SimpleSort
Design Documentation

Justyn Duthler
Kevin Ryan
Shirley Phuong
Trevor Carleton
Kevin Spence
Brevan Chun

I. Executive Summary

Improperly sorted trash is a large problem that adversely affects the health of the environment. According to the United States Environmental Protection Agency only 32% of Americans recycle [4]. To combat this problem, Team 7 has designed a product to increase the rate at which trash is correctly sorted in a consumer environment. SimpleSort is an at-home garbage can that automatically sorts garbage into recycling, compost, or trash compartments. Sorting is accomplished using image classification and a rotating mechanism that drops a trash item into the correct compartment. The design allows consumers to recycle and compost at home without having to know how to manually sort their waste, or taking the time to do so.

SimpleSort offers consumers:

- Automatic garbage sorting for single items at a time
- Over 80% accuracy when sorting items
- Low power consumption
- Sales cost of $180
- Improved composting and recycling rates
- An effortless way of improving environmental health from the comfort of your home

II. Ethics Statement

1) Safety
   - No harm will come to the consumer through normal use of product.
   - Servo motors automatically stop when facing minor resistance.
   - Design of product falls within electrical safety guidelines.
2) Privacy
   - SimpleSort does not require an internet connection.
   - The camera will remain off until an item is placed into the trash can.
   - No data from trash can will be collected.
3) Environmental Impact
   - To prevent SimpleSort from contributing to landfill, any malfunctioning SimpleSort can will be available for free repair within the first three years of purchase.
4) Quality
   - We guarantee that the products received are of high quality and not defective.
   - Products will go through rigorous testing before being made available to the public to ensure quality and safety.

CONTENTS

III. Problem Definition

A. Need & Goal Statements

Need: There are too many recyclable and compostable items being placed in the trash.

Goal: Increase recycling and composting rates, while decreasing the rate of garbage improperly sorted.

B. Design Objectives

Our primary design objective is to create a consumer and environmentally friendly product that is able to place at least 80% of all recyclable, compostable, and landfill items in their corresponding sections of the trash can. For our current prototype, we are aiming to design a product that is made for at-home use. Since this is designed for individual households, we would like to have inexpensive manufacturing costs to provide an affordable and marketable trash can, as well as a power consumption of less than 50W.

Design Objective	**Units**	**Target/Range**
Sorting Accuracy	Percentage	>80%
Manufacturing Costs	Dollars	$80
Sale Costs	Dollars	$180
Power Consumption	Watts	<50W

Fig. 1: Design Objectives

IV. Background

A. Personas

Figures 2, 3, and 4 show the different consumer archetypes that we will interact with during the process of creating and selling our product. We predicted some of their characteristics to better understand how to design our project. Charles Davenforth represents the type of user who would be interested in the product because he regularly recycles, but would like to have an automated system to save him time. Chuck Bigguns represents the type of user who would be interested in the product because he does not regularly recycle, but may be forced to recycle because of his apartment complex or would like to recycle but does not have time to. Simplehuman is a company that specializes in designing at-home kitchen and bathroom products that we referenced when designing our prototype, and David Harrison is our professor who helped oversee our project.

Name	Role	Context	Goals
Charles Davenforth	- Trash producer - Busy homeowner - Environmentally conscious person	- Home use - Provide effortless trash sorting	- Increase amount of trash properly sorted with minimal effort - Contribute less to landfills

Name	Role	Context	Goals
Chuck Bigguns	- Trash producer - Busy homeowner - Does not recycle - Apartment complex forces him to sort trash	- Home use - Will sort trash for him because he will not manually	- Automatically sort trash for someone who would not regularly - Contribute less to landfills

Fig. 2: End Users

Name	Role	Context	Goals
Simplehuman https://www.simplehuman.com/	- Expert in home automation and home appliances	- 21 year old company - Use as reference for creating automated home appliances	“To Design tools that help people become more efficient at home” -simplehuman

Fig. 3: Experts/Consultants

Name	Role	Context	Goals
David Harrison	- CSE 123 Professor at UCSC	- Tasked us to complete project - Numerous years of experience with completing projects	- Task students to complete an engineering design project in teams of 6 using the engineering design life-cycle

Fig. 4: Client

B. Research into Existing Products

When researching existing products that partially meet our need statement, the first types of products we looked for were products we had previously seen. We quickly found garbage sorting systems where the trash, recycling, and compost bins are lined up side-by-side such as the Three Stream Recycling Containers offered on recyleaway [1], or trashcansWarehouse [2]. These types of larger trash cans are commonly seen on college campuses or various types of recreational facilities, but their market prices are extremely high at over $500.

There are also companies offering smaller at-home sorting systems with three components that are generally made out of stainless steel and range from around $100–$200. Neither of the previous two systems offer any type of automatic sorting, but we found that Bin-e [6], and CleanRobotics [15] were two companies producing self sorting trash bins. However Bin-e's product was not order-able as of June 2021, and no sales information could be found regarding CleanRobotic's Trashbot. We also found that Bin-e could only sort recycling and trash, and handle one item at a time. Likewise, Trashbot could only handle one item at a time but was able to sort recycling, trash, and compost. Looking into the existing products validated our reasoning for attempting to design this product. The biggest drawback of our product is that we can only sort one item at a time, but the two most comparable products also can not sort more than one item at a time. The price points of both the self sorting and manual sorting trash cans also gave us reason to design our product since our estimated price point is significantly lower.

V. Detail of Design

A. Architecture

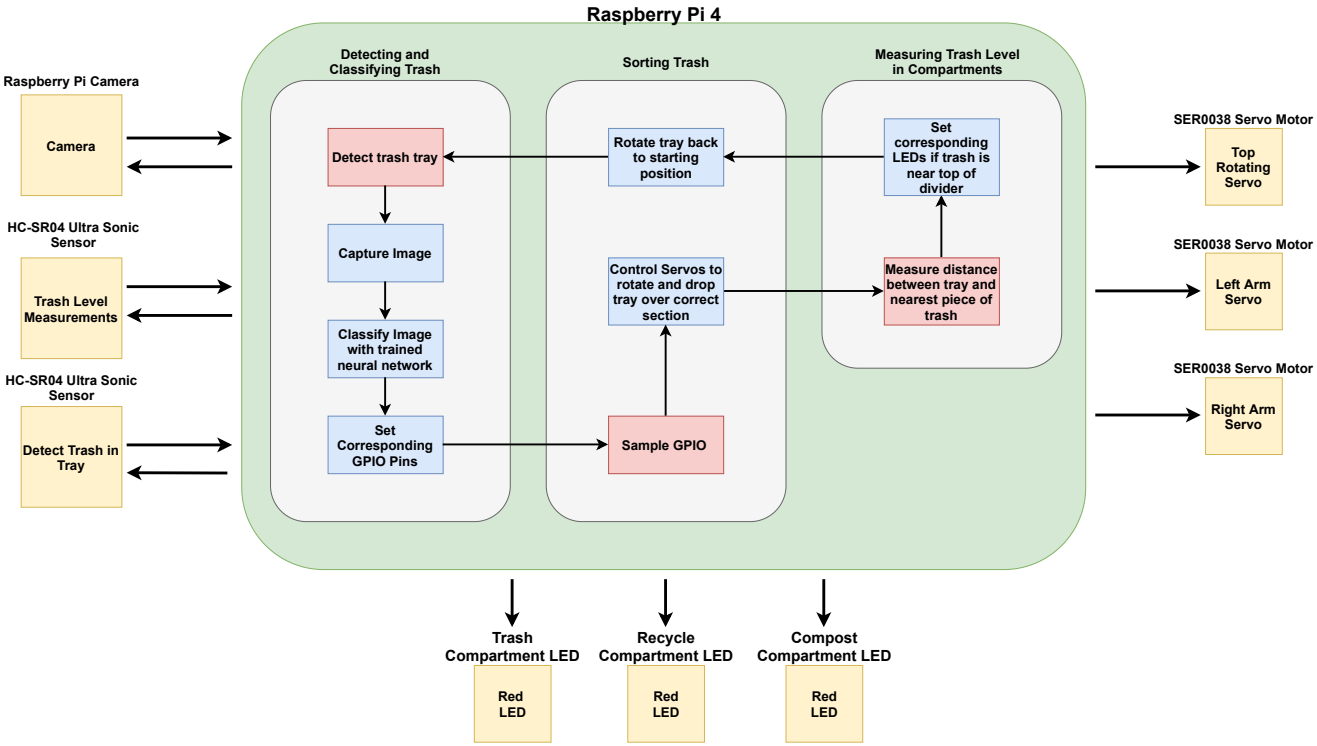


Fig. 5: System Architecture

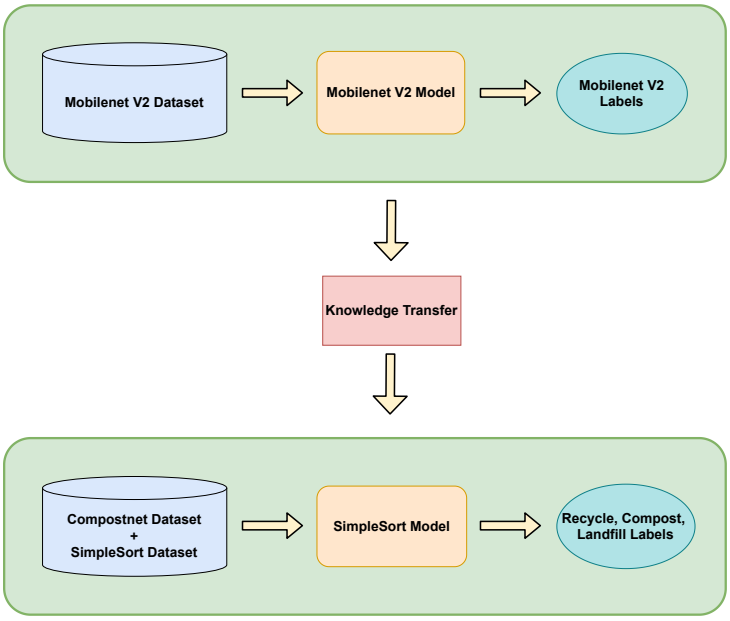


Fig. 6: Image Classification Transfer Learning Architecture

The architecture of the project can be seen in Figure 5. As seen in the yellow external boxes, the physical hardware consists of two ultrasonic sensors, three LEDs, three servo motors, and a camera. The first ultrasonic sensor is responsible for measuring the amount of trash in each compartment. The next

ultrasonic sensor is used to detect when an item has been placed into the sorting tray and will indicate to the software to begin the classification and sorting process. The three LEDs on the bottom of the diagram are lit up red when the trash level for the corresponding compartment is measured to be full. The three servos are responsible for rotating the tray over the correct compartment and then dropping the tray to release the item. The camera is used to capture images of the waste items placed in the tray, which are then passed to the image classification model. The green section represents the state machine running on the microprocessor for this project, the Raspberry Pi 4. This state machine implements the loop for classifying and sorting trash. The process follows the pattern of waiting for an item to be placed in the tray and taking an image and classifying the item when it is inside the tray. Next, setting GPIO pins for communicating to the servo code where to drop the item, controlling the servos to rotate the tray and drop the item, and measuring the amount of trash in the compartment the item is placed into. Finally, the corresponding LED is set if the compartment is full, rotating the tray back to the initial position, and waiting once again for an item to be placed into the tray.

Figure 6 shows the architecture for our image classification model. The model uses transfer learning to achieve a higher classification accuracy. The base model used is Google's MobileNetV2 model [12] which is trained on over 1 million images and contains over 1,000 classes. For our data set, we used CompostNet [9] which contains 7 classes of images: cardboard, compost, glass, metal, paper, plastic, and trash. In addition to this, we added our own photos taken with the Raspberry Pi camera into the data set to gear it towards our own product. Since our data set is not very large, transfer learning greatly increases our accuracy compared to creating a new model from scratch.

B. GPIO Pinout

Raspberry Pi 4 Header			
Pin	Assignment	Pin	Assignment
3V3	none	5.0V	TS 6v, LAS 6v, RAS 6v
SDA	none	5.0V	TLDS 5v, TDDS 5v
SCL	none	GND	Level Converter GND, TLDS GND, TDDS GND
#4	#24	TXD	none
GND	TS GND, LAS GND, RAS GND	RXD	none
#17	TS PWM	#18	TDDS Trig
#27	LAS PWM	GND	none
#22	RAS PWM	#23	none
3.3V	Level Converter L	#24	#4
MOSI	none	GND	none
MISO	none	#25	#5
SCLK	none	CE0	none
GND	none	CE1	none
EED	none	EEC	none
#5	#25	GND	none
#6	none	#12	none
#13	LED Landfill	GND	none
#19	LED Recycle	#16	Level Converter L2
#26	LED Compost	#20	Level Converter L1
GND	Level Converter GND	#21	TLDS Trig
Level Converter			
Pin	Assignment	Pin	Assignment
L1	#20	H1	TLDS Echo
L2	#16	H2	TDDS Echo
L	3.3V	H1	5.0V
GND	GND	GND	GND
L3	none	H3	none
L4	none	H4	none

Fig. 7: Raspberry Pi 4 and Level Converter Pinout
TS = Top Servo
LAS = Left Arm Servo
RAS = Right Arm Servo
TLDS = Trash Level Distance Sensor
TDDS = Trash Detection Distance Sensor

Figure 7 details the pins on the Raspberry Pi 4 [10] and the voltage level converter [14] which are used to control the external hardware. The pins can also be seen in the following circuit schematics, Figure 8 through 11.

C. Circuit Schematics

Figure 8 details the complete circuit schematic of the design. Bringing everything together is the header from the Raspberry Pi 4 [10]. This header contains the necessary ground pins, 5V ports, 3.3V ports, and GPIO pins for communicating to the connected hardware. Three DFRobot SER0038 [13] servo motors are powered using one of the 5V ports. Each servo motor is controlled using a PWM signal generated from the corresponding GPIO pin. Three LEDs are connected in series with a 330

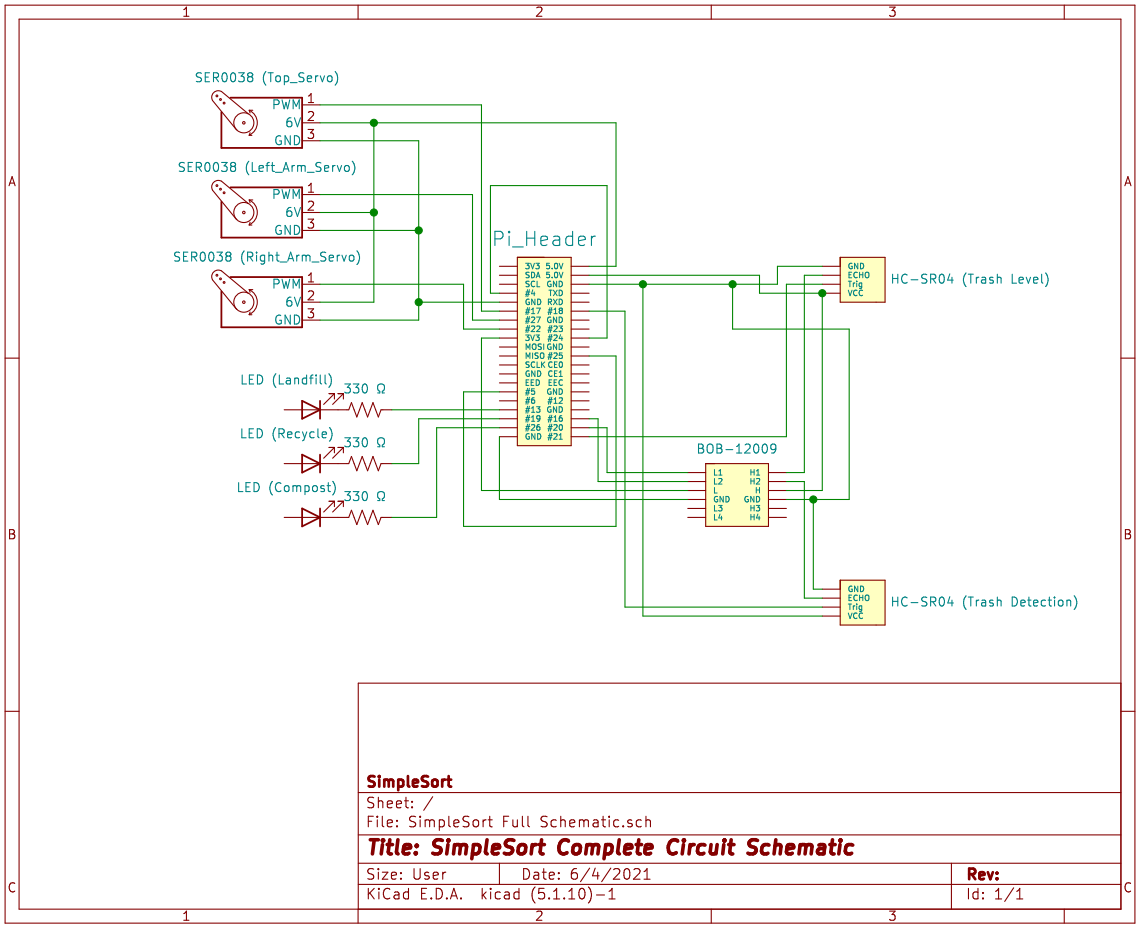


Fig. 8: Complete Circuit. This figure was made using KiCad [3]

ohm resister to GPIO ports for indicating the amount of trash in each compartment. Two HC-SR04 ultrasonic sensors [16], one for measuring the amount of trash in each compartment and one for detecting trash in the tray, are connected to the other 5V port. Then two GPIO pins are connected to the trigger input and the outputs are connected to the SparkFun BOB-12009 level converter [14] and the corresponding outputs are connected to GPIO pins. The level converter steps down the 5V echo pulse to 3.3V to avoid damaging the 3.3V GPIO pins on the Raspberry Pi. The following figures break down the complete circuit into its fundamental parts and goes into further detail of how the circuit is setup.

Figure 9 details the complete trash level detection circuit. This circuit is used to measure the amount of trash in each compartment and alert the user with LEDs to when a specific compartment is full. This is achieved with the SparkFun HC-SR04 ultrasonic sensor. This sensor sends a pulse when it receives a 10μs pulse on the trigger pin. When the pulse reflects off the nearest boundary and returns to the sensor, a high signal is set on the echo pin. The time between the echo pulse turning high and then turning low is the distance between the sensor and the nearest boundary. As stated previously, the HC-SR04 sensor requires 5V power and the echo and trigger pins thus use 5V. The Raspberry Pi uses 3.3V for its GPIO and using the ultrasonic sensor and the micro controller requires a voltage converter. This is achieved with the SparkFun BOB-12009 level converter. This chip requires power from the high voltage, 5V in this case, and power from the low voltage, 3.3V in this case. The trigger pin is directly connected to

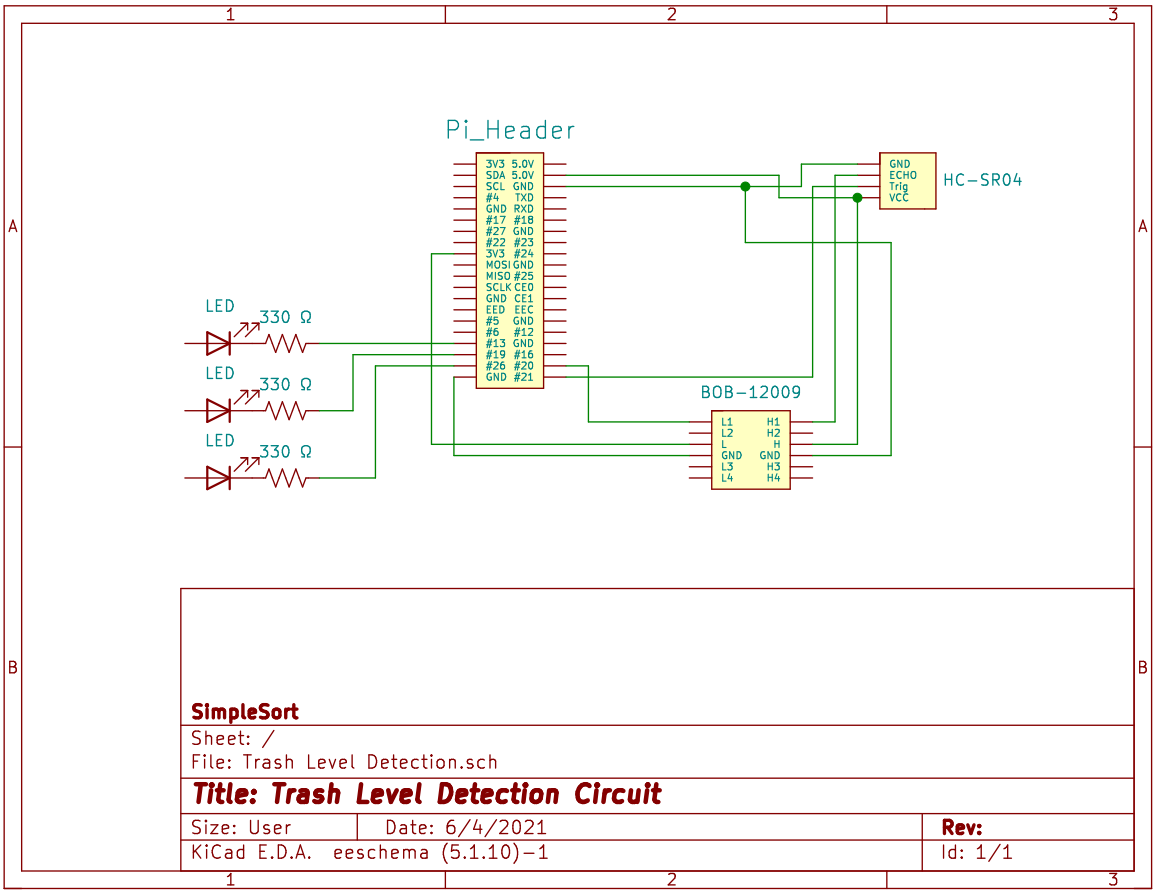


Fig. 9: Trash Level Detection Circuit. This figure was made using KiCad [3]

pin #21 on the Raspberry Pi because the 3.3V from the Pi GPIO is enough to trigger a pulse. The echo pulse is connected to the first pin on the level converter, H1, and its voltage shifted value is outputted on L1 and connected to pin #20 on the Pi. Three active low LEDs are connected to 330 ohm resisters and then to pins #13, #19, and #26 on the Pi. These LEDs will be turned on and pulled to GND when the distance calculated from the sensor for its corresponding compartment indicates the compartment is full.

