less images.

In our initial design, we wanted to load the model onto the Raspberry Pi and finetune the model again using pictures using the camera sensor. However, we were unable to load the model onto the Raspberry Pi so we chose to integrate the model with the backend of the web application. This can be extended in future work to streamline the design.

2.7.3 Results

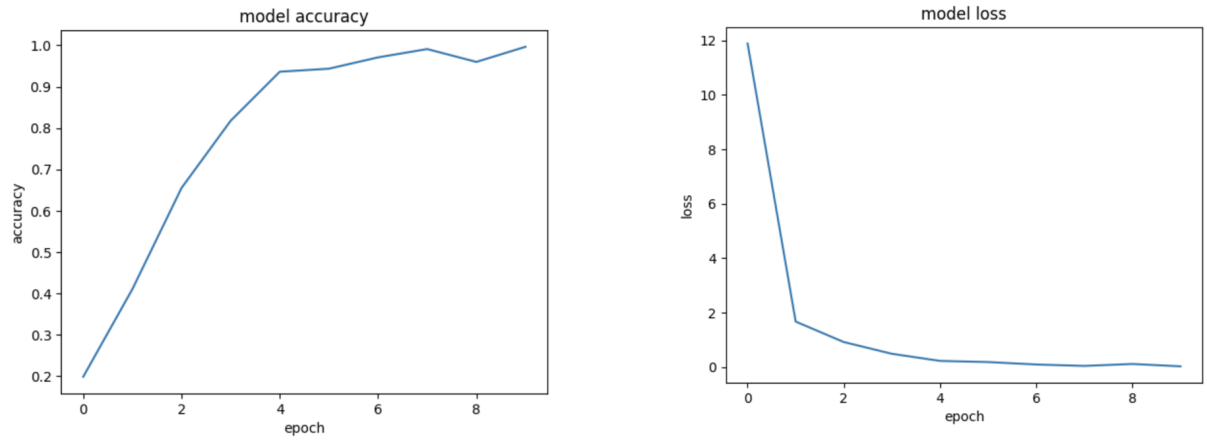

Figure 6: VGG16 Trained Model Accuracy and Losses

```
2/2 [==============================] - 0s 110ms/step
[4 1 0 1 0 3 2 0 1 3 0 0 6 5 2 1 5 6 1 0 5 2 0 0 1 1 5 6 3 0 6 2 0 3 0 1 5
 4 6 3]
[4 1 0 1 0 3 2 0 1 3 0 0 6 5 2 1 5 6 1 0 5 1 0 0 1 1 5 6 3 0 6 2 0 3 0 1 5
 4 6 3]
0.975
```

Figure 7: VGG16 Performance on Test Set

We were able to successfully train and then test the VGG16 model to achieve a 97.5% accuracy, which attains the model accuracy that we were looking for in our high level requirements. The figure above shows the accuracy and the loss after each epoch (which was chosen to be 10). The model accuracy and loss increase and decrease exponentially, respectively. We ended up integrating the model with the backend of the web application and we were able to take in an image input, run the model to perform image classification, and then update the user interface to reflect the plastic type. With more time, we would have liked to be able to take in live camera input either from a webcam or the camera sensor and in doing so we would be able to expand on a plastics dataset to use to continuously fine tune the model.

3. Cost and Schedule

3.1 Cost Analysis

3.1.1 Cost of Materials

This table does not factor in the cost of wires and generic connectors off of the PCB. The cost of passive components (capacitors, resistors, and inductor) was simplified to $0.10 per component as that is close to the average cost per item.

Description	**Manufacturer**	**Part Number**	**Quantity**	**Cost ($)**
Microcontroller	STMicroelectronics	STM32F103C8T6	1	6.42
Camera	Arducam	OV2640	1	25.99
Stepper Driver	Allegro	A4988	1	3.05
Stepper Motor	Adafruit	324	1	14.00
Trash Can	Sterilite	1.5 Gallon Trash Can	2	1.96
30W Wall Converter	Qualtek	QFWB-30-12-US01	1	9.93
LDO	Diodes Inc.	AZ1117CH-3.3TRG1	1	0.45
Buck Converter	Vishay Siliconix	SIC402ACD-T1-GE3	1	1.99
Passive RLC components	Various	Various	35	3.50
Microcomputer	Raspberry Pi	Raspberry Pi 2	1	43.59
Proximity Sensor	DFRobot	SEN0381	1	12.90
		Total		$123.78

Table 1: Materials Cost Analysis

3.1.2 Cost of Labor

Since the majority of this group is in Computer Engineering, the hourly rate listed below is based on the average salary of a Computer Engineering graduate, which is $105,352 [13]. This is approximately $52.68 an hour based on a 40 hour work week for 50 weeks a year. In addition to the work of our team, we also used the machine shop for the physical design and that work took approximately one day.

Name	Hourly Rate ($)	Hours	Total ($)	Total x 2.5 ($)

Jennifer Chen	52.68	150	7902	19755
Smruthi Srinvasan	52.68	150	7902	19755
Jason Wright	52.68	150	7902	19755
			Total	$59,265

Table 2: Labor Cost Analysis

3.2 Schedule

- Week 6 (9/25-9/29):
    - Design document: Everyone
    - Schematic development: Jason
    - Machine learning model selection: Jennifer and Smruthi
- Week 7 (10/2-10/6):
    - Design review: Everyone
    - PCB review and necessary modifications: Jason
    - Order parts and materials: Everyone
    - Conduct additional research on VGG: Jennifer and Smruthi
    - Final discussion with machine shop on mechanical design: Everyone
- Week 8 (10/9-10/13):
    - Order PCB: Everyone
    - Train VGG model: Smruthi and Jennifer
- Week 9 (10/16-10/20):
    - Program microcontroller: Jason
    - Fine tune model on Raspberry Pi: Smruthi
    - Begin user interface development: Jennifer
- Week 10 (10/23-10/27):
    - Final PCB revisions: Jason
    - Order PCB (if necessary): Everyone
    - Finish user interface: Jennifer
    - Integrate microcontroller with Raspberry Pi: Smruthi
- Week 11 (10/30-11/3):
    - Finalize assembly: Jason
    - Complete integration and begin testing: Jennifer and Smruthi
- Week 12 (11/6-11/10):
    - Continue testing: Everyone
- Week 13 (11/13-11/17):
    - Mock demo: Everyone
- Week 14 (11/20-11/24):
    - Final modifications: Everyone

- Week 15 (11/27-12/1):
    - Final demo: Everyone
- Week 16 (12/4-12/8):
    - Final presentation: Everyone

4. Conclusion

4.1 Accomplishments

While we were unable to get the entire system working, we were able to get a majority of the individual components functional. The final functionality can be separated into two parts, the software component with the user interface and the machine learning model and the hardware component with the microcontroller, stepper motor, and proximity sensor. On the software side, we were able to successfully classify plastic container symbols using the machine learning model and communicate that classification with the user interface to display information regarding the plastic in addition to nearby recycling centers. In the hardware system, we were able to program the microcontroller to use the motor to tilt the platform when an item is placed in front of the proximity sensor. This was not exactly the intended function of the hardware portion because we were not able to implement an integral connection between the hardware and software, but we got very close that with a few modifications we can achieve full functionality. Despite this, we were able to successfully complete portions of our system and we are confident that with a little extra time we would have been able to connect the software and hardware through the model and microcontroller to accomplish full functionality.

4.2 Uncertainties

A major uncertainty within our project was the Raspberry Pi not working. We were unable to boot the OS downloaded from the Raspberry Pi website and at one point the green LED indicator on the board would not even flash. This resulted in the loss of an integral component. While we were able to work around loading the model onto the Raspberry Pi, we had designed our system with the intention of the Raspberry Pi being the connection between the microcontroller and user interface. The failure of the board resulted in us not being able to connect the software to the hardware. We were able to make a determination of recyclability and simulate the physical sorting mechanism, but were unable to test the physical sorting mechanism based on recyclability.

Another uncertainty is the training set for the machine learning model. Since it can be difficult to obtain images of the less common plastics, it reduced the size of the dataset for us to train and test on. Ideally, we would have had thousands of images to train on which would have made our model more robust and capable of accurately identifying containers regardless of placement on the platform. Due to the fact that we had a more limited dataset, there was less room for user error when it came to placing the container on the platform. With a more robust model, potential user error can be handled. Moving forward, we will place an emphasis on selecting a larger dataset, if available.