Figure 10 details the trash tray detection circuit. This circuit is used to detect when an object is placed within the tray. If an object is placed between the sensor and the edge of the tray, the measured distance will be less than the distance from the sensor to the edge of the tray, indicating to the software that an item is in the tray. This circuit is almost identical to the trash level detection circuit, excluding the LEDs. The trigger pin on the HC-SR04 sensor is connected to pin #18 on the Pi. The echo pin on the sensor is connected to H2 on the level converter and the output, L2, is connected to pin #16 on the Pi.

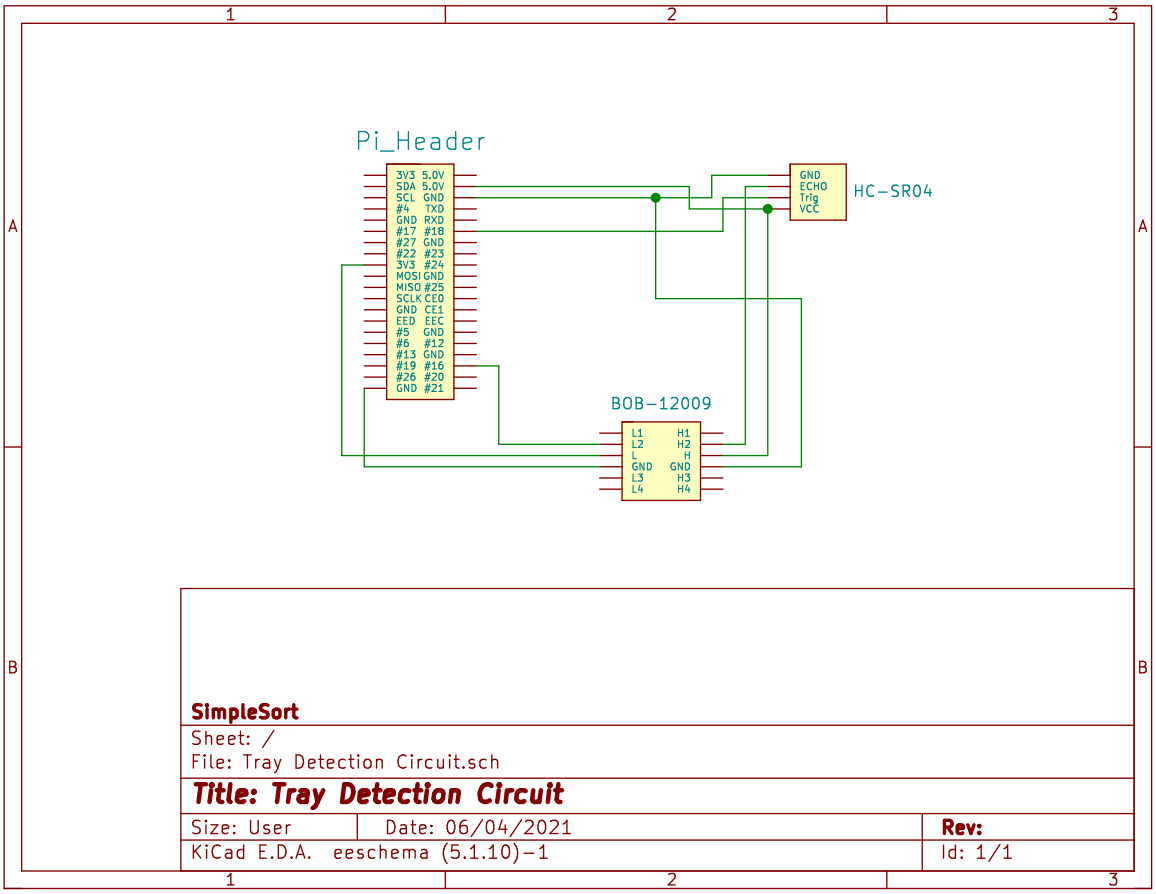


Fig. 10: Tray Detection Circuit. This figure was made using KiCad [3].

Figure 11 outlines the circuit used to connect the servo motors to the Raspberry Pi. Each of the servo motors has three connections: power, ground, and a PWM control signal. The servo motor inputs are connected to a female pin header, so wires are used to connect the servo to the breadboard holding the Pi header. The power of each of the servos is connected to the 5V supply of the Raspberry Pi, while the servo grounds are connected to the Raspberry Pi ground. The top servo PWM is connected to pin #17, with the left and right arm servos connected to pins #27 and #22 respectively.

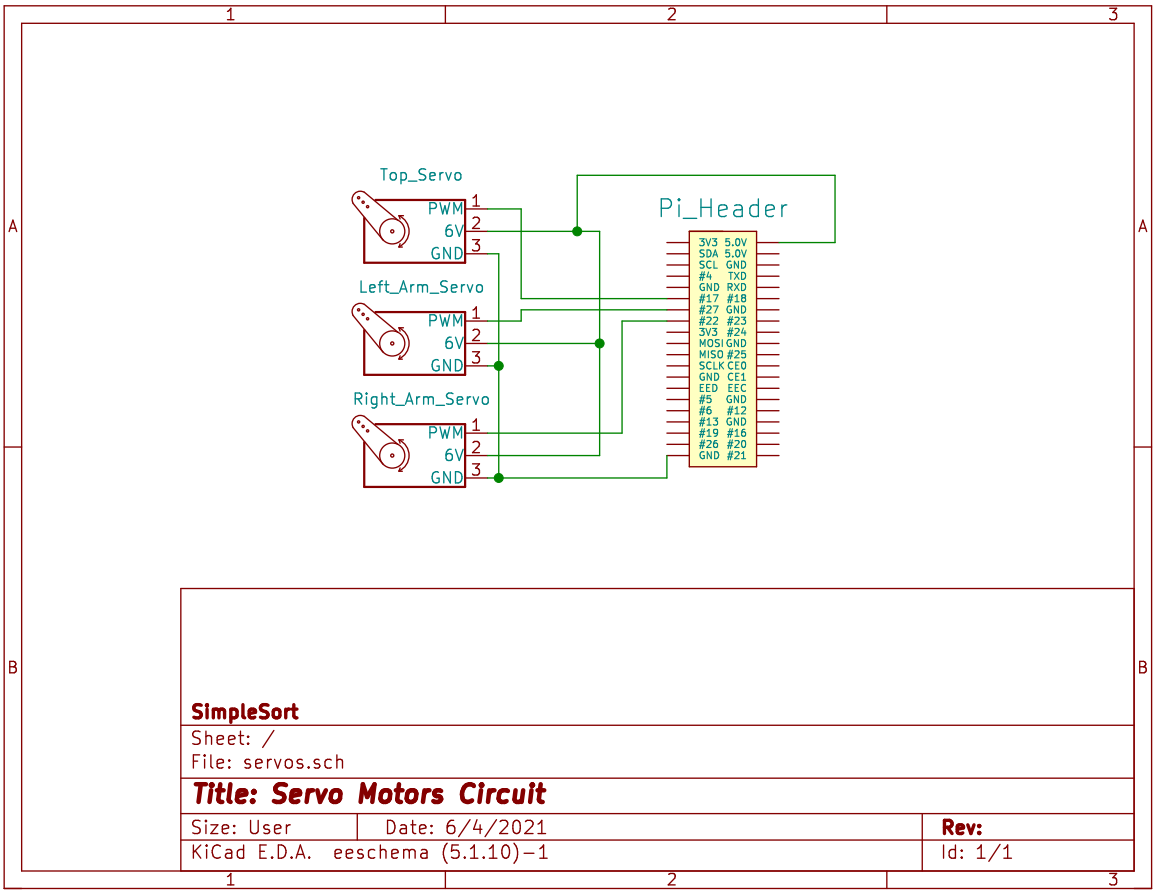


Fig. 11: Servo Motors Circuit. This figure was made using KiCad [3].

D. Software Flow Charts

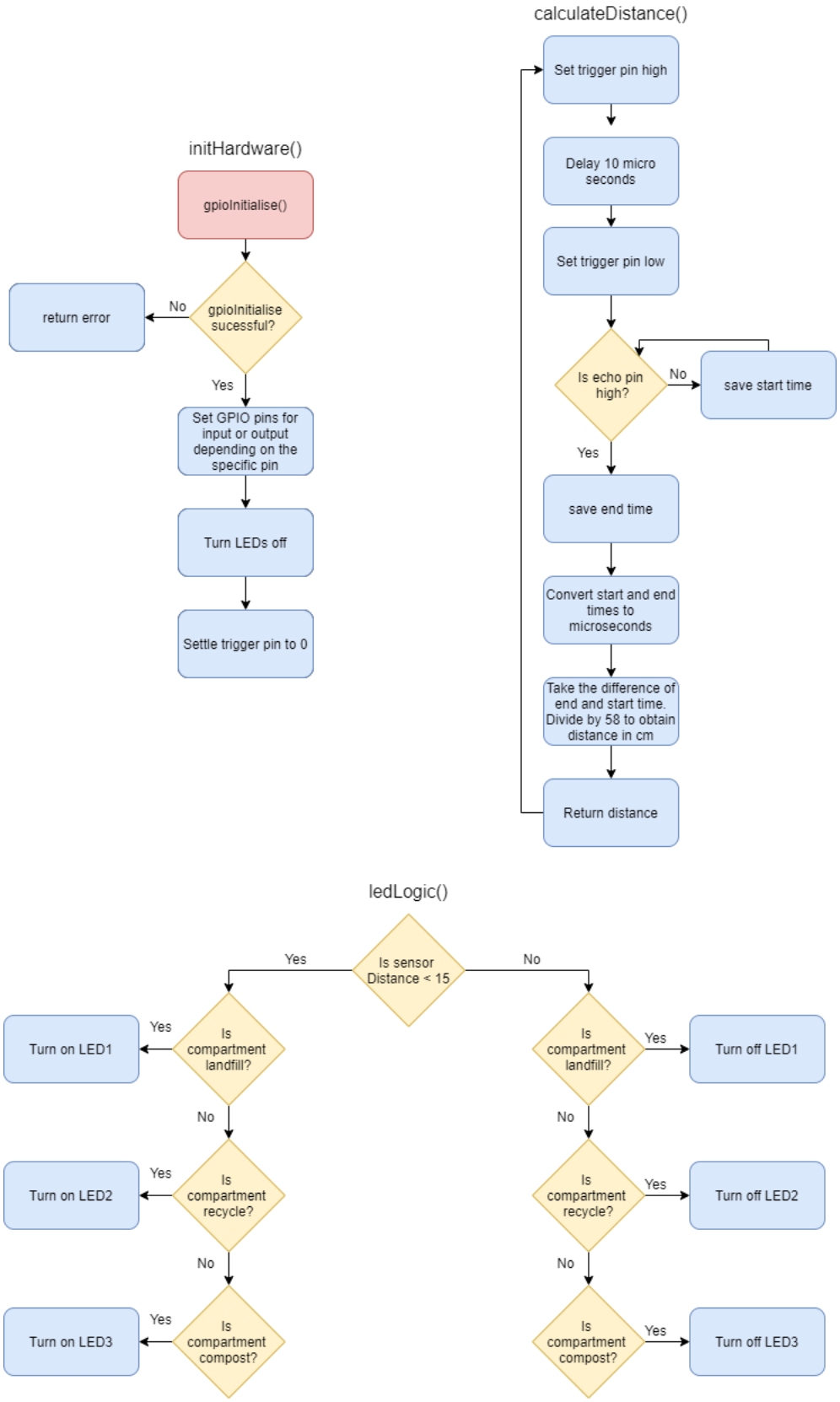


Fig. 12: Trash Level Detection Software Overview. Figure drawn using Diagrams [7].

Figure 12 shows the overall flow of the level detection software. The program first initializes GPIO pins to input and output in order to produce trigger and echo pins for the ultrasonic sensor. The initialize flowchart is shown on the upper left. Once the GPIO's are initialized, measurements of trash levels can be taken. The measurement flowchart is shown on the upper right. To take a measurement of the trash level, the trigger output pin goes high for $10\mu s$, saves the start time, and waits for the echo pin to go high, and saves the end time. Using the start and end times and the speed of the ultrasonic pulse, the program calculates the distance to an object. At the bottom the LED logic can be seen. If an object is detected within 15cm, the corresponding compartment's LED is set to on. If no object is detected within that distance the LED for that compartment is set to off. The level detection software takes a measurement of either the trash, compost, or recycle compartments each time one of those is selected by the image recognition software.

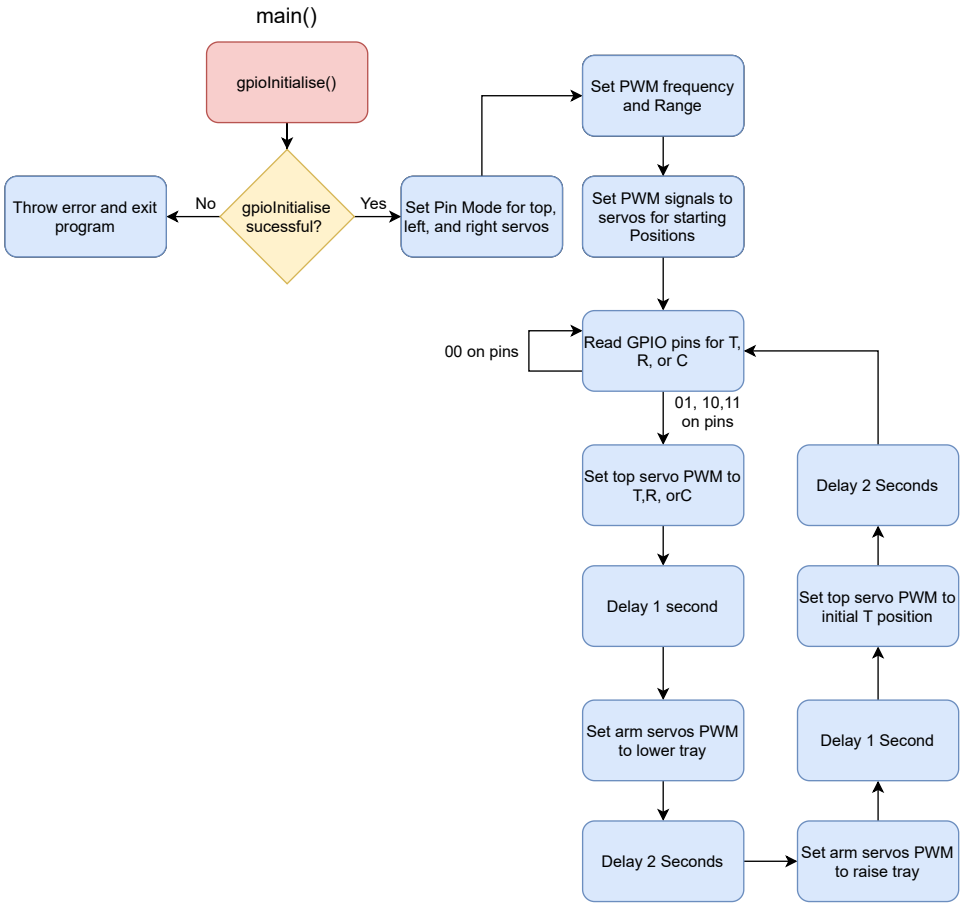


Fig. 13: Sorting Mechanism Control Software Overview. Figure drawn using Diagrams [7].

Figure 13 shows an overview of the code that controls the sorting mechanism. First, the GPIO pins are initialized and the PWM frequency is set. After the PWM signal is set to put the servos in the starting positions, the GPIO input pins read an input from the image recognition software. After receiving an input, the top servo PWM is set to either T, R or C, rotating the sorting mechanism above the corresponding portion of the bin. After a one second delay, the arm servos are set to lower the tray, then after a two second delay the arm servos are raised to the original position. The systems delays for a second after raising the tray, then the top servo PWM is reset to its original position and the system delays for two seconds before sampling the GPIO again for a new T, R, or C input.

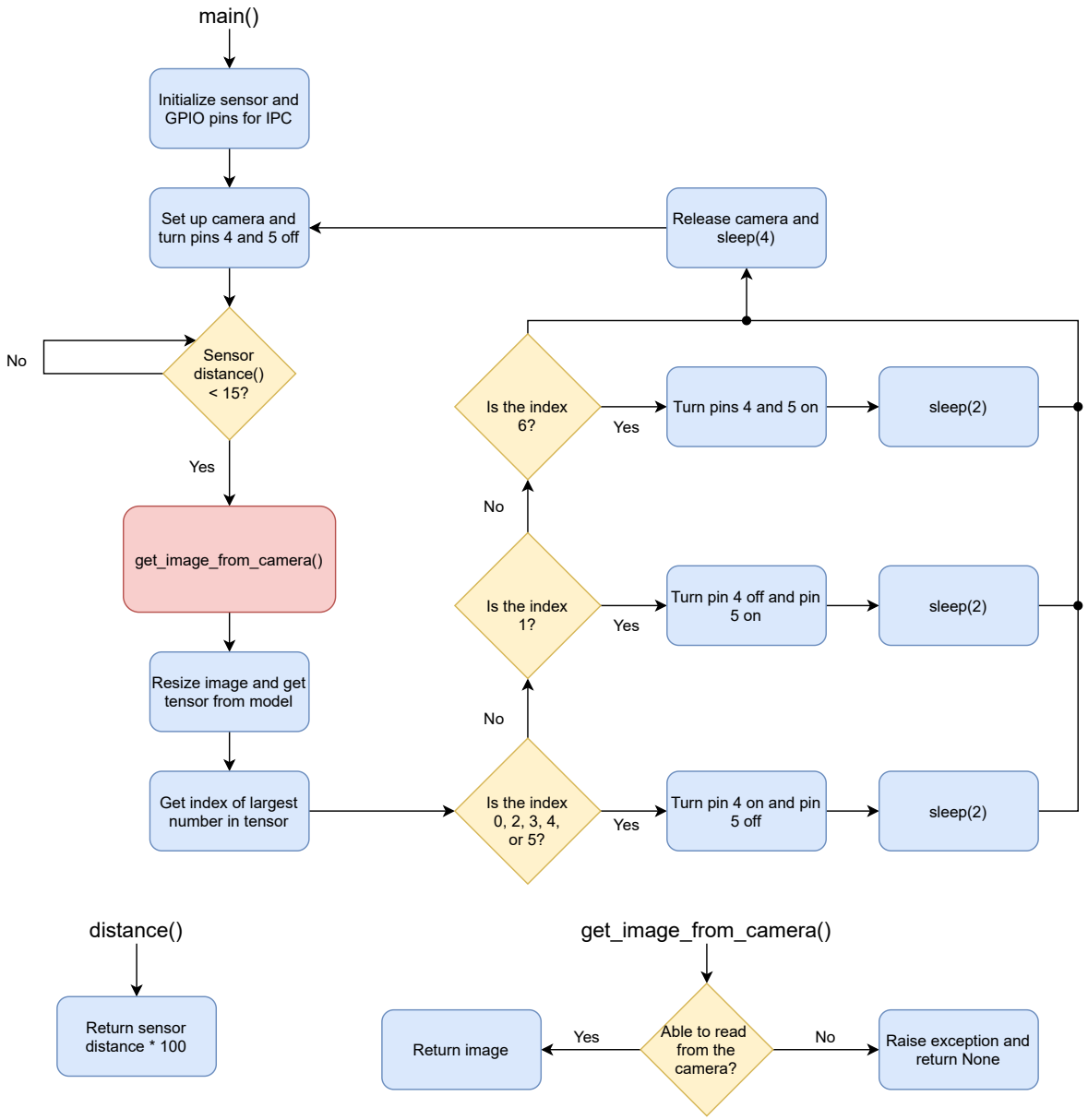
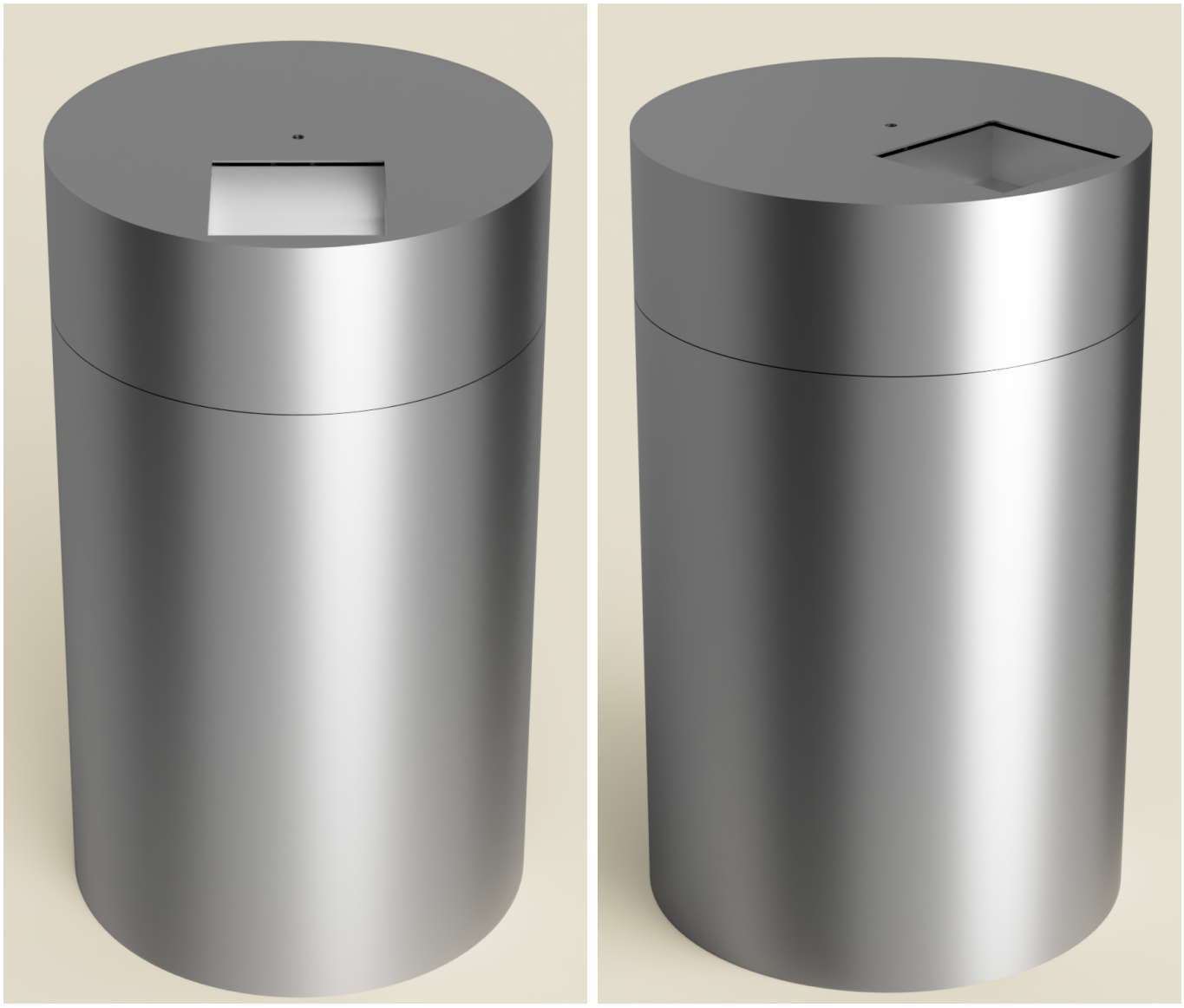


Fig. 14: Image Classification and Processing Software Overview. Figure drawn using Diagrams [7].

Figure 14 shows an overview of the code that controls the inputs and outputs of our image processing and classification. First, the GPIO pins and camera are initialized, and then an ultrasonic sensor is set to measure distances. If the distance measured becomes less than our threshold of 15cm (slightly less than the width of our sorting tray) then trash will have been detected. Once the item of trash is detected, the camera takes a picture of the tray and the image is appropriately formatted and ran through the image classification model. Given the output tensor from this model, the appropriate GPIO pins are set indicating trash, recycle, or compost so that the servo code can sample these. After a 2 second delay the program then sets the pins to off and returns to waiting for an item to be placed in the tray.

E. Aesthetic Prototype

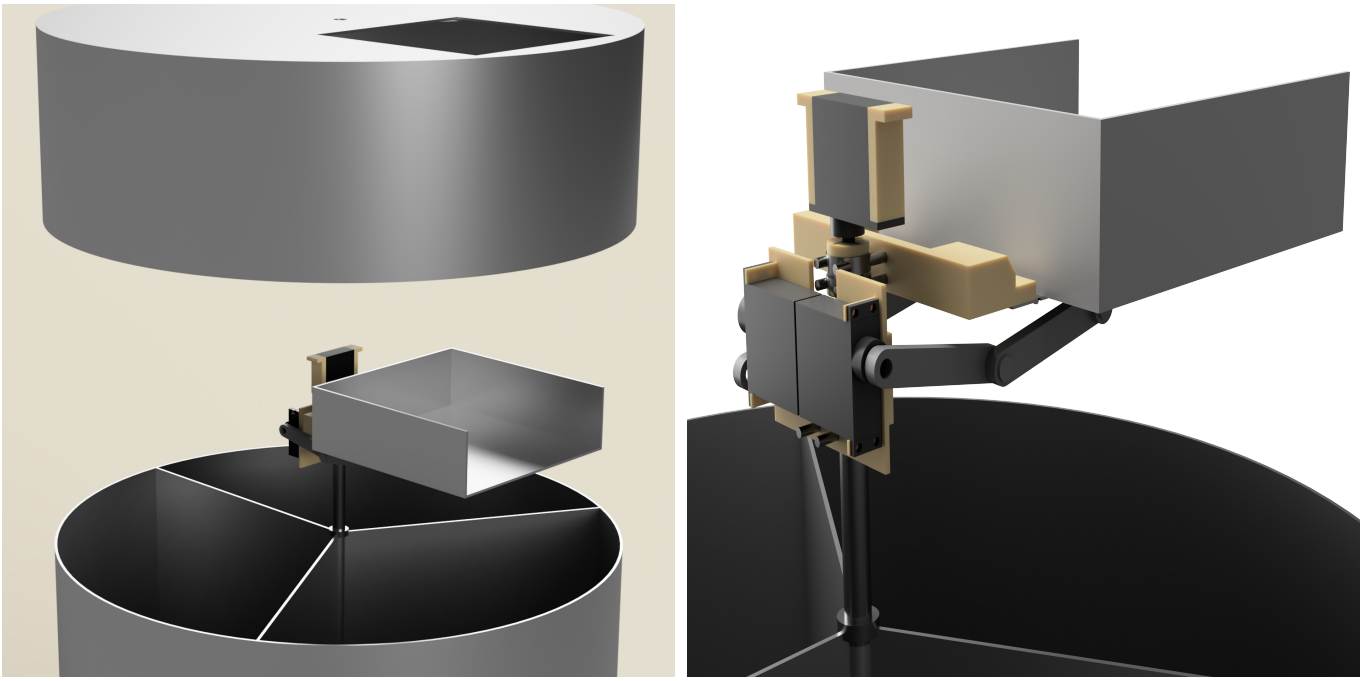


(a) (b)

Fig. 15: (a)(b) Renderings of the aesthetic prototype design created with AutoDesk's Fusion 360 [5].

The aesthetic prototype is how we envision a final product. The design shown in Figure 15 features a stainless steel exterior, which gives a simple and pleasing aesthetic fit for home usage. This design is also a somewhat compact size at 750mm in height and 95 liters or 25 gallons in total capacity. A smaller sized garbage can should be accommodating for a majority of households.

Looking closer at the internal components of the design, Figure 16 (a) shows the garbage can without the lid. Figure 16 (b) shows a more detailed view of the sorting mechanism itself. The sorting mechanism is composed of 3 servo motors and a drop tray. An item placed in the drop tray is classified using image recognition and the sorting mechanism will move the tray over the appropriate compartment and drop the item. This mechanism will be discussed in greater detail in the following section.



(a) (b)

Fig. 16: Renderings of the (a) lid assembly (b) sorting mechanism. Both figures were created with AutoDesk's Fusion 360 [5].

F. Design for Manufacture & Assembly

The overall design of the trash can (Figure 15) is intended to be minimalist so it fits well into most households. The design includes a stainless steel exterior, which provides both strength and style. Lifting off the lid reveals the sorting mechanism and the three compartments (Figure 17), one for each type of garbage: landfill, recycling, and compost. The trash can is 750mm tall with the lid, 600mm without the lid, and 450mm in diameter. The total capacity of the trash can is approximately 95 liters.

The sorting mechanism moves the garbage item into the appropriate section (trash, recycling, or compost). This mechanism is controlled by three servo motors: two servo motors control the drop tray and a third controls the rotation. The two servo motors controlling the drop tray are mounted to the central rod using a custom 3D printed bracket (Figure 26 (1)). The tray is also mounted to the central rod using a custom 3D printed bracket (Figure 26 (2)). 3D printed parts are necessary due to the unique mounting constraints of this design. Nearly everything is attached to the central rod. The drop tray is mounted some distance away from the central rod and the servo motors need to be oriented and rigidly attached to provide support for the drop tray. Off-the-shelf hardware could not be sourced for these parts. For production manufacturing, these components can be fabricated more efficiently using different methods and materials. Some off-the-shelf components include the central rod and u-bolt clamps, which are used to mount the custom brackets. All the electronic components are off-the-shelf.

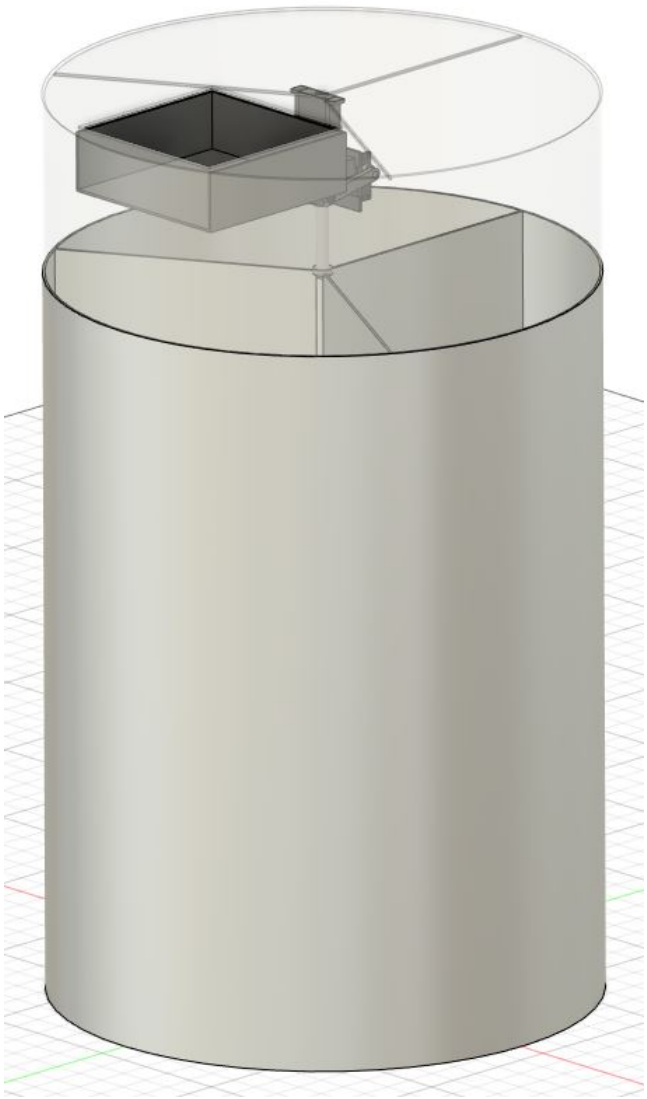
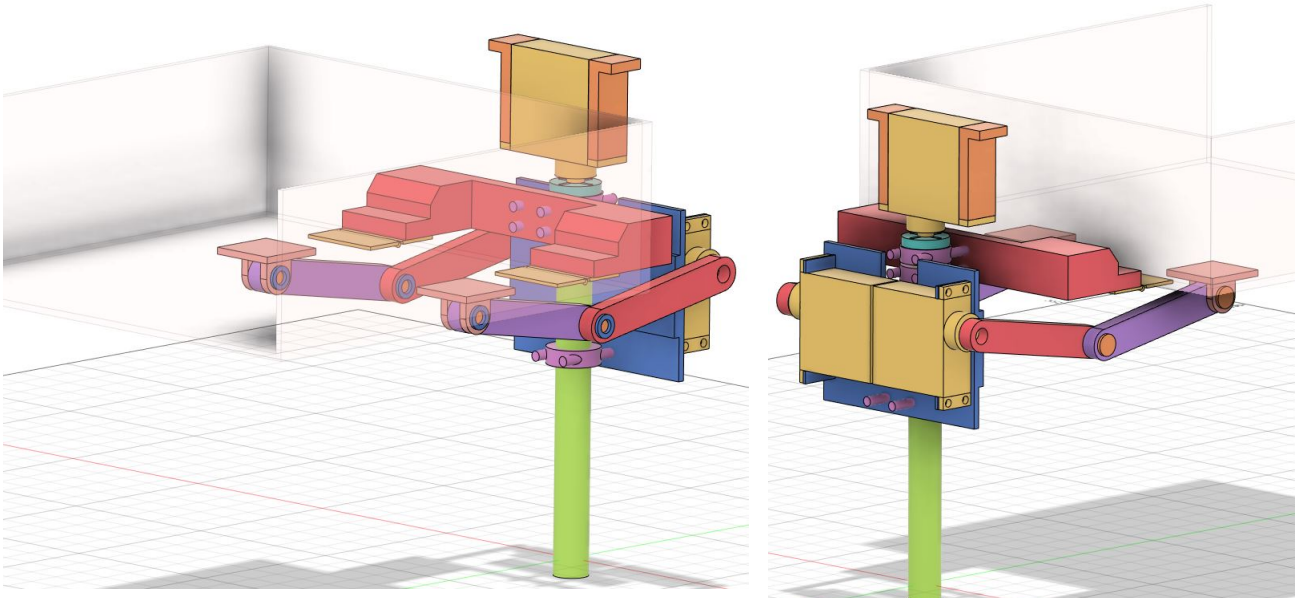


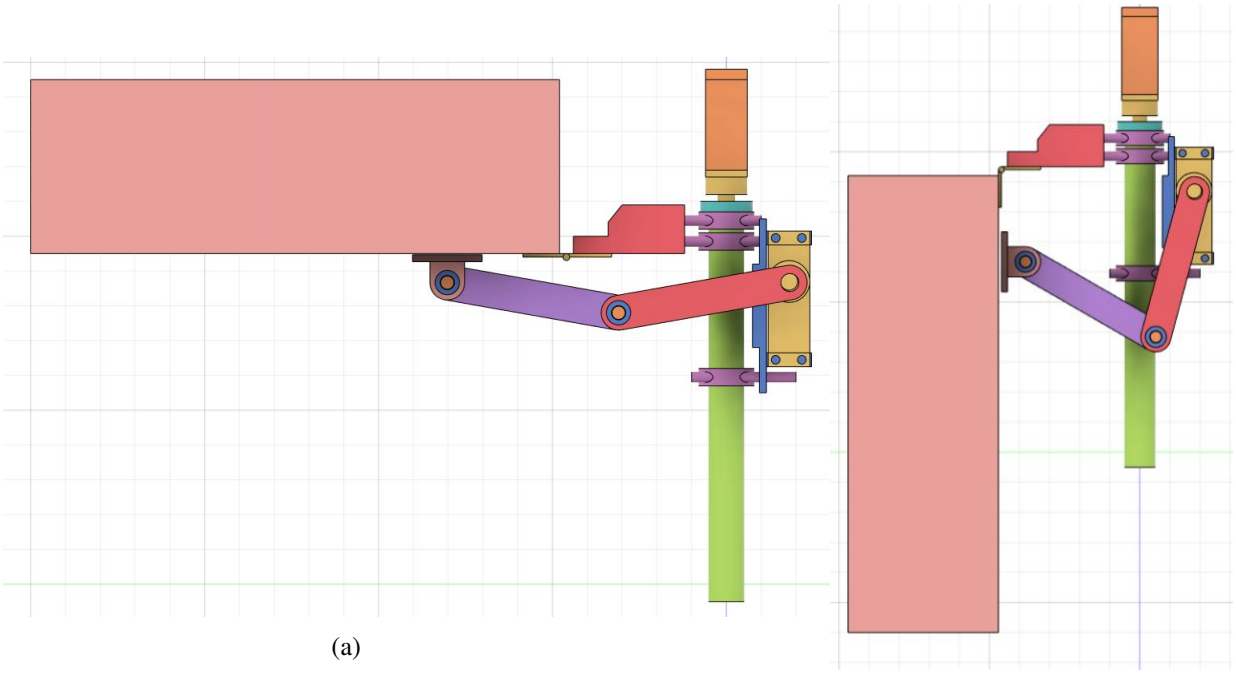
Fig. 17: Removing the lid highlights the internal sorting mechanism mounted above the three dividers in the cylindrical trash can. The figure was created with AutoDesk's Fusion 360 [5].

The top servo is mounted to the underside of the lid and attached to the central rod using 3D printed mounts. The servo-to-lid mount (Figure 26 (6)) is likely only necessary for the working prototype as a fabricated lid could have the necessary mounting points built-in. The top servo then rotates the appropriate amount based on information received from the image recognition process. After rotating above the appropriate section, the servo motors controlling the drop arms will engage and tilt the tray down, dropping the item into the bin (shown in Figure 19). Each drop arm is 3 pieces and 110mm in total length. The drop tray is 150mm by 150mm by 50mm tall and should be large enough to accommodate common items and up to something the size of a large 16oz. aluminium beverage can. This design utilizes two servo motors and two drop arms to ensure the tray is supported adequately. The drop tray mechanism has no lock to help support the weight of items placed on it, only the motor resistance provides support.



(a) (b)

Fig. 18: (a)(b) The sorting mechanism (without the drop tray) and individually colored components. Most of the components are mounted to the central rod (shown in green) which rotates relative to the servo mounted above it (shown in yellow). Both figures were created using AutoDesk's Fusion 360 [5].



(a) (b)

Fig. 19: The drop tray is 150mm by 150mm by 50mm and should accommodate most items. (a) The sorting mechanism in the standard "up" position. (b) The sorting mechanism in the "down" position, when dropping an item into the bin. Both figures were created using AutoDesk's Fusion 360 [5].