4.3 Ethics and Safety

Throughout the project development process, we followed the IEEE code of ethics, and made sure to learn and apply new skills through design and implementation and follow all safety guidelines [15]. Per IEEE Code of Ethics, we made sure to protect any user data and consider data privacy pertaining to data collected such as the user's location. We did not store any sensitive information pertaining to the user's location in our web application. One potential safety concern would be that the plastic container should not have any liquids and should be empty. Since the plastic container will be placed on the mounted system in between the two bins, any potential leakage could lead to electrical safety issues. Another potential concern would be the weight of the container. If the container is too heavy or causes the platform to be unbalanced, it could cause the container to fall or damage part of the system.

4.4 Future Work

Given the opportunity to continue working on this project, there are several changes and extensions we would like to make. The main recommendation for future work would be to fully integrate all of the subsystems. Due to the faulty Raspberry Pi, we were missing the connection between the microcontroller and machine learning model. Since we were able to house the machine learning model with the user interface, the next step would be to make the connection between the model and microcontroller. This would likely need some modifications to the PCB design, particularly the selection of a different microcontroller that has WiFi capabilities. We would also like to extend the user interface functionality from our web application to a mobile application in order to increase the ease of use for users. This additional functionality could look like compatibility with the Google Maps app and the ability to upload a picture from the phone camera. Lastly, we can also integrate this with other recyclable detecting systems, such as metal and glass detection.

References

[1] EPA. “The U.S. Recycling System” (2022), [Online]. Available: https://www.epa.gov/circulareconomy/us-recycling-system#:~:text=For%20the%20environment%2C%20recycling%3A,and%20process%20new%20raw%20materials. (visited on 09/12/2023).

[2] A. Bradford, A. Truelove, S. Broude. “Trash in America: Moving From Destructive Consumption to a Zero-Waste System” (2018), [Online]. Available: https://frontiergroup.org/resources/trash-america/#:~:text=Currently%2C%20though%2C%20the%20majority%20(,much%20material%20at%2034.6%20percent. (visited on 09/09/2023)

[3] *“SiC402A/B”,* SiC402A/B, Vishay Siliconix, Jun. 2020. [Online]. Available: https://www.vishay.com/docs/63729/sic402abcd.pdf (visited on 09/21/2023).

[4] *“Reference Board User’s Manual for SiC403 (6 A), SiC402 (10 A), and SiC401 (15 A) Synchronous Buck Regulators”,* SiC402A/B, Vishay Siliconix, Nov. 2014. [Online]. Available: https://www.vishay.com/docs/62923/sic401.pdf (visited on 09/21/2023).

[5] *“STM32F103”,* STM32F103, STMicroelectronics, Sep. 2023. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f103cb.pdf (visited on 09/21/2023).

[6] *“AZ1117C”,* AZ1117C, Diodes, Sep. 2022. [Online]. Available: https://www.diodes.com/assets/Datasheets/AZ1117C.pdf (visited on 09/21/2023).

[7] T. Hardin. “Plastic: It’s Not All the Same” (2021), [Online]. Available: https://plasticoceans.org/7-types-of-plastic/ (visited on 11/21/2023).

[8] Geoapify. ““Places API Playground”.” (2023), [Online] Available: https://apidocs.geoapify.com/playground/places/ (visited on 11/15/2023).

[9] Google Maps Platform. ““The Maps Embed API Overview”.” (2023), [Online]. Available: https://developers.google.com/maps/documentation/embed/get-started (visited on 09/14/2023).

[10] Boesch, Gaudenz. ““VGG Very Deep Convolutional Neural Networks (VGGNet) - What you need to know”.” (2023), [Online]. Available: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/ (visited on 09/21/2023)

[11] PyTorch. ““VGG16_BN”.” (2017), [Online]. Available: https://pytorch.org/vision/main/models/generated/torchvision.models.vgg16_bn.html#torchvision.models.vgg16_bn (visited on 09/21/2023).

[12] Kaggle. ““Plastic Identification Symbol”.” (2020) [Online] Available: https://www.kaggle.com/code/abhichoudhury/plastic-identification-symbol/input (visited on 09/21/2023).

[13] T. G. C. of Engineering. ““The Grainger College of Engineering - Computer Engineering”.” (2023), [Online]. Available:

https://grainger.illinois.edu/academics/undergraduate/majors-and-minors/computer-engineering (visited on 09/21/2023).

[14] Datagen. “Understanding VGG16: Concepts, Architecture, and Performance.” (2023), [Online]. Available: datagen.tech/guides/computer-vision/vgg16/#:~:text=The%20VGG16%20model%20can%20achieve,smaller%203%C3%973%20filters (visited on 12/4/2023).

[15] IEEE. “”IEEE Code of Ethics”.” (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 09/13/2023).