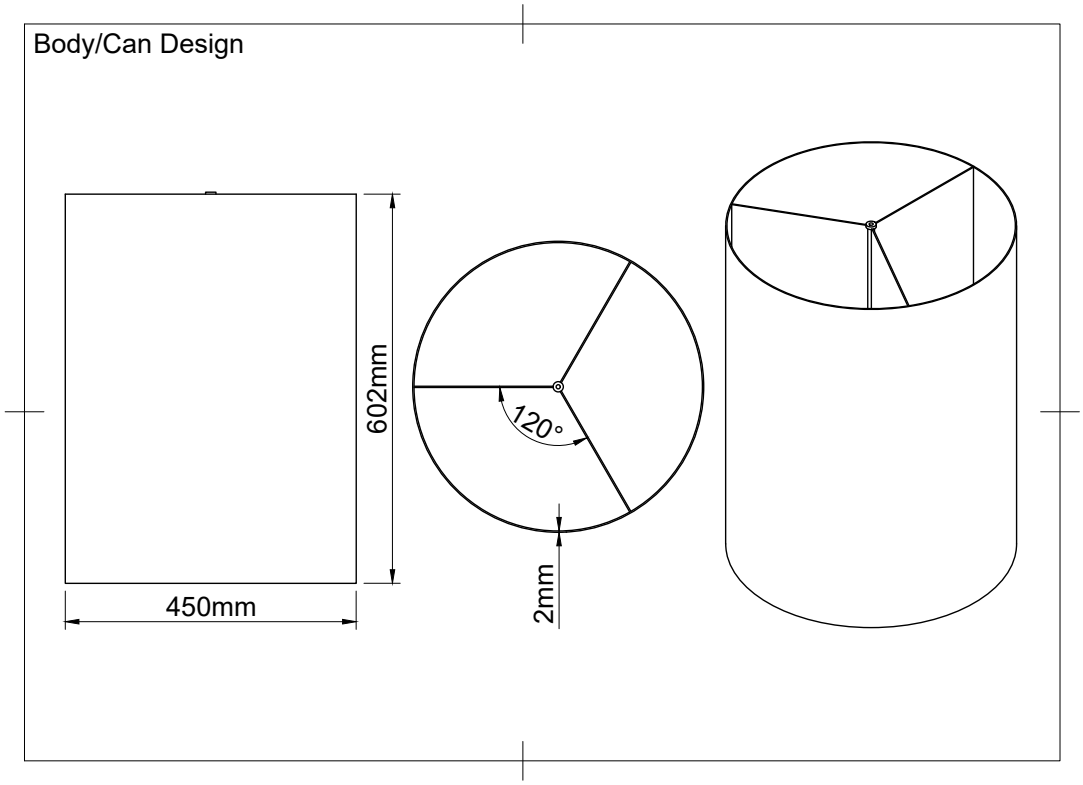


Fig. 20: Body manufacturer drawing. The body is 60cm tall and 45cm in diameter

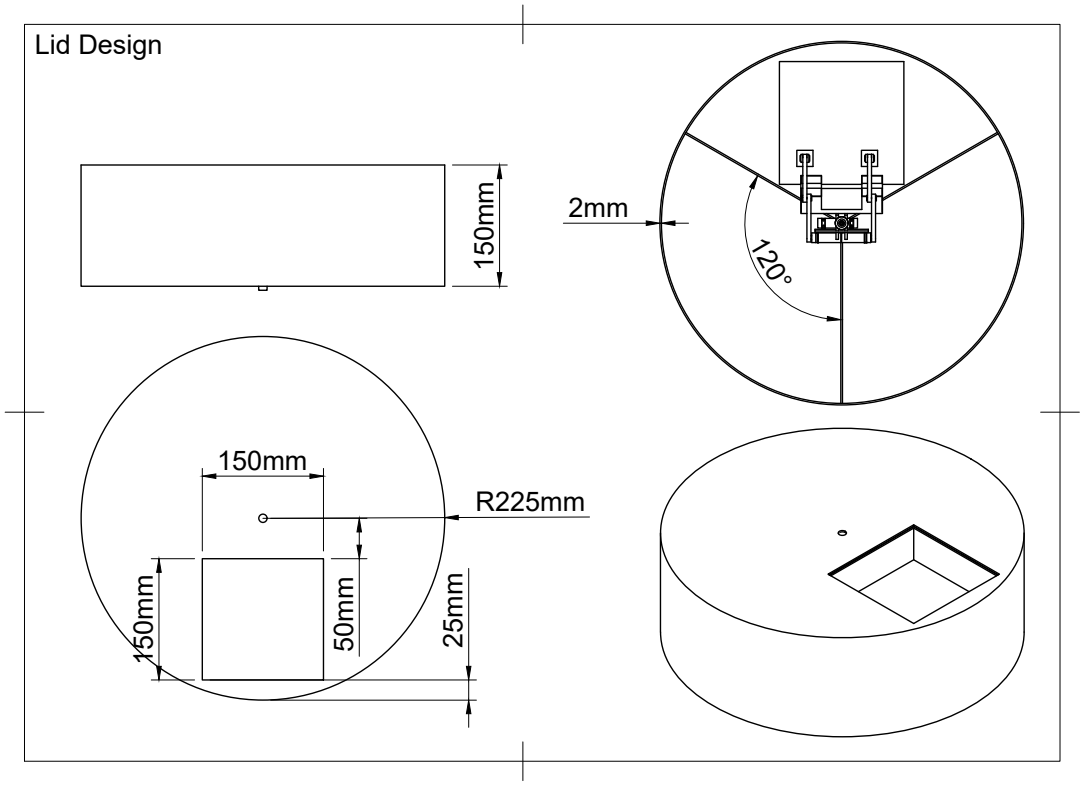


Fig. 21: Lid manufacturer drawing. The opening is 15 by 15 cm and offset 50mm from the center.

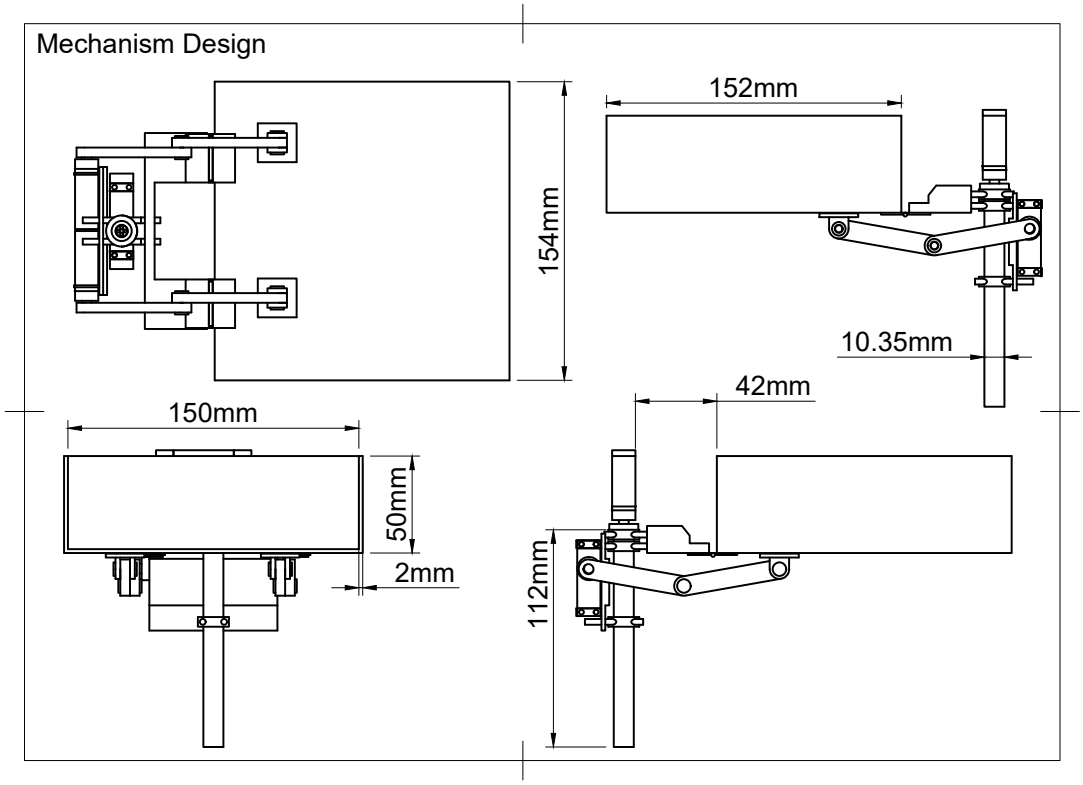


Fig. 22: Sorting mechanism manufacturer drawing. The tray is 15cm by 15cm and 5cm deep. The central rod is about 112mm tall and slots into the body of the garbage can.

VI. Prototype Evaluation

A. Physical Prototype

The functional prototype, a complete trash can with image recognition and a sorting mechanism, is shown in Figure 24. The list of physical parts that we used to create the prototype can be found in Figure 28. We used 6 different 3D printed parts to assist with mounting the servos and trays to the central rotating rod. These 3D printed parts can be found in Figure 26. Looking at Figure 24 the camera can be seen mounted above the opening in the lid that contains the tray. The camera was not able to be mounted on the inside of the can due to the lens not being able to view the entire tray when mounted inside the lid. The ultrasonic sensor for detecting when an item is placed within the tray can be seen on the side of tray facing the opening.

The prototype sorting mechanism can be seen in Figure 25. The 3D printed parts used to mount the arm servos and the tray can be seen. The parts used to connect the arms to the tray can be seen in Figure 27. Also in this figure, the ultrasonic sensor for measuring the amount of trash in each compartment can be seen. The top servo and the 3D printed parts to mount the servo to the top of the lid and the servo to the support rod cannot be seen in the images, but these parts are mounted directly in the center of the lid. The last 3D printed part is mounted in the center of the bottom of the trash can and is used to hold the support rod in place.

The wiring of the prototype can be seen in Figure 23. In order to develop the prototype, it was decided the wiring should be external. This was decided as during the prototyping stage, if new hardware was introduced or changes of the current hardware were to occur, it would be straightforward to make these changes.

Comparing the functional prototype to the design outlined in the previous sections, all of the features from the design were implemented in the prototype. The hardware in the prototype is far less integrated as it would be in the final design but it demonstrates the provided design functions as intended.

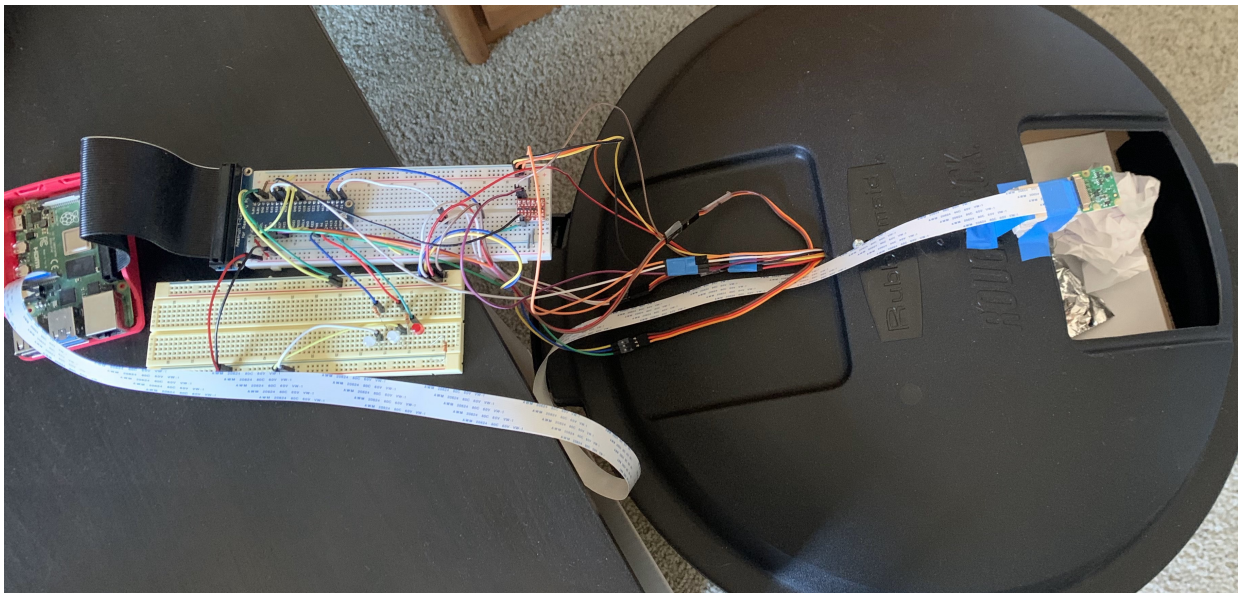


Fig. 23: Prototype Wiring

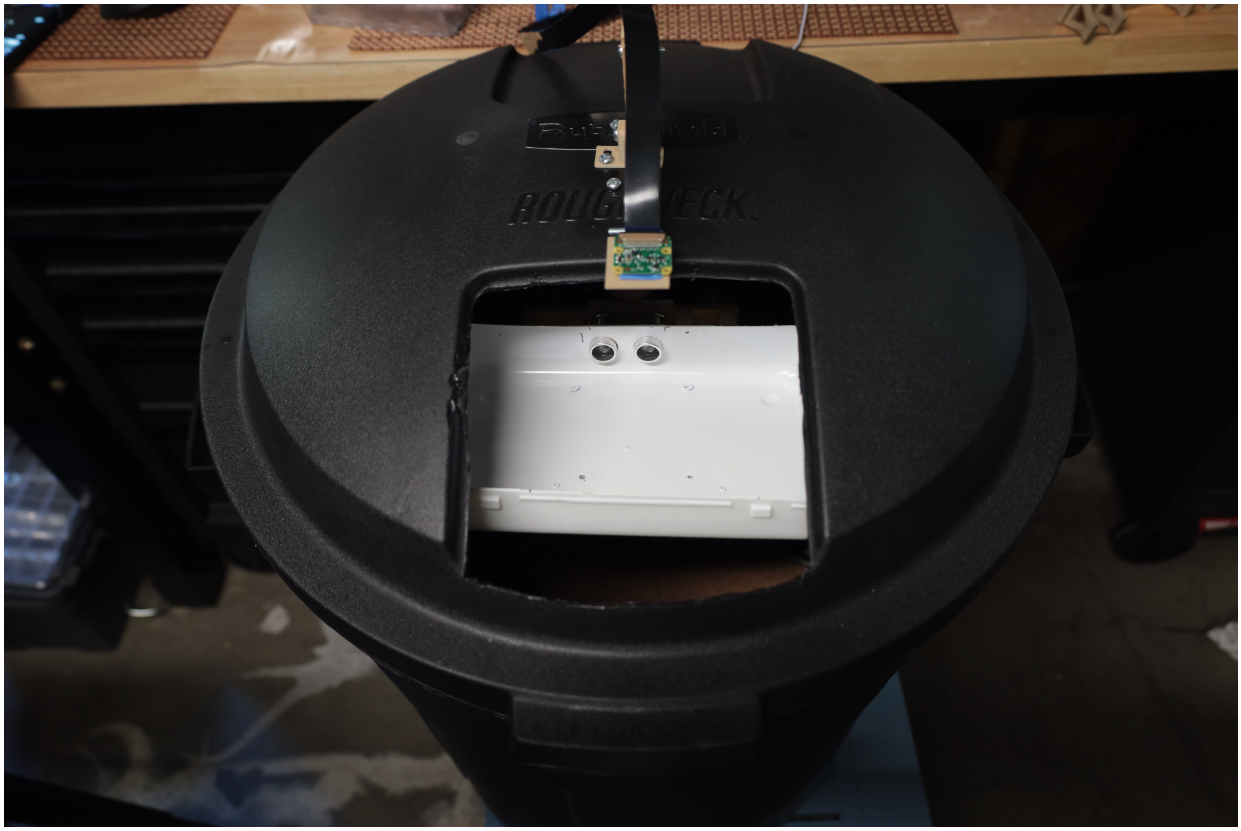


Fig. 24: Prototype Complete Can

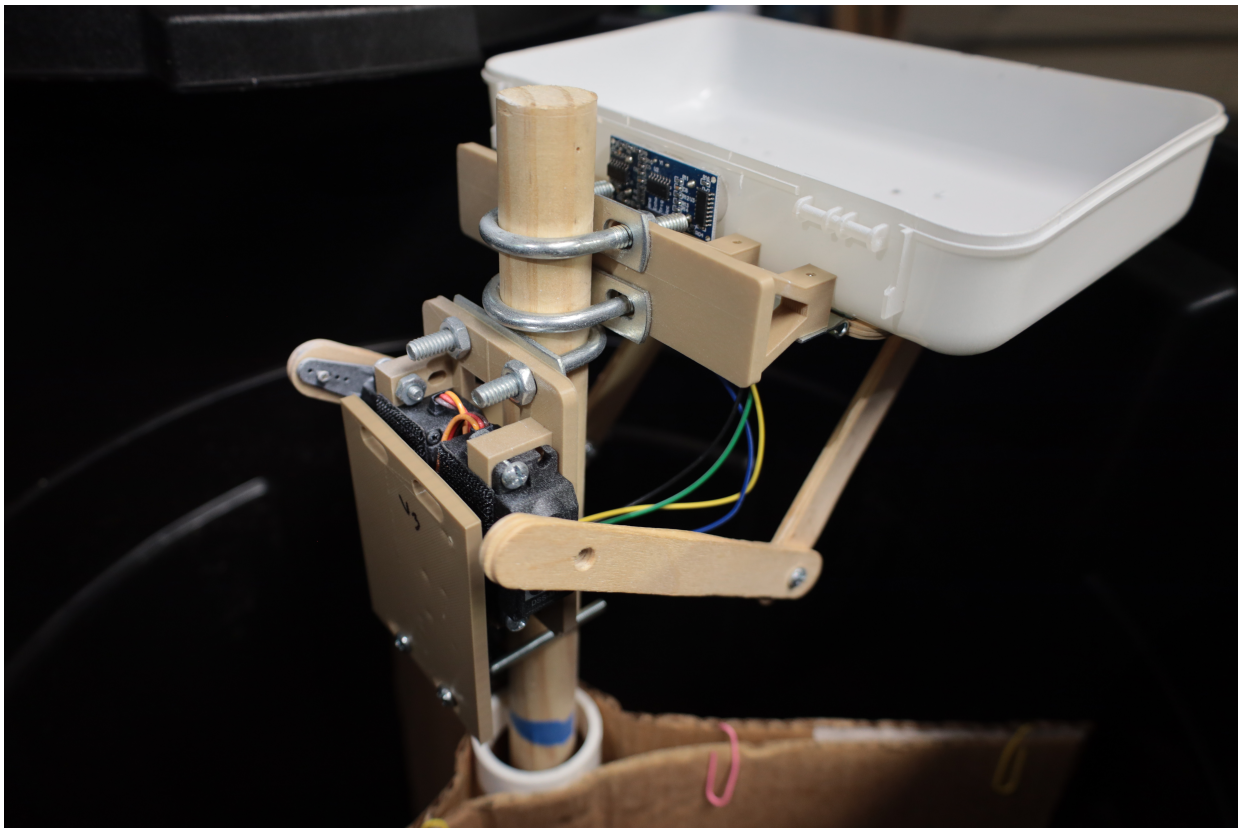


Fig. 25: Prototype Sorting Mechanism

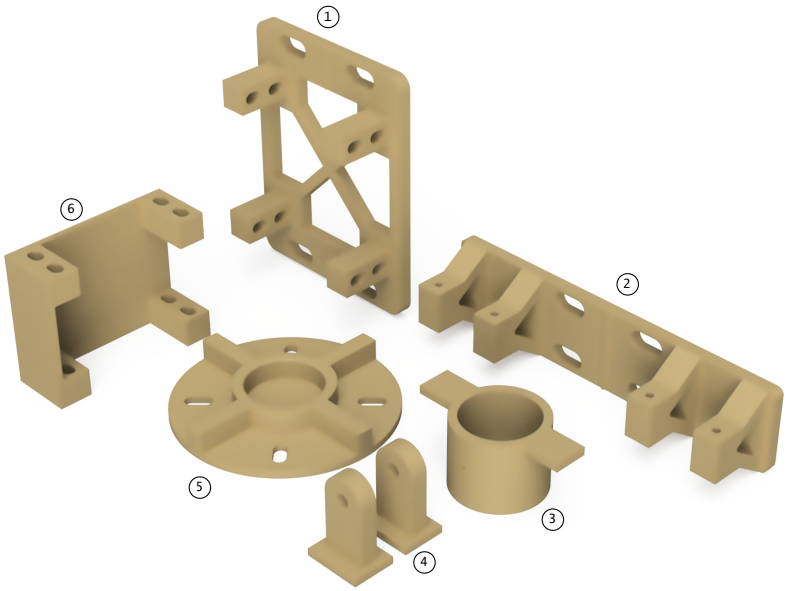


Fig. 26: Prototype 3D Printed Parts **1:** Attaches the arm servos to the back of the central rod. **2:** Attaches the hinges to the tray. **3:** Connects the top servo to the central rod. **4:** Connects the arms to the tray. **5:** Acts as a base plate for the rotating rod, reducing friction. **6:** Attaches the top servo to the bottom of the lid.

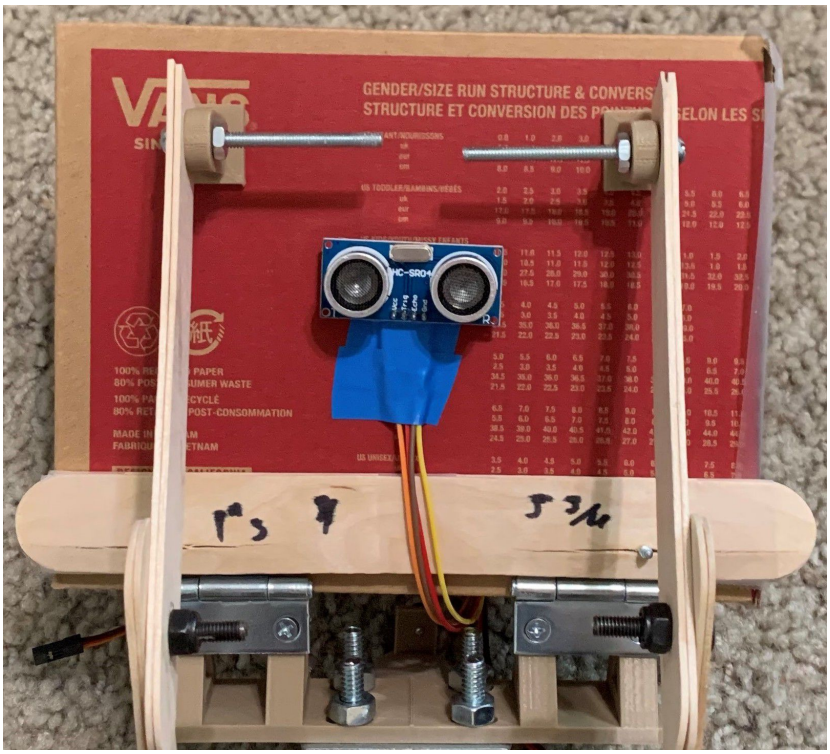


Fig. 27: Prototype Trash Level Sensor

List of Parts	
Part Name	Part Description
Raspberry Pi 4 [10]	Runs image recognition software, controls camera and servos
Raspberry Pi Camera Module v2 [11]	Image capture for item classification
HC-SR04 Ultrasonic Sensor [16]	Item detection and trash level
SER0038 Servo [13]	Provide movement to the sorting mechanism
BOB-12009 Level converter [14]	Voltage conversions for ultrasonic sensor
20 gal 18 inch diameter trash can	Garbage can body and lid
1” diameter x 48”L raw wood dowel	Attached to servos and tray for sorting mechanism
1/2” hinges	Attached to the tray to allow it to move up and down
U-bolt zinc ¼" x 2-5/6"	Attaches 3D printed parts to central rod
#8-32 ¾” machine screws	Attaches servos to 3D printed parts
#8-32 stop nuts	Attaches 3D printed parts to the tray
Various 3D printed parts	Connects servo and tray to central rod

Fig. 28: This table contains a list of all the parts that we used for our design.

B. Testing

The testing for the prototype occurred in various stages. The first stages tested the individual components as they were implemented. The next stage tested the combination of individual components to ensure they worked together as intended. The final stage consisted of testing the complete prototype. The following sections give a brief summary of the tests that occured for each stage and their results. Specific testing methods and results can be seen in Appendix 3.

1) Individual Components:

- Trash Level Detection: The ultrasonic sensor and the corresponding code for it was tested to ensure the sensor was measuring correct distances. Objects were placed in front of the sensor at various distances and the measured distanced from the sensor was compared to the actual distance. These tests demonstrated the sensor was able to accurately measure distances within plus or minus a few centimeters.
- Image Classification Model: The neural network for predicting which compartment a piece of trash belongs in was tested by feeding stock photos from the internet. In these tests we were able to show a prediction accuracy of over 80% when running untrained images through the model.
- Motion Detection: There were two versions of motion detection implemented for this project: one with the camera and one with an ultrasonic sensor. The accuracy of sensing an object placed in front of the camera or in the tray was tested. The results of the tests showed the motion detection with the camera would often lead to false positives or false negatives. However, with the ultrasonic sensor detecting motion, when an object is placed in front of the sensor, the object was always recognized to be there.
- Servo Functionality: The servos responsible for rotating the tray over the correct compartment and dropping the tray to release the item into the compartment were tested by ensuring they rotated the correct amount of degrees. The top servo was able to rotate +120 degrees and -120 degrees to reach both the recycling and compost sections. The left and right arm servos were able to rotate 100 degrees to have enough angle to drop the tray completely.

2) Combination of Components:

- Sorting Mechanism: After the prototype sorting mechanism was constructed, the rotation of the entire mechanism and the dropping of the tray was tested. The testing demonstrated the rotation of the mechanism and the dropping of the tray was successful. The top servo was able to rotate +120 degrees and -120 degrees to reach the left and right compartments. The timing between the arms dropping and lifting the tray gave enough time for an item to fall out. The timing of the top servo waiting for the tray to be lifted back up was also correct.
- Image Classification On Raspberry Pi: The accuracy of the image classification neural network when ran on the Raspberry Pi with images from the Pi camera was tested. Items corresponding to each class (compost, recycling, trash) were captured and the result from the model was compared to the actual class the trash item belonged in. Once again we were able to demonstrate a greater than 80% accuracy with the model. The speed of how long it took the model to develop a prediction for items was also tested in order to ensure a quick operation of the trash can. The speed measured from these tests demonstrated an extremely fast throughput for classifying images.
- Image Classification GPIO: The GPIO communication between the image classification software and the sorting mechanism control software was tested to ensure compatibility between the two systems. The tests ran for this involved running an image through the model and observing the GPIO status and comparing it to the expected GPIO status for each class. This test resulted in complete success with the GPIO pins always being set the correct values for the given input class.
- Trash Level Detection Installed Under Lid: The trash level detection software and circuit was tested inside of the trash can by filling compartments with varying levels of trash. Through these tests we demonstrated the sensor was able to accurately measure the amount of trash in each compartment and when the compartment was full it was able to alert the user through the red LEDs.

3) Complete Prototype:

- Prototype Testing: The prototype was tested to demonstrate it worked as intended. Over 70 items were placed into the tray and the results where recorded. These results can be seen in Fig 29 in the section below.

C. Sorting Accuracy

With our physical prototype we were able to achieve our goal of a sorting accuracy greater than 80%. As seen in Figure 29, we achieved a total sorting accuracy of 82.89% with the 76 waste items tested. The accuracy of each class (compost, recycling, and trash) were also tracked. Compost received the highest accuracy with 91.30%, with recycling following with the second highest accuracy of 82.76%, and trash with the lowest accuracy of 75.00%. The disparity of accuracies between the classes can be attributed to the biased data set we trained our neural network on. The compost class had the largest amount of images, over one thousand images, followed by recycling, around one thousand images, and trash had the least amount of images, around two hundred images. While we did achieve an overall accuracy above the goal we set, we would have liked to achieve over 80% accuracy for all classes. This can be done by simply adding more images to our data set, specifically in the trash category. With a larger and more diverse data set for all classes, it would be possible to achieve an even higher and more accurate trash classification and sorting.

A description of each item tested, the model's prediction for the item, the actual class the item belongs to, and the outputted tensor can be seen in Appendix 6, Figure 68.

D. Next Steps

The first functional prototype met our primary design goal of fully autonomously sorting trash items into recycle, compost, and landfill with an accuracy greater than 80%. Moving forward with the project

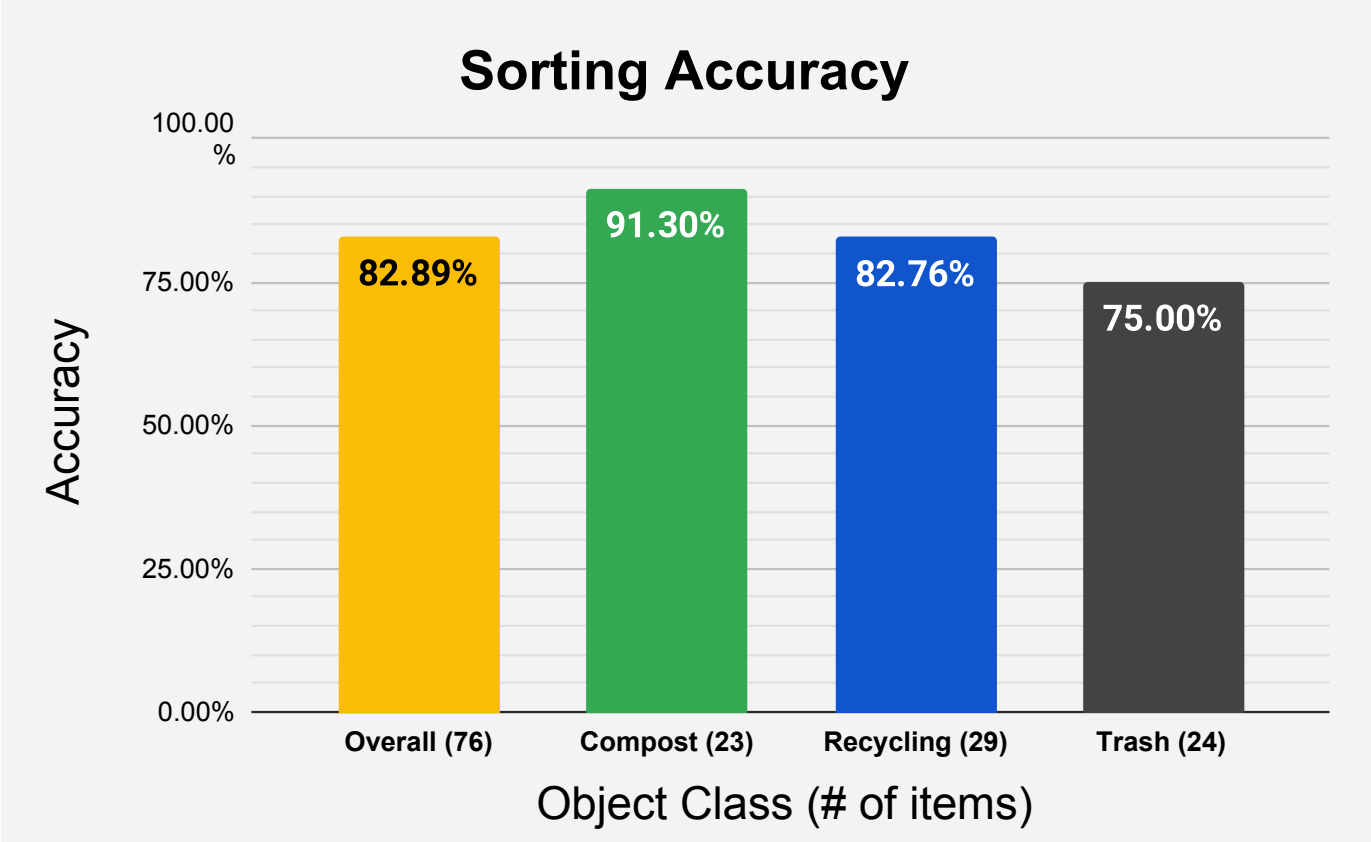


Fig. 29: Sorting Accuracy Results

there are many new additions we would like to add. Regarding the hardware, we would like to research new methods of detecting trash in the tray and developing a more compact sorting mechanism to increase the capacity of the actual compartments. We would also like to begin designing custom PCBs to integrate the wiring within the trash can to make a seamless product. Regarding the image classification model, we would like to gather a larger data set to improve the accuracy of each class. This data set would also include items that seem to belong in one category but actually belong to another, such as a food container that is plastic but because of the food inside it must be put in landfill. With the improvements on the hardware and software, we would like to develop another prototype that closely resembles the aesthetic model. Finally, we would like to put more research into the scalability of SimpleSort. While we believe the current design is optimal for home use, we would like to expand it to larger applications. SimpleSort could be placed in college campuses, airports, or any public place that requires trash bins.

VII. Appendix 1 - Problem Formulation

A. Conceptualizations

Fig. 30: Mind Map

B. Brain Storming Output

- Gardener Robot
  - Scheduled watering
  - Detecting/scaring pests
  - Mowing lawn
    - Large motor
    - Spinning blades
    - Safety issues with blades, flipping over
  - Weed detection
  - Solar panel, battery pack
  - Add and remove features

  - Planter Box gardener
    * Separate devices in each planter box
    * Weed detection with camera
- Cleaning Robot
  - UV light
- Smart Handicap Scooter
  - Avoid collisions
  - Automated driving
- Hard Fall Detection
    * Cycling
    * Hiking
    * Snowboarding
    * Smartphone app to contact emergency services or emergency contacts
  - Every day user
  - If it detects a fall, but then the user starts moving again, don't send the emergency alert.
  - Not sport specific
- Smart Home Sensors
  - Similar to ring, for protecting packages
  - Motion sensor
    * For alerts or data
  - Wireless Connection
    * To a central hub?
- Smart Garbage Sorter
  - Detects the type of waste an item is
  - Puts waste into the correct compartment
- Smart Street Light
  - Turns on at night
  - Turns brighter when motion is detected
- Automated Climate Control System
  - Alerts users to open/close windows depending on weather
  - Set temperatures to certain levels
- Container system for detecting the amount of product in a container
  - Send alert to phone if container is almost empty
    * Figure out what to stock up on at the store
  - Add a timer for perishable goods
  - Hardware
    * Proximity detector
    * Camera

C. Decision Tables

Design 1: Trash Compactor
Electronic pistons on either side of a rectangular trash can that pushes the placed piece of trash over the correct compartment. A servo actuated trap door drops in order to place the trash into the compartment.

Design 2: Teeter-Totter Lid
Single lid that tilts to drop trash into one of only two compartments. Requires a single servo to rotate the platform.

Design 3: Spinning Can/Lid
A circular trash can on wheels is connected to a servo that will spin the can until the correct compartment is below the trash. Once positioned correctly, the trap door holding the trash can drop, allowing the item to fall into the bin.

Design 3A: Spinning Trash Can
Uses servo and wheels to spin the trash can (the lid is stationary). One trap door to drop the trash. Requires more powerful servo.

Design 3B: Spinning Top
Spins the top platform (lid) instead of the can. Also uses a trap door to drop the trash. Requires less force and uses less power to spin the lid than spin the can.

Design 3C: Spinning Brush
Rotating brush to push trash into the correct compartment. Requires 3 trap doors, one for each compartment. Requires less power than spinning the can but can be messy.

We started off with three different sorting mechanism designs. The first design, the Trash Compactor, uses electronic pistons to push the waste items into the correct compartments. The Teeter-Totter Lid design has a tipping lid that allows trash to fall into its corresponding category. The last design, the Spinning Can, consists of spinning a circular can or lid that will rotate until the correct compartment is located under the trash. Once the correct compartment is under the trash, the lid holding the trash can drop, allowing the item to fall into the bin. In order to choose the best design, we weighed each option based on cost, sorting speed, ease of implementation, and hardware required. Based on all of the factors, we chose the spinning can/lid design. It was one of the least costly designs and scored well with the sorting speed, ease of implementation, and hardware required sections.

After deciding on the spinning can/lid design, we created more spinning designs that would improve on the initial design. The Spinning Trash Can is the initial design. The Spinning Top consists of spinning the lid of the garbage can instead of spinning the can itself. The Spinning Brush has a rotating brush that pushes trash into the correct compartment. We weighed each of these designs with the same factors as before and ultimately decided to use the the Spinning Top design because it scored the highest. The Spinning Top was easier to implement than the Spinning Can as it requires spinning less mass. The Spinning Brush design scored low in sorting speed, and we also were concerned that brushing trash across the inside of the can may be unsanitary.

Criteria	Units	Weight	Design 1 ‘Trash Compactor’		Design 2 ‘Teeter-totter lid’		Design 3 ‘Spinning Can’	
			Raw	Weighted	Raw	Weighted	Raw	Weighted
Cost	$	15	2	30	5	75	4	60
Speed at Sorting	⏰	35	4	140	7	245	7	245
Ease of Implementation	⏰	25	3	75	4	100	5	125
Hardware Required	$	25	2	50	5	125	5	125
Totals		**100**		**295**		**545**		**555**

Fig. 31: Sorting Mechanism Decision Table

Criteria	Units	Weight	Design 3A ‘Spinning Trash Can’		Design 3B ‘Spinning Top’		Design 3C ‘Spinning Brush’	
			Raw	Weighted	Raw	Weighted	Raw	Weighted
Cost	$	15	4	60	5	75	3	45
Speed at Sorting	⏰	35	7	245	8	280	6	210
Ease of Implementation	⏰	25	5	125	5	125	4	100
Hardware Required	$	25	5	125	6	150	5	125
Totals		**100**		**555**		**630**		**480**

Fig. 32: Spinning Can Designs Decision Table

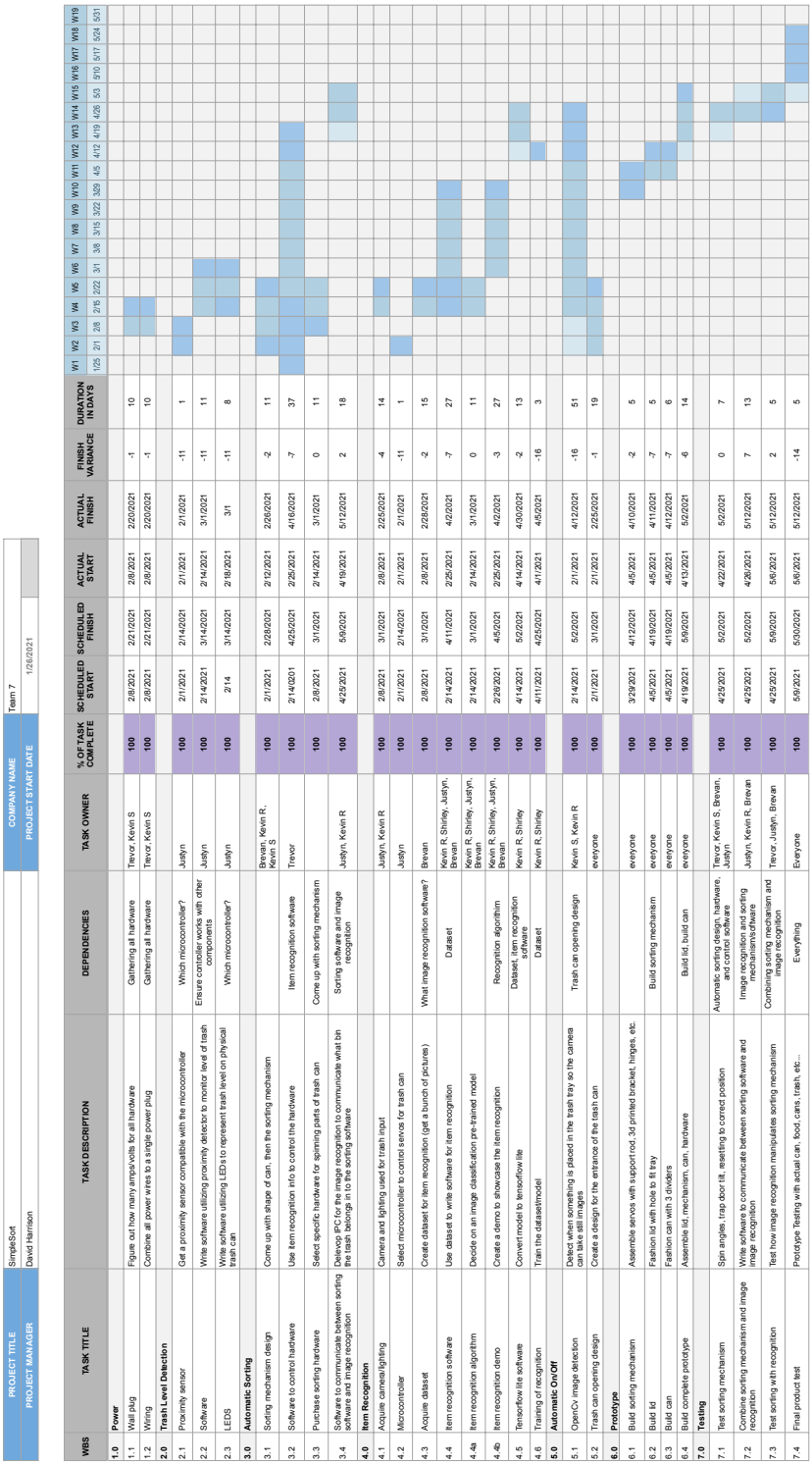
D. Morphological Charts

Fig. 33: Design 3 A,B,C Morphological Chart

The three designs 3A, 3B, and 3C can be visually seen in Figure 33. These designs are early concepts sketched to gain further understanding of each design in order to accurately give them the raw values seen in Figure 32.