Appendix A: PCB Schematics

A.1 Microcontroller Schematic

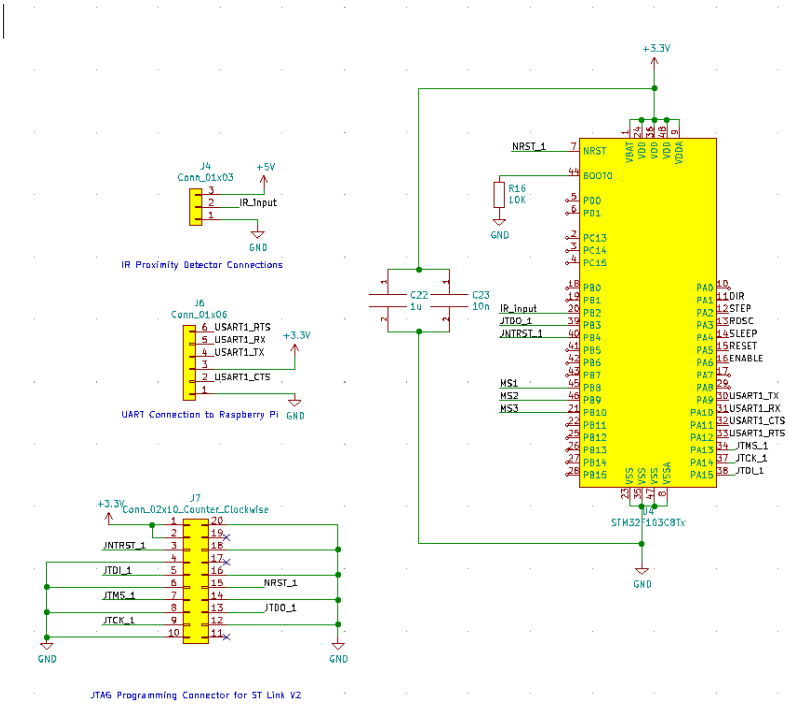

Figure 8: Microcontroller connections

A.2 Stepper Motor Driver Schematic

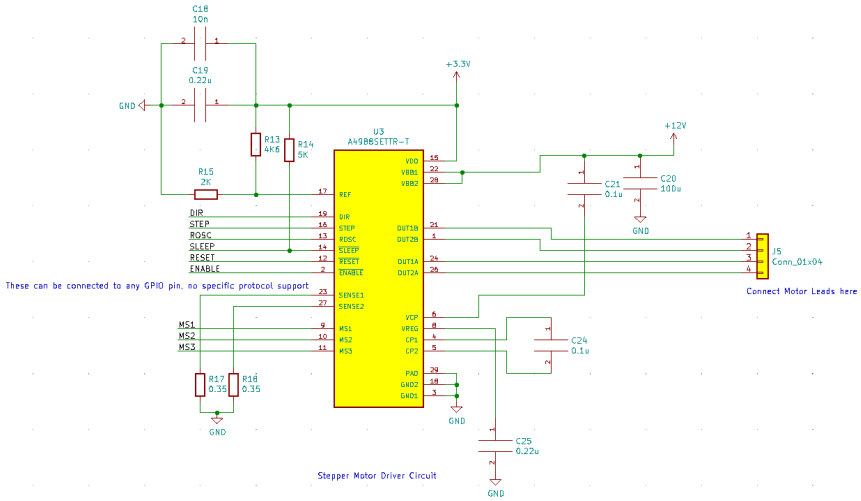

Figure 9: Stepper Motor Driver

A.3 Buck Converter and LDO Schematic

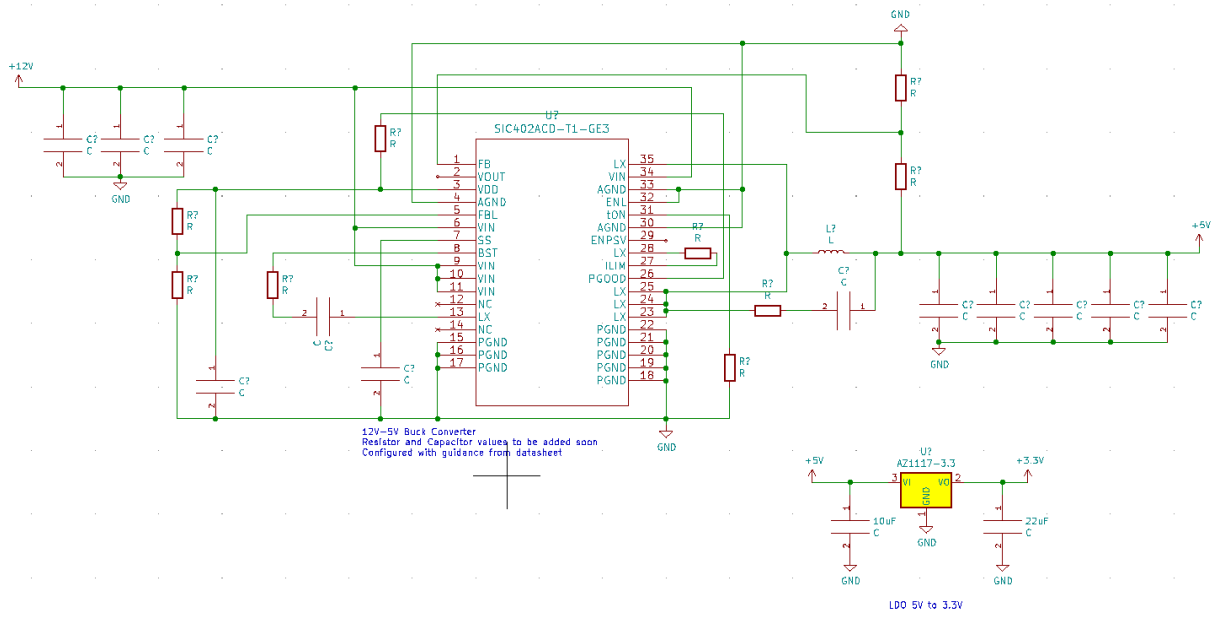

Figure 10: Buck Converter and LDO Schematic

Appendix B: Requirements and Verification Tables

B.1 Control Subsystem Requirements and Verification Table

Requirement	Verification
• Stepper motor capable of rotating the amount required to move the item and return to level position ± 1.8° (1 step)	• Test sorting motion on objects of different weights and shapes to find necessary angle difference (estimated 45°) • Check angle of return position and confirm new items will be able to stand
• MCU can communicate with multiple other components and collect data for analysis in an organized way with use of one of several supported protocols	• MCU receives data input from camera and GPS modules with minimal losses (>98% success rate) • MCU communicates back and forth with microcomputer, can be verified through raspberry pi output • Display the data inputs to the U/I to verify
• Microcomputer interfaces with the internet and retrieves data in a reasonable time (<5 seconds)	• Run a script that times the data retrieval and outputs the time • Display script output on the U/I and check the time is <5 seconds and the information retrieved is accurate

Table 3: Control Subsystem RV Table

B.2 Power Subsystem Requirements and Verification Table

Requirement	Verification
• Ripple on the 5V system stays under 250 mV, as needed by the raspberry pi and can support expected current draw by the LDO and raspberry pi (1.5 A)	• Ripple can be measured with an oscilloscope, current support is available on datasheets