VIII. Appendix 2 - Planning

A. Gantt Chart



PROJECT TITLE	SimpleSort	COMPANY NAME	Team 7
PROJECT MANAGER	David Harrison	PROJECT START DATE	1/26/2021

WBS	TASK TITLE	TASK DESCRIPTION	DEPENDENCIES	TASK OWNER	% OF TASK COMPLETE	SCHEDULED START	SCHEDULED FINISH	ACTUAL START	ACTUAL FINISH	FINISH VARIANCE	DURATION IN DAYS
1.0	**Power**										
1.1	Wall plug	Figure out how many amps/volts for all hardware	Gathering all hardware	Trevor, Kevin S	100	2/8/2021	2/21/2021	2/8/2021	2/20/2021	-1	10
1.2	Wiring	Combine all power wires to a single power plug	Gathering all hardware	Trevor, Kevin S	100	2/8/2021	2/21/2021	2/8/2021	2/20/2021	-1	10
2.0	**Trash Level Detection**										
2.1	Proximity sensor	Get a proximity sensor compatible with the microcontroller	Which microcontroller?	Justyn	100	2/1/2021	2/14/2021	2/1/2021	2/1/2021	-11	1
2.2	Software	Write software utilizing proximity detector to monitor level of trash	Ensure compiler works with other components	Justyn	100	2/14/2021	3/14/2021	2/14/2021	3/1/2021	-11	11
2.3	LEDS	Write software utilizing LEDs to represent trash level on physical trash can	Which microcontroller?	Justyn	100	2/14	3/14/2021	2/18/2021	3/1	-11	8
3.0	**Automatic Sorting**										
3.1	Sorting mechanism design	Come up with shape of can, then the sorting mechanism		Brevan, Kevin R, Kevin S	100	2/1/2021	2/28/2021	2/12/2021	2/26/2021	-2	11
3.2	Software to control hardware	Use item recognition info to control the hardware	Item recognition software	Trevor	100	2/14/001	4/25/2021	2/25/2021	4/16/2021	-7	37
3.3	Purchase sorting hardware	Select specific hardware for spinning parts of trash can	Come up with sorting mechanism		100	2/8/2021	3/1/2021	2/14/2021	3/1/2021	0	11
3.4	Software to communicate between sorting software and image recognition	Delevop IPC for the image recognition to communicate what bin the trash belongs in to the sorting software	Sorting software and image recognition	Justyn, Kevin R	100	4/25/2021	5/9/2021	4/19/2021	5/12/2021	2	18
4.0	**Item Recognition**										
4.1	Acquire camera/lighting	Camera and lighting used for trash input		Justyn, Kevin R	100	2/8/2021	3/1/2021	2/8/2021	2/25/2021	-4	14
4.2	Microcontroller	Select microcontroller to control servos for trash can		Justyn	100	2/1/2021	2/14/2021	2/1/2021	2/1/2021	-11	1
4.3	Acquire dataset	Create dataset for item recognition (get a bunch of pictures)	What image recognition software?	Brevan	100	2/8/2021	3/1/2021	2/8/2021	2/28/2021	-2	15
4.4	Item recognition software	Use dataset to write software for item recognition	Dataset	Kevin R, Shirley, Justyn, Brevan	100	2/14/2021	4/11/2021	2/25/2021	4/2/2021	-7	27
4.4a	Item recognition algorithm	Decide on an image classification pre-trained model		Kevin R, Shirley, Justyn, Brevan	100	2/14/2021	3/1/2021	2/14/2021	3/1/2021	0	11
4.4b	Item recognition demo	Create a demo to showcase the item recognition	Recognition algorithm	Kevin R, Shirley, Justyn, Brevan	100	2/26/2021	4/5/2021	2/25/2021	4/2/2021	-3	27
4.5	Tensorflow lite software	Convert model to tensorflow lite	Dataset, item recognition software	Kevin R, Shirley	100	4/14/2021	5/2/2021	4/14/2021	4/30/2021	-2	13
4.6	Training of recognition	Train the dataset/model	Dataset	Kevin R, Shirley	100	4/11/2021	4/25/2021	4/11/2021	4/5/2021	-16	3
5.0	**Automatic On/Off**										
5.1	OpenCv image detection	Detect when something is placed in the trash tray so the camera can take still images	Trash can opening design	Kevin S, Kevin R	100	2/14/2021	5/2/2021	2/1/2021	4/12/2021	-16	51
5.2	Trash can opening design	Create a design for the entrance of the trash can		everyone	100	2/1/2021	3/1/2021	2/1/2021	2/25/2021	-1	19
6.0	**Prototype**										
6.1	Build sorting mechanism	Assemble servos with support rod, 3d printed bracket, hinges, etc.		everyone	100	3/29/2021	4/12/2021	4/5/2021	4/10/2021	-2	5
6.2	Build lid	Fashion lid with hole to fit tray	Build sorting mechanism	everyone	100	4/5/2021	4/19/2021	4/5/2021	4/11/2021	-7	5
6.3	Build can	Fashion can with 3 dividers		everyone	100	4/5/2021	4/19/2021	4/5/2021	4/12/2021	-7	6
6.4	Build complete prototype	Assemble lid, mechanism, can, hardware	Build lid, build can	everyone	100	4/19/2021	5/9/2021	4/13/2021	5/2/2021	-6	14
7.0	**Testing**										
7.1	Test sorting mechanism	Spin angles, trap door tilt, resetting to correct position	Automatic sorting design, hardware, and control software	Trevor, Kevin S, Brevan, Justyn	100	4/23/2021	5/2/2021	4/22/2021	5/2/2021	0	7
7.2	Combine sorting mechanism and image recognition	Write software to communicate between sorting software and image recognition	Image recognition and sorting mechanisms/software	Justyn, Kevin R, Brevan	100	4/25/2021	5/2/2021	4/26/2021	5/12/2021	7	13
7.3	Test sorting with recognition	Test how image recognition manipulates sorting mechanism	Combining sorting mechanism and image recognition	Trevor, Justyn, Brevan	100	4/25/2021	5/9/2021	5/6/2021	5/12/2021	2	5
7.4	Final product test	Prototype Testing with actual can: food, cans, trash, etc...	Everything	Everyone	100	5/6/2021	5/30/2021	5/6/2021	5/12/2021	-14	5

Fig. 34: Gantt Chart

B. CPM ADP Table/Graph

No.	Activity	Duration	Depends On
A	Wall Plug: How many amps/volts for all hardware	2	
B	Proximity detector for trash detection	5	
C	Sorting mechanism design	5	
D	Data set for item recognition	4	
E	Automatic on/off	11	
F	Test camera/lighting	4	
G	Build prototype trash can/lid	10	
H	LEDs for trash level detection	11	B
I	Software for item recognition	10	D
J	Software to communicate with item recognition and sorting mechanism	1	I, K
K	Sorting mechanism control software	10	C
L	Test sorting mechanism (spin angles, trap door tilt, etc.)	1	C
M	Test image recognition	2	I
N	Put together image recognition and sorting mechanism	1	L, M, J
O	Test sorting with image recognition	1	N
P	END: Test final product	3	All

Fig. 35: CPM ADP Table

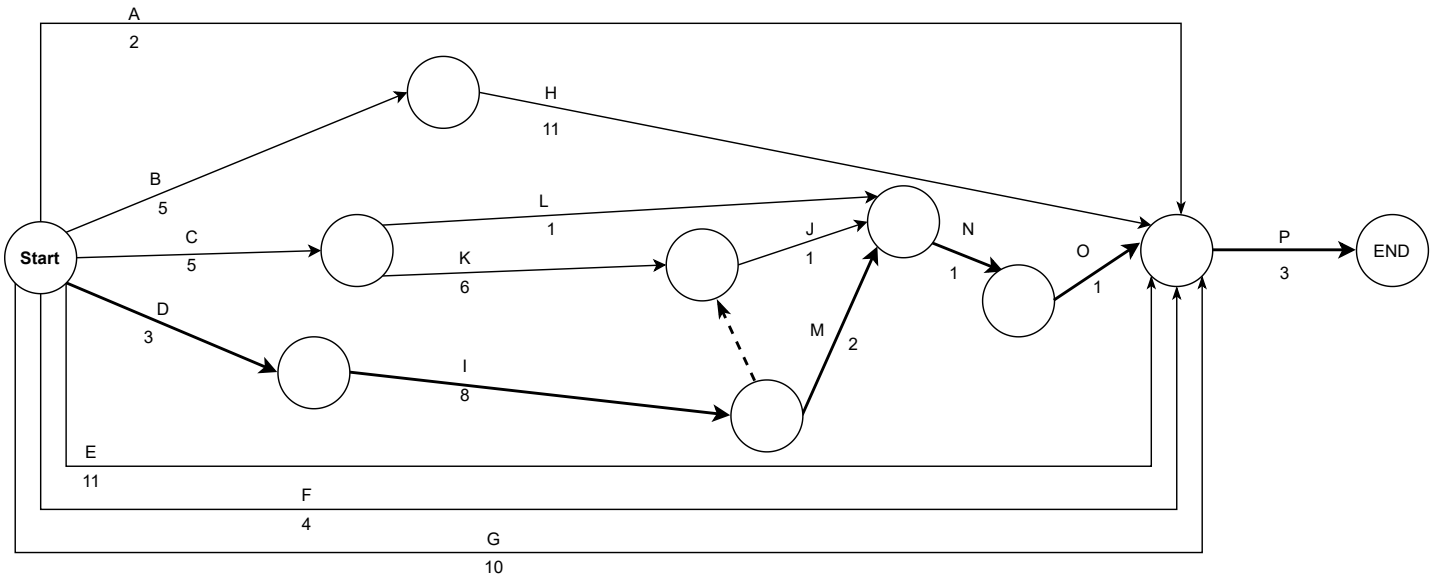


Fig. 36: CPM ADP Graph

C. Division of Labor

Task	Team Members
Motion Detection	Kevin Ryan
Image Classification and Processing	Shirley Phuong, Kevin Ryan, Justyn Duthler
Sorting Mechanism	Brevan Chun, Trevor Carleton, Justyn Duthler, Kevin Spence
Trash Level Detection	Justyn Duthler
Final Prototype Testing/Production	Everyone

Fig. 37: Division of Labor

D. Collaboration

When we first started the project, we all stated our strengths and interests. Based on these things, we were able to assign certain tasks for each person. Despite assigning tasks, everyone was open to helping each other if anyone was having trouble with their portion of the project.

In order to keep everyone up to date with all parts of the project, we pushed all of our code to one code repository [8]. We split up different aspects of the project into separate folders to keep everything organized. By doing this, we were able to perform unit tests for each component of our project before putting everything together for our prototype.

Our team also held weekly meetings over Discord in order to give updates on our progress as well as plan out our goals for the upcoming week. We kept track of things we discussed during our meetings in a shared Google Drive. This is where we worked on class submissions as well as where we shared picture and video updates of our prototype.

All of our test results during the prototyping phase were documented on Overleaf. We used Overleaf to create this design document and we were able to frequently update it as we gained progress. The test plans and results can be seen in the next section.

IX. Appendix 3 - Test Plans & Results

A. Test Plans

Trash Level Detection: Single Sensor Test

1) Introduction
   This test involves testing the sensor circuit and software for the trash level detection of SimpleSort. One HC-SR04 ultrasonic sensor will be tested to ensure it can measure the distance to the nearest object. The sensor will be tested at varying distances and the distance the sensor code reports will be compared to the actual distance measured from the sensor to the object.

2) References
   The circuit for the trash level detection can be seen in Figure 9. For this test, only one sensor is connected to the Raspberry Pi and tested. This can be seen in the image in the corresponding testing result section.

3) Features To Be Tested
   - Distance measurement of single sensor

4) Features Not To Be Tested
   - Testing does not include the two other sensors in the final design
   - Testing will not occur with the sensor attached to the lid of the trash can

5) Approach
   On a flat surface, tape is placed at 15cm, 30cm, 45cm, 60cm, and 75cm. These distances cover the entire length of the final trash can, allowing testing to show the accuracy of the trash level detection circuit. Place the trash level detection circuit exactly where the measurements were drawn from. With the software running, place an object, such as a can, at 75cm. Observe the distance printed to the terminal and ensure the corresponding LED is off. Repeat this step for each distance up until 15cm. Then place the object less than 15cm. Observe the distance printed on the terminal. Repeat these previous steps with the object moving back and forth. For example move the object to 10cm, then to 75cm, then to 20cm, and so on.

6) Pass/Fail Criteria
   Pass:
   - The trash level detection circuit is able to accurately display all distances, within plus or minus 3cm.

   Fail:
   - The trash level detection circuit cannot accurately display all distances, greater than 3cm difference.
   - The program crashes, hangs, or does not run continuously.

Trash Level Detection: Three Sensor Test

1) Introduction
   This test will cover the trash level detection feature of SimpleSort. The three HC-SR04 ultrasonic sensors will be tested to ensure they properly measure the distance to the nearest piece of trash when they are installed in the final product. Each sensor and the distance it measures will be individually tested. While the sensor program as a whole monitors all three of the sensors, each sensor needs to individually measure and report correct distances.

2) References
The circuit for the trash level detection can be seen in Figure 9.

3) Features To Be Tested
- Distance measurement
- LED off when object is at a distance greater than 15cm
- LED on when object is at a distance less than 15cm

4) Features Not To Be Tested
- Testing will not occur with the circuit attached to the lid of the trash

5) Approach
On a flat surface, tape is placed at 15cm, 30cm, 45cm, 60cm, and 75cm. These distances cover the entire length of the final trash can, allowing testing to show the accuracy of the trash level detection circuit. Place the trash level detection circuit exactly where the measurements were drawn from. With the software running, place an object, such as a can, at 75cm. Observe the distance printed to the terminal and ensure the corresponding LED is off. Repeat this step for each distance up until 15cm. Then place the object less than 15cm. Observe the distance printed on the terminal and ensure the corresponding LED is on. Move the object extremely close to the sensor, within 2cm, and observe the LED is still on. Repeat this process but change the order of the distances. For example starting less than 15cm, moving the object to 30cm, moving back to 15cm, etc... This random movement will ensure the software can handle any changes of trash level, such as when removing the trash. This process will be repeated for each of the three ultrasonic sensors.

6) Pass/Fail Criteria
Pass:
- The trash level detection circuit is able to accurately display all distances, within plus or minus 3cm.
- The LED for each sensor turns on when an object is within 15cm of the sensor.

Fail:
- The trash level detection circuit cannot accurately display all distances within 3cm of margin.
- The LED for each sensor turns on when an object is within 15cm of the sensor.
- The circuit turns on the corresponding LED when the object is greater than 15cm away
- The circuit turns off the corresponding LED when the object is within 15cm.
- The program crashes, hangs, or does not run continuously.

Trash Level Detection Installed in Lid

1) Introduction
This test will cover the trash level detection circuit when it is installed in the lid of the prototype. Each HC-SR04 sensor will be installed around the circumference of the trash bin over the corresponding section. As with the previous tests, each sensor and the distance it measures will be individually tested. The LED functionality for each sensor will be tested as well.

2) Features To Be Tested
- Distance measurement
- LED off when the object is at a distance greater than 15cm
- LED on when the object is at a distance less than 15cm

3) Features Not To Be Tested
   - This test does not cover measurements when the sorting mechanism is actively moving.

4) Approach
   Start with the trash completely empty. Record the measurements from all three sensors. Then place an object or objects that fill the corresponding sections half way. Record the measurements from all three sensors and record the state of the LEDs for each section. Then place an object or objects that fill the corresponding sections within 15cm of the sensors. Record the measurements from the sensors and record the state of the LEDs for each section. Then repeat the previous steps in different orders to ensure the circuit and software react correctly to the change in distances.

5) Pass/Fail Criteria
   Pass:
   - The circuit is able to accurately display all the distances, within plus or minus 3cm.
   - The LED for each sensor turns on when an object is within 15cm of the sensor.

   Fail:
   - The circuit cannot accurately display all the distances, greater than a 3cm difference.
   - The LED for each sensor turns on when an object is greater than 15cm away
   - The LED for each sensor turns off when an object is less than 15cm away.
   - The program crashes, hangs, or does not run continuously.

Trash Level Detection Installed on Bottom of Sorting Tray

1) Introduction
   This test will cover the trash level detection circuit when it is installed on the bottom of the sorting mechanism tray. This new trash level detection circuit can be seen in figure 9. The distance measured by the sensor for each compartment will be tested as well as the LEDs.

2) Features To Be Tested
   - Distance Measurements
   - LED off when object is at a distance greater than 15cm
   - LED on when object is at a distance less than 15cm

3) Features Not To Be Tested
   - All features of the trash level detection will be tested

4) Approach
   The same approach from the previous test, Trash Level Detection Installed in Lid, is taken for this test. However, because the sensor is installed on the bottom side of the sorting tray, the testing involves using a another program that spins and drops the lid.

5) Pass/Fail Criteria
   Pass:
   - The circuit is able to accurately display all distances, within plus or minus 3cm
   - The LED for each compartment turns on when an object is within 15cm of the sensor.

   Fail:
   - The circuit cannot accurately display all the distances, greater than a 3cm difference.
   - The LED for each sensor turns on when an object is greater than 15cm away
   - The LED for each sensor turns off when an object is less than 15cm away.

- The program crashes, hangs, or does not run continuously.

Image Classification Model Tested With Stock Photos

1) Introduction
   The model in this test implements transfer learning with the pre-trained MobilenetV2 model. The code is written in Python and uses TensorFlow to train the model. Once the model is trained and saved, it is tested using saved stock photos from the internet.

2) Features To Be Tested
   - Accuracy of classification
   - Percent confidence for each item

3) Features Not To Be Tested
   - Accuracy of classification with Pi camera photos

4) Approach
   Before testing, save multiple photos of different types of waste in a test folder. Write a basic testing program that takes in the path to the image. Use TensorFlow and OpenCV to resize the image and grab the saved model. Use model.predict() to get the class prediction and the percent confidence for each image. The path to the image can be changed and the program can be re-run for each image.

5) Pass/Fail Criteria
   Pass:
   - Images are sorted into the right category
   - High confidence percentage for each item

   Fail:
   - Images are sorted into the wrong category
   - Low confidence percentage even if the class is correct

Image Classification TensorFlow Lite Model Tested With Stock Photos

1) Introduction
   This model was obtained by converting the previously saved model into a TensorFlow Lite model. This was done due the long runtime of the previous model on a Raspberry Pi. It uses the same MobilenetV2 pre-trained model and it was tested with the same stock photos to ensure consistency between the two models.

2) Features To Be Tested
   - Accuracy of classification
   - Confidence for each classification

3) Features Not To Be Tested
   - Accuracy of classification with Pi camera photos

4) Approach
   Before testing, save multiple photos of different types of waste in a test folder. Write a basic program to loop through the test folder and produce output tensors for each of the photos. This is done by running interpretor.get_tensor(). The highest number in the tensor corresponds to the

classification result.

5) Pass/Fail Criteria
Pass:
- Images are sorted into the right category
- Tensor displayed shows a high confidence

Fail:
- Images are sorted into the wrong category

Image Classification Model Tested on Raspberry Pi

1) Introduction
This test captures and processes images from the Raspberry Pi camera and feeds them into the image classification model. A program will take a photo with the camera and predict a class using the same TensorFlow model.

2) Features To Be Tested
- Accuracy of classifications
- Speed of classifications

3) Features Not To Be Tested
- Classification confidence

4) Approach
Point camera at various objects and start image processing software. Press a key to take a picture and predict the classification for that item.

5) Pass/Fail Criteria
Pass:
- Object is classified correctly
- Total run-time is under 30 seconds

Fail:
- Object is incorrectly classified
- Total run-time is over 30 seconds

Image Classification TensorFlow Lite Model Tested on Raspberry Pi

1) Introduction
This test captures and processes images from the Raspberry Pi camera and feeds them into the TensorFlow Lite image classification model. A program will take a photo with the camera and predict a class using the same TensorFlow Lite model from the stock photos test.

2) Features To Be Tested
- Accuracy of classifications
- Speed of classifications

3) Features Not To Be Tested
- Classification confidence

4) Approach
Point the camera at various objects and start image processing software. Press a key to take a photo and from there, record the time it takes for the model to classify the image. Take note of which class the image is put into.

5) Pass/Fail Criteria
Pass:
- Object is classified correctly
- Classification time is under 10 seconds

Fail:
- Object is incorrectly classified
- Classification time is over 10 seconds

Image Classification Speed Test

1) Introduction
This test compares the speed of the image processing and classification using TensorFlow vs. TensorFlow Lite models.

2) Features To Be Tested
- Speed of classifications

3) Features Not To Be Tested
- Classification confidence
- Classification accuracy

4) Approach
Run the image processing program and use bash time command to measure runtime. After the photo is taken, record the time it takes for the model to classify the image.

5) Pass/Fail Criteria
Pass:
- Average speed is fastest

Fail:
- Average speed is slowest

Image Processing GPIO

1) Introduction
This test will test whether Raspberry Pi activates the correct GPIO depending on how a piece of trash has been classified. These GPIO outputs will be used to tell the sorting mechanism software what compartment the item needs to be placed in.

2) Features To Be Tested
- GPIO output depending on classification

3) Features Not To Be Tested
- Classification accuracy

4) Approach
Connect an RGB LED to pins 4 and 5 of Raspberry Pi and run image processing software. Pins will be connected to red and blue wires of LED, so the possible output colors are red, blue, purple, and off. Pin 4 is connected to red and Pin 5 is connected to blue.

	GPIO 4	GPIO 5
standby	0	0
compost	0	1
recycle	1	0
trash	1	1

Fig. 38: Output to Corresponding GPIO

5) Pass/Fail Criteria
Pass:
- Standby: Purple
- Recycle: Blue
- Compost: Red
- Trash: off (no light)

Fail:
- Standby: Red, Blue, or off
- Recycle: Red, Purple, or off
- Compost: Blue, Purple, or off
- Trash: Red, Blue, or Purple

Motion Detection

1) Introduction
This test uses the Raspberry Pi camera or ultrasonic sensor (depending on iteration) to detect when something has been placed in front of the camera

2) Features To Be Tested
- Camera can detect any change in what is in front of it, as well as avoiding false positives. Iterations being tested are startV1.py, startV2.py, startV4.py, and startV6.py.

version	method	sample time interval	threshold	successive positives
startV1	camera	1 second	50	1
startV2	camera	1 second	60	3
startV4	camera	0.5 seconds	50	4
startV6	ultrasonic sensor	60ms	13cm	1

Fig. 39: Different Iterations of Motion Detection

For the camera method, the threshold refers to how many pixels have changed after reformatting the image to 10x10 and for the ultrasonic sensor the threshold refers to the maximum distance measured before indicating motion.

3) Features Not To Be Tested
   - Whether or not an actual trash item to be sorted is placed in front of camera

4) Approach
   Place hand 10cm from camera and observe if terminal prints "motion detected". Terminal will print "motion not detected" if no motion is detected during the interval.

5) Pass/Fail Criteria
   Pass:
   - Software accurately detects if there is a change to camera frame

   Fail:
   - Software incorrectly detects "motion detected" if camera frame has not changed
   - Software incorrectly detects "motion not detected" if camera frame has changed

Servo Functionality

1) Introduction
   This test will involve the three SER0038 servos and the C program that is used to control them. There will be two separate testing phases for the servos. In the first test each servo will be tested individually to simply see if the servo is operational. In the next test all three servos will be tested at the same time with the code that will be used in the final design. The program will take an input for each trash compartment in use (trash, recycle, and compost) and the servos will be monitored to see if they are rotating the correct amount at the appropriate times.

2) Features To Be Tested
   - Functionality of each servo
   - Rotation of each servo individually
   - Timing of rotation individually
   - Rotation of each servo as a group with final code
   - Timing of rotation as a group with final code

3) Features Not To Be Tested
   - Functionality of the servos implemented in the physical trash can will not be tested. Instead this will be done in the prototype testing.

4) Approach
   a) Test 1
      - Plug the appropriate servo wires into the ground and 5V power of the Raspberry Pi and plug the signal wire into pin 25.
      - Run the test code called servotest.
      - Verify that the servo rotates back and forth 270 degrees, pausing at the end of each rotation for 3 seconds.
      - Repeat the process for each servo to verify that they all work correctly.
   b) Test 2
      - First measure the top rotating servo which is used to rotate the compartment that holds the trash item.
      - Enter each type of item into the program and observe the servo and measure the angle that it turns.

- For the trash compartment the servo should stay in place. For compost it should rotate 120 degrees and for recycle it should rotate 120 degrees in the other direction.
- After each command it should immediately move to the correct compartment, pause for 5 seconds, then return to the original trash position.
- The other two servos operate the arms that will release the trash into the compartments. Test these by observing them rotate 100 degrees at the same time after each of the compartment commands is entered into the program.
- These servos should rotate then pause for 3 seconds and then rotate back to the original position.
- Once it is determined that all 3 servos work independently they should be tested together.
- Enter each trash, recycle, and compost input. Observe that the top servo rotates to the correct compartment, 1 second passes, the two arm servos rotate, 3 seconds pass, they return to their original position, 1 second passes, and the top rotating servo returns to the original position.

5) Pass/Fail Criteria

Pass:

- All three servos are operational.
- All three servos rotate the correct amount.
- All three servos rotate at the correct time.

Fail:

- Any servo doesn't function from test 1.
- Any servo rotates too much, too little, or not at all.
- The servos don't rotate in the correct direction.
- The servos don't rotate with appropriate timing in respect to each other.

Sorting Mechanism Functionality

1) Introduction

These tests will check to see if the sorting mechanism, composed of the three SER0038 servos, is functional. The first test will ensure that the top rotating servo is capable of rotating the trash tray consistently. The second test will check to see if the other two servos are capable of lowering and raising the trash tray.

2) Features To Be Tested

- Rotating the tray
- Raising and lowering the tray

3) Features Not To Be Tested

- Functionality with the image recognition
- Functionality with the motion detection
- Functionality with trash in the tray

4) Approach

a) Test 1

- Plug the appropriate servo wires into the ground and 5V power of the Raspberry Pi and plug the signal wire into pin 17.
- Run the test code called topservo. Use commands 'r' and 'c' to test the top servo functionality.

- Verify that the servo rotates clockwise or counter-clockwise 120 degrees from the starting position, pausing at the end of each rotation for 3 seconds.
- Mount the servo to the lid and to the central rod. With the rod and servo mounted to the garbage can, run the test again. Ensure the servo is able to turn the rod clockwise or counter-clockwise 120 degrees from the starting position. Take note of any resistance the servo may face when turning the rod.

b) Test 2

- With the two arm servos mounted to the rod, plug the appropriate wires into ground and 5V power on the Raspberry Pi. With the tray mounted close to you, and the servos mounted behind, plug the left servo control wire into pin 22 and the right servo control wire into pin 27. Do not attach the tray and arms to the servos just yet.
- Run the topservo test program. Use commands ‘d’, ‘u’, and ‘t’ to test the functionality of the arm servos.
- Ensure that the servos are rotating in the proper direction: the left servo should rotate clockwise when going down, the right servo should go counter-clockwise when going down and opposite when going up. The two arm servos should turn near simultaneously when moving.
- Attach the arms to the servos and the tray with the tray in the up position by default. Re-run the topservo program using previously mentioned commands to test functionality with the tray attached.

5) Pass/Fail Criteria

Pass:

- Servo motor is able to rotate the trash tray above the correct compartment, then rotate back to the starting position.
- Servo motor is able to raise and lower the tray.

Fail:

- Servo motor is unable to spin the trash tray.
- Servo motor does not rotate the correct distance when connected to the trash tray.
- Servo motor is unable to lower and raise the trash tray.

Sorting Actual Trash

1) Introduction

This test will check to see if the SER0038 servo motors remain functional with trash placed on the tray. The first test will check if the tray has enough support to hold up and sort everyday trash. The second test will test the limits of the sorting mechanism by attempting to sort some of the heavier items that may be placed in an everyday trash can. These tests will both also be testing if the servo being used to spin the sorting mechanism rod remains functional with additional weight on the tray.

2) Features To Be Tested

- Sorting mechanism’s ability to support and rotate trash placed in the tray
- Sorting mechanism tray drop angle

3) Features Not To Be Tested

- Functionality of the sorting mechanism with the image recognition software
- Whether the sorting mechanism design will remain functional with the image recognition hardware connected to the trash can

- Functionality with the trash detection sensors attached

4) Approach
   a) Test 1
      - Connect each of the servos to the 5V power, red wire on servo, and ground, brown wire on servo, of the Raspberry Pi. Connect the PWM pin of the servo, orange wire, to the corresponding pin locations of the Raspberry Pi, pin 22 for the left arm servo, pin 27 for the right arm servo, pin 17 for the rod servo.
      - Run topservo test code with the servos connected to ensure that the trash, compost, and recycling functionalities are working prior to testing with any items in the tray.
      - Run topservo test code and type in the trash input with a small piece of trash placed in the tray.
      - Repeat the previous step, testing with small pieces of trash, compost, and recycling .
   b) Test 2
      - After ensuring the sorting mechanism is functioning properly with small pieces of trash, continue testing the sorting mechanism with larger pieces of everyday trash including but not limited to apples, filled water bottles, and sandwiches.
      - While testing ensure that the drop angle and tray design facilitate an easy drop of the trash.

5) Pass/Fail Criteria
   Pass:
   - Sorting mechanism successfully rotates and supports trash
   - Tray angle drops enough for trash to leave the tray

   Fail:
   - Sorting mechanism is unable to spin trash placed in the tray
   - Trash makes contact with inner components of the trash can while rotating to the correct portion of the bin
   - Trash does not leave the tray when lowered either due to tray design or coming into contact with inner components

Prototype Testing

1) Introduction
   These tests will check the complete functionality of the prototype. An object should be detected when placed in the tray, identified, moved into the corresponding compartment, and an LED should turn on if a compartment is full. Passing this test signals the completion of the first prototype.

2) Features To Be Tested
   - Object detection
   - Image processing
   - Sorting mechanism
   - Trash level detection

3) Approach
   Run the startSimpleSort.sh script to start the image recognition and sorting mechanism control programs. Place an item in the tray and observe the terminal output as well as the trash can itself. Continue placing items and logging the behavior of the object detection and recognition.

4) Pass/Fail Criteria
   Pass:

- Object is detected when placed in the tray
- Object is identified (80% accuracy overall)
- Object is moved into a compartment
- Full compartment causes LED to turn on

Fail:

- Object is not detected when placed in the tray
- Object identification accuracy is less than 80% overall
- Object fails to be moved into a compartment
- LED fails to turn on for a full compartment

B. Test Results

Trash Level Detection: Single Sensor Test

Sensor Distance Measurements	
Distance (cm)	Measured Distance Sensor 1 (cm)
75	74-75
60	59
45	46-47
30	30-31
15	16-17
10	8-9
50	49
80	81-82
20	19-20

Fig. 40: Single Ultrasonic Sensor Distance Results

From the results above, the single ultrasonic sensor circuit and software passed the testing. All distances reported were within 3cm of the true distance from the object to the sensor. The discrepancies in the distances could have been caused from placing the object not exactly at the measured distance as 1cm is fairly small. However, the reported measurements demonstrate the circuit and the software is working appropriately. The following image is of the testing circuit.

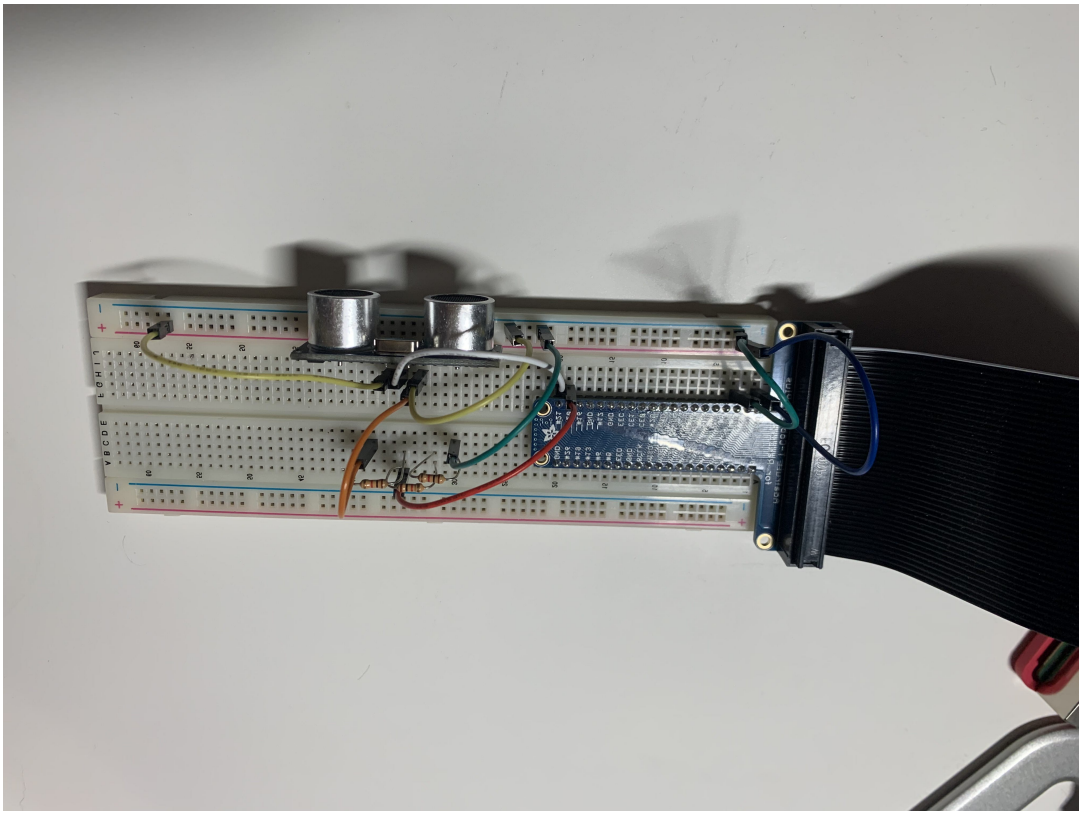


Fig. 41: Single Ultrasonic Sensor Circuit

Trash Level Detection: Three Sensor Test

Sensor Distance Measurements			
Distance (cm)	Measured Distance Sensor 1 (cm)	Measured Distance Sensor 2 (cm)	Measured Distance Sensor 3 (cm)
75	72-73	73	73-74
60	57-58	57-58	59-60
45	43-44	43	46
30	31-32	28-29	31
15	16-17	15	15

LED Status				
Distance (cm)	LED Status 1	LED Status 2	LED Status 3	LED Expected Status
> 15	Off	Off	Off	Off
< 15	On	On	On	On
< 15 to 30	On then Off	On then Off	On then Off	On then Off
< 15 to 60	On then Off	On then Off	On then Off	On then Off

Fig. 42: Three Ultrasonic Sensors Distance Results

From the results above, the trash level detection circuit and software passed the tests as all distances were within 3cm of the true distance. The accuracy will be improved upon once the sensors are physically mounted to the lid as the sensors in the prototype test circuit are allowed to freely move around. This meant during testing the sensor could have been 1 or 2 cm off from the baseline. Also while testing, it

was found that the program would hang if the sensors were close together. This is because one sensor would receive the echo pulses from another sensor and cause the original sensor to not read an echo pulse. However, when the sensors are properly distanced, as they will be in the lid, this issue does not occur. The following images demonstrate the testing circuit and the testing environment.

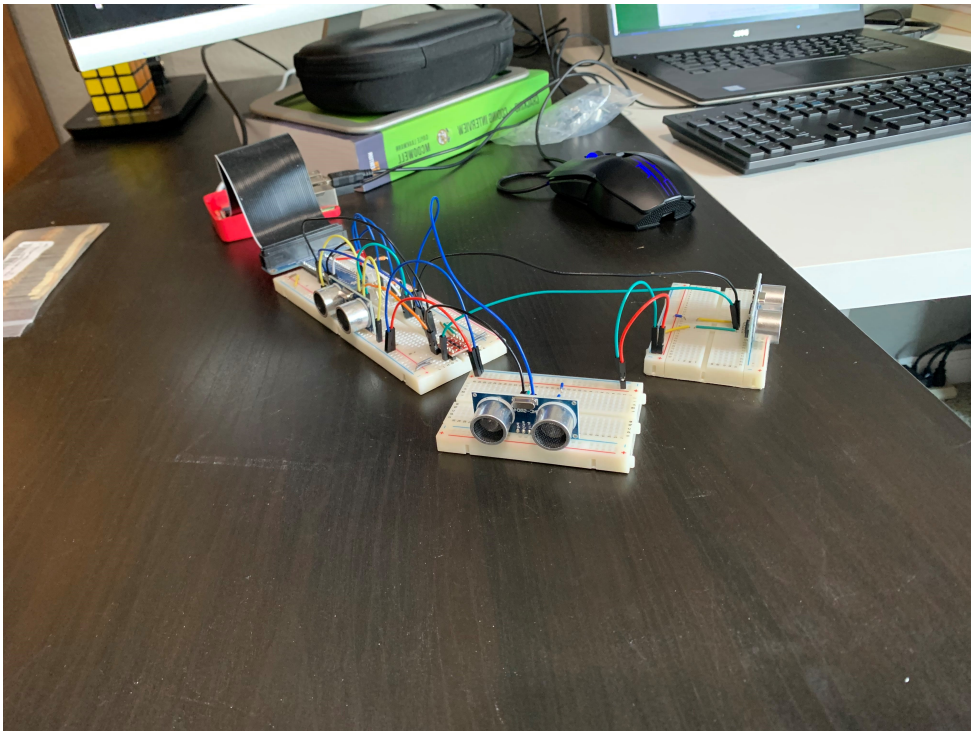


Fig. 43: Three Ultrasonic Sensor Circuit

Fig. 44: Testing Distance Layout

Trash Level Detection Installed in Lid

Sensor Distance Measurements			
Trash Level	Measured Distance Sensor 1 (cm)	Measured Distance Sensor 2 (cm)	Measured Distance Sensor 3 (cm)
Empty	52-53*	51*	51-52*
Half Full	35*	36-37*	35-36*
Full	14-15*	13-14*	13*

LED Status				
Trash Level	LED Status 1	LED Status 2	LED Status 3	LED Expected Status
Empty	Off*	Off*	Off*	Off*
Full	On*	On*	On*	On*
Full to Empty	On then Off*	On then Off*	On then Off*	On then Off*

* = Program hanged while testing

Fig. 45: Three Ultrasonic Sensor Distance In Lid Results

Based on the results above, the trash level detection circuit was able to accurately measure the distance for each compartment of the trash can. However, as indicted in the tables, the program frequently hanged while testing. In order to obtain all of the results, the program was restarted frequently. Because of this, the circuit does not past the test when installed in the trash can. The reason for why the program would hang is due to each of the sensors outputting their echo pulses in such a small area. This caused a multitude of problems such as pulses being read by another sensor and lost pulses due to the noise inside of the can. This test has highlighted that the trash level detection circuit needs to be redesigned in order to ensure the program will no longer hang.

Trash Level Detection Installed on Bottom of Sorting Tray

Sensor Distance Measurements			
Trash Level	Measured Distance Trash Compartment (cm)	Measured Distance Recycle Compartment (cm)	Measured Distance Compost Compartment (cm)
Empty	48	46	48
Half Full	31	31	32
Full	14	12	14

LED Status				
Trash Level	LED Status 1	LED Status 2	LED Status 3	LED Expected Status
Empty	Off	Off	Off	Off
Full	On	On	On	On
Full to Empty	On then Off	On then Off	On then Off	On then Off

Fig. 46: Trash Level Detection Installed on Bottom of Sorting Tray Results

Looking at the tables in Figure 46, it can be seen that the single ultrasonic sensor installed on the bottom side of the sorting tray was able to accurately measure distance. The previous issues of noise pollution from other sensors were eliminated and the program continuously ran with no issues. The LED

functionally, when a compartment was completely full, functioned as expected.