• Ripple on the 3.3V system is kept under 300 mV for continuous MCU operation, current must support all 3.3V components	• Ripple can be measured with an oscilloscope, current support is available on datasheets

Table 4: Power Subsystem RV Table

B.3 Sensor Subsystem Requirements and Verification Table

Requirements	Verification
• The image the camera produces must be at least 244 x 244 to ensure that the symbol can be read properly by the VGG model	• Place a plastic container in front of the camera sensor • Verify that the pixel count is at least 244 x 244 pixels using OpenCV
• The GPS must provide coordinates within a 10 meter radius given that the device is in a location free of large obstacles	• Place device complete with the GPS sensor in a location with known coordinates • Display the GPS coordinates sent by the device on the web page (for verification purposes only) • Calculate the distance between the true location and GPS coordinates • Verify the calculated distance is within a 10 meter radius

Table 5: Sensor Subsystem RV Table

B.4 User Interface Subsystem Requirements and Verification Table

Requirements	Verification
• The web application must display information of the type of plastic the user is attempting to recycle	• Navigate to the web page • View the left hand side for plastic type and the corresponding information • Verify the plastic type displayed matches the type printed on the container

• The web application must be able to display recycling locations within a 30 mile radius, if applicable to a user's location	• Navigate to the web page • View the right hand side for the map visualization • Verify that the locations (if any) displayed accept the plastic type

Table 6: User Interface Subsystem RV Table