Image Classification Model Tested With Stock Photos

Image Classification Accuracy			
Image	Class Predicted	Correct Class	Confidence
banana peel	compost	compost	92.87%
twix wrapper	trash	trash	79.9%
cardboard	cardboard	cardboard	65.58%
plastic water bottle	plastic	plastic	71.3%
takeout box	plastic	trash	81.34%
apple core	compost	compost	99.93%

Fig. 47: TensorFlow Stock Photos Classification Results

Based on these results, the model does not do as well with brand new photos. In the figure below, the accuracy is very high but that is only due to the repetitive training of the same photos. When the model is given an untrained photo, its confidence is not very high despite mostly getting the classes correct. This is most likely due to unfairness in the data set. The compost class is trained with the most photos and it has the highest confidence. In order to improve the model, more photos need to be added to the other classes.

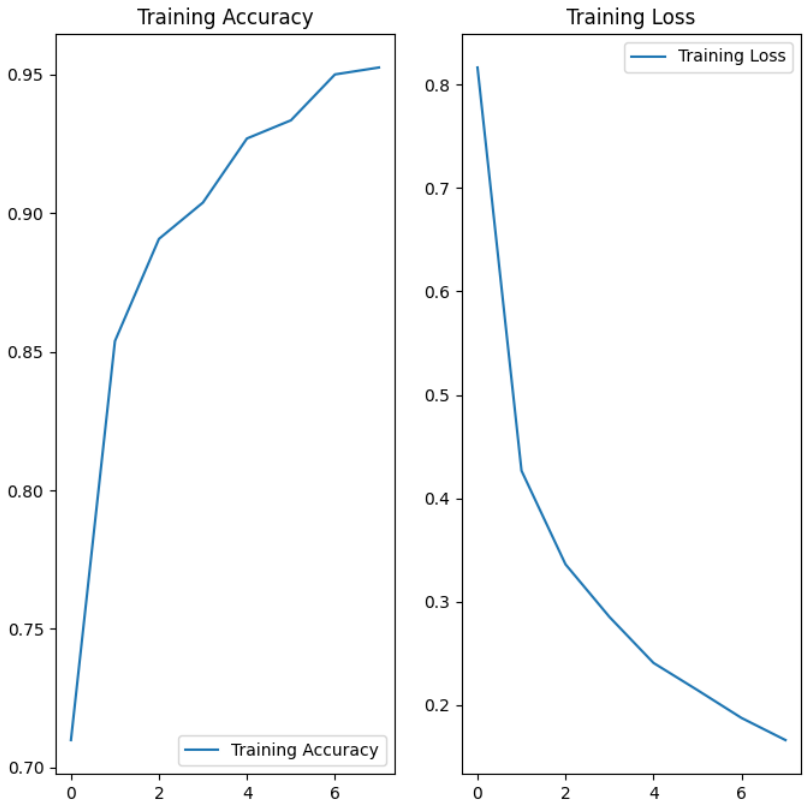


Fig. 48: Transfer Learning Training Accuracy and Loss

Image Classification TensorFlow Lite Model Tested With Stock Photos

Image Classification TensorFlow Lite Accuracy			
Image	Class Predicted	Correct Class	Tensor
banana peel	compost	compost	[-4.61, **2.95**, -0.39, -4.45, -3.45, -6.10, -1.22]
twix wrapper	trash	trash	[-2.68, -1.20, -3.22, -1.78, -1.49, -4.98, **-0.11**]
cardboard	cardboard	cardboard	[**0.58**, -0.68, -3.92, -4.30, -1.68, -2.73, -3.25]
plastic water bottle	plastic	plastic	[-4.82, -5.04, -4.42, -6.06, -5.83, **0.34**, -2.21]
takeout box	plastic	trash	[-2.93, -3.11, -2.22, -2.74, -0.35, **0.38**, -2.43]
apple core	compost	compost	[-8.13, **4.71**, -6.79, -10.37, -9.00, -7.63, -4.14]

Fig. 49: TensorFlow Lite Stock Photos Classification Tensor Results

The numbers in the tensor represent each of the classes in the following order: cardboard, compost, glass, metal, paper, plastic, and trash. Looking at the scores for each image, it's obvious that some classes have a higher score than others. The banana peel and the apple core have high scores compared to the other predicted classes. This is probably due to the number of compost images in the data set. By examining these numbers, we are also able to see what other classes come close to the correct class when the model is predicting. Using this information, we can improve our data set in order to obtain a higher accuracy and score confidence.

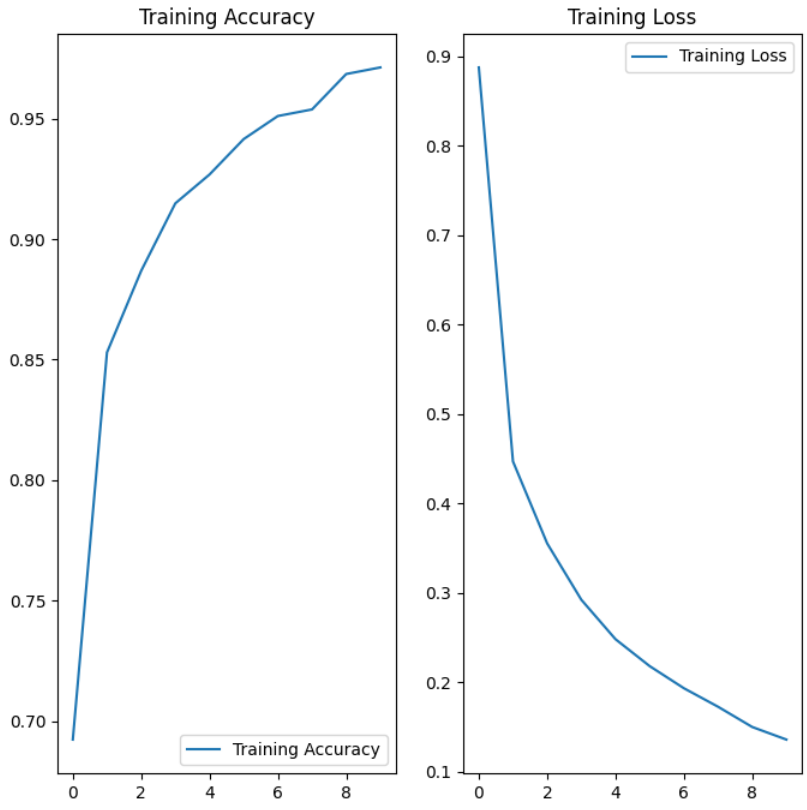


Fig. 50: Transfer Learning Training Accuracy and Loss - TensorFlow Lite Model

Image Classification Model Tested on Raspberry Pi

Image Classification Accuracy			
Image	Class Predicted	Correct Class	Confidence
apple	compost	compost	52.46%
plastic bottle	recycling	recycling	60.20%
granola wrapper	trash	trash	92.81%
serum packaging box	recycling	recycling	48.68%
tissue	compost	trash	48.19%
banana	compost	compost	99.32%

Fig. 51: TensorFlow Classification and Processing Accuracy Results

This model was able to predict most of the classes correctly. However, the percent confidence for each item varies greatly. This can be improved by introducing more items into our data set. The main drawback of this model is the time it takes to classify each image. The speed results can be seen in Figure 53. For each item, the program took almost 2 minutes to load the model and then classify it. Based on these results, we need to introduce an improved model that has good performance on smaller processors.

Image Classification TensorFlow Lite Model Tested on Raspberry Pi

Image Classification TensorFlow Lite Accuracy			
Image	Class Predicted	Correct Class	Tensor
apple	compost	compost	[-2.39, **-0.06**, -3.05, -3.43, -6.67, -2.4, -2.38]
plastic bottle	plastic	plastic	[-5.67, -7.21, 1.18, -2.82, -4, **3.35**, -2.41]
granola wrapper	trash	trash	[-4.57, -5, -4.72, -2.86, -5.65, -2.8, **2.07**]
black soda can	glass	metal	[-7.16, -3.59, **0.37**, 0.016, -4.55, -2.82, -5.44]
tan napkin	cardboard	trash	[**-1.62**, -2.09, -6.86, -7.35, -4.23, -6.07, -1.76]
french fries	compost	compost	[-5.69, **3.21**, -8.49, -8.05, -2.77, -9.66, -2.27]

Fig. 52: TensorFlow Lite Classification and Processing Tensor Results

In terms of getting the correct overall category (landfill, recycling, and compost), the model performs pretty well. As seen in the table above, the model mixes up some of the recycling categories but still gets it in the overall recycling category. The model also has a problem with a few items depending on the color such as the tan napkin. Since it is a similar color to cardboard, the napkin was sorted into the cardboard category. These issues can probably be solved by increasing the size of our data set and feeding more photos into our model.

Compared to the previous model, this model classifies images instantaneously. While the camera takes a few seconds to start up, this is not a problem in the prototype because the camera is already running. Therefore, the classification time test passes.

Image Classification Speed Test

	TensorFlow	TFlite
Time 1	1m41.993s	0m14.728s
Time 2	1m41.857s	0m14.862s
Time 3	1m40.759s	0m14.765s
Average	1m41.536s	0m14.785s

Fig. 53: Speed Test Results

From the speed test it can be seen that the TFlite (TensorFlow Lite) model was a dramatic improvement over the TensorFlow model. It should be noted that the total real time includes the time to initialize the camera and and import all the required libraries. In the actual loop that runs in the prototype the classification with the TFlite model is less than a second.

Image Processing GPIO

Image Processing Output	LED color
standby	purple
standby	purple
standby	purple
standby	purple
compost	red
compost	red
compost	red
compost	red
recycle	blue
recycle	blue
recycle	blue
recycle	blue
trash	off (dim)
trash	off (dim)
trash	off (dim)
trash	off (dim)
TOTAL SUCCESS	100 %

Fig. 54: Image processing GPIO results

This test was a complete success. It can therefore reliably be assumed that the correct corresponding pin output is set depending on the results of the image classification. This assurance verifies that the trash sorting accuracy with the prototype is dependent on the model and image processing instead of errors with the pin output.

Motion Detection

version	test	pass/fail
startV1	no motion 20 seconds	fail
startV1	no motion 20 seconds	fail
startV1	no motion 20 seconds	fail
startV1	object placed in front of camera	fail
startV1	object placed in front of camera	pass
startV1	object placed in front of camera	fail
startV2	no motion 20 seconds	pass
startV2	no motion 20 seconds	fail
startV2	no motion 20 seconds	fail
startV2	object placed in front of camera	fail
startV2	object placed in front of camera	pass
startV2	object placed in front of camera	pass
startV4	no motion 20 seconds	pass
startV4	no motion 20 seconds	pass
startV4	no motion 20 seconds	pass
startV4	object placed in front of camera	pass
startV4	object placed in front of camera	pass
startV4	object placed in front of camera	fail
startV6	no motion 20 seconds	pass
startV6	no motion 20 seconds	pass
startV6	no motion 20 seconds	pass
startV6	object placed in front of sensor	pass
startV6	object placed in front of sensor	pass
startV6	object placed in front of sensor	pass

Fig. 55: Motion Detection results

The method of using an ultrasonic sensor is much more reliable than using the camera, as long as the object is placed directly in front of the sensor. startV3 and startV5 were not used because they included no new updates to the motion detection software.

Servo Functionality Test 1

Servo Functionality		
Servo	Rotation Angle	Delay (s)
1	270°	3
2	270°	3
3	270°	3

Fig. 56: Servo Functionality Test Results

Figure 56 shows the results of the first servo functionality test, which tested to see if the servo motors were capable of rotating forwards and backwards a full 270 degrees. We also added a 3 second delay between rotations to simulate the dropping of the tray. Each servo rotated the full 270 degrees and paused for the correct amount of time.

Servo Functionality Test 2

Top Servo Test		
Input	Rotation Angle	Delay (s)
Trash	0°	5
Recycling	120°	5
Compost	-120°	5

Fig. 57: Top Servo Test Results

Figure 57 shows the results of the top servo portion of the second servo functionality test. This test checked our intended top servo functionality where when provided a trash input the servo does not rotate, when provided a recycling input the servo rotates 120 degrees, and when provided a compost input the servo rotates -120 degrees. The servo rotated the correct amount for each input and delayed for 5 seconds before rotating back to the trash position as intended.

Combined Servo Test		
Servo	Pass/Fail	Delay (s)
Top Servo	Pass: Rotated 0°, 120° or, -120°	5
Arm Servo	Pass: Rotated 100°	3
Arm Servo	Pass: Rotated 100°	3

Fig. 58: Combined Servo Test Results

Figure 58 shows the results of the combined servo portion of the second servo functionality test. This test ensured our top servo remained functional when the arm servos were also connected. When provided a trash input the top servo does not rotate, when provided a recycling input the servo rotates 120 degrees, and when provided a compost input the servo rotates -120 degrees. During the top servo's 5 second delay, the arm servos rotated 100 degrees and delayed for 3 seconds before rotating back in place. Each of the servos passed the test.

These servo tests verified the functionality of our servos and the code we used to operate them. The code used in these tests was hard coded to provide a proper input, isolating the servo functionality for testing. An image of the test set up for the servos is shown in Figure 59.

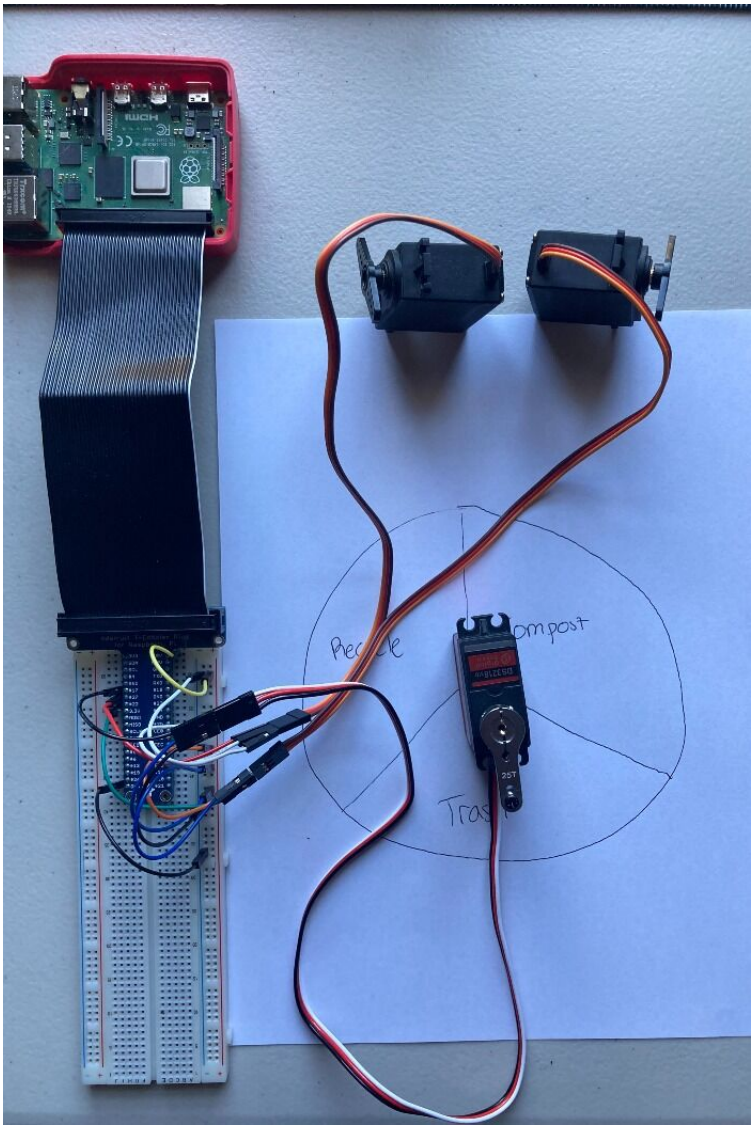


Fig. 59: This image shows the setup used to test the functionality of the servos. The two arm servos are shown at the top and the top rotating servo is shown in the middle. We used hard coded test files to run the servo test.

Sorting Mechanism Functionality Test 1

Sorting Mechanism Rod Test		
Input	Surface	Pass/Fail
Compost	Tile	Pass
Compost	Carpet	Fail
Compost	Inside of Trash Can	Pass

Fig. 60: Sorting Mechanism Functionality Test 1 Results

Figure 60 shows the results of the rod portion of the sorting mechanism functionality test. This test ensured our top servo was able to rotate the center rod the corresponding distances when provided a known input. When provided a trash input the rod does not rotate, when provided a recycling input the rod rotates 120 degrees, and when provided a compost input the rod rotates -120 degrees. The top servo

was able to successfully rotate the rod when the bottom of the rod was set on kitchen tile, but to test how friction affected the rotation of the rod we also tested with the bottom of the rod set on carpet and found the rotation distances were not always consistent. However, when placed inside the can, the surface was smooth enough for consistent rotation distances.

Sorting Mechanism Functionality Test 2

Sorting Mechanism Tray Test		
Input	Lower	Raise
Compost	Pass	Pass
Trash	Pass	Pass
Recycle	Pass	Pass

Fig. 61: Sorting Mechanism Functionality Test 2 Results

Figure 61 shows the results of the tray portion of the sorting mechanism functionality test. This test ensured our arm servos were able to raise and lower the tray. The top servo was able to successfully rotate the rod. When provided each of the possible inputs the servos were able to raise and lower the empty tray and pass each of the tests.

Sorting Mechanism Functionality Test 3

Sorting Mechanism Combined Test			
Input	Rotate	Raise/Lower	Timing
Compost	Pass	Pass	Pass
Recycle	Pass	Pass	Pass
Trash	Pass	Pass	Pass

Fig. 62: Sorting Mechanism Functionality Test 3 Results

Figure 62 shows the results of the combined sorting mechanism functionality test. This test ensured our top servo was able to rotate the center rod the corresponding distances when provided a known input, and before the center rod rotated back that the arm servos were able to raise and lower the tray. For each of the inputs we found that the sorting mechanism was rotated the correct distance by the top servo, and the tray was successfully raised and lowered. The timing of each servo rotation was also successful with the tray being raised and lowered before the rod was rotated back to its resting position.

Sorting Trash Test 1

Sorting Small Items			
Item Placed	Rotate	Raise/Lower	Timing
Sponge	Pass	Pass	Pass
Wrapper	Pass	Pass	Pass
Cherry	Pass	Pass	Pass

Fig. 63: Sorting Trash Test 1 Results

Figure 63 shows the results of testing the full sorting mechanism functionality with small items that are commonly thrown away in kitchens. We found that the sorting mechanism was able to properly sort these items when manually fed the proper input category. These were all relatively small items so we also tested larger items, which is shown below in Figure 64.

Sorting Trash Test 2

Sorting Heavy Items			
Item Placed	Rotate	Raise/Lower	Timing
Apple	Pass	Pass	Pass
Filled Water Bottle	Pass	Pass	Pass
Canned Beans	Pass	Pass	Pass

Fig. 64: Sorting Trash Test 2 Results

Figure 64 shows the results of testing the full sorting mechanism functionality with larger items that are commonly thrown away in kitchens. We wanted to push the limits of the sorting mechanism tray to see how much weight our prototypes could handle. We found that the sorting mechanism was able to properly sort an apple, a filled water bottle, and a full can of beans. When testing these items, the proper input was manually input to the system. Considering our prototype was made with inexpensive hardware store parts, we are confident if we upgraded the durability of the parts used we would be able to sort even heavier items.

Prototype

Initial test results are shown in figure 65. More tests, with the tensor values included, can be found in figure 68. With over 75 tests, the prototype maintains an identification accuracy greater than 80%. The object detection and sorting mechanism each remain near flawless in detecting items placed in the tray and moving items into a compartment, respectively. The LEDs also operate as expected, turning on when a compartment becomes full. All test criteria pass, demonstrating the successful operation of the prototype.

Prototype Testing			
Object	Correct Class	Predicted Class	Operation
orange peel	compost	compost	success
apple	compost	compost	success
black soda can	recycling	recycling	success
plain cardboard	recycling	recycling	success
candy wrapper	landfill	landfill	success
ketchup packet	landfill	landfill	success

Fig. 65: Initial prototype testing results

X. Appendix 4 - Previous Designs

A. Circuits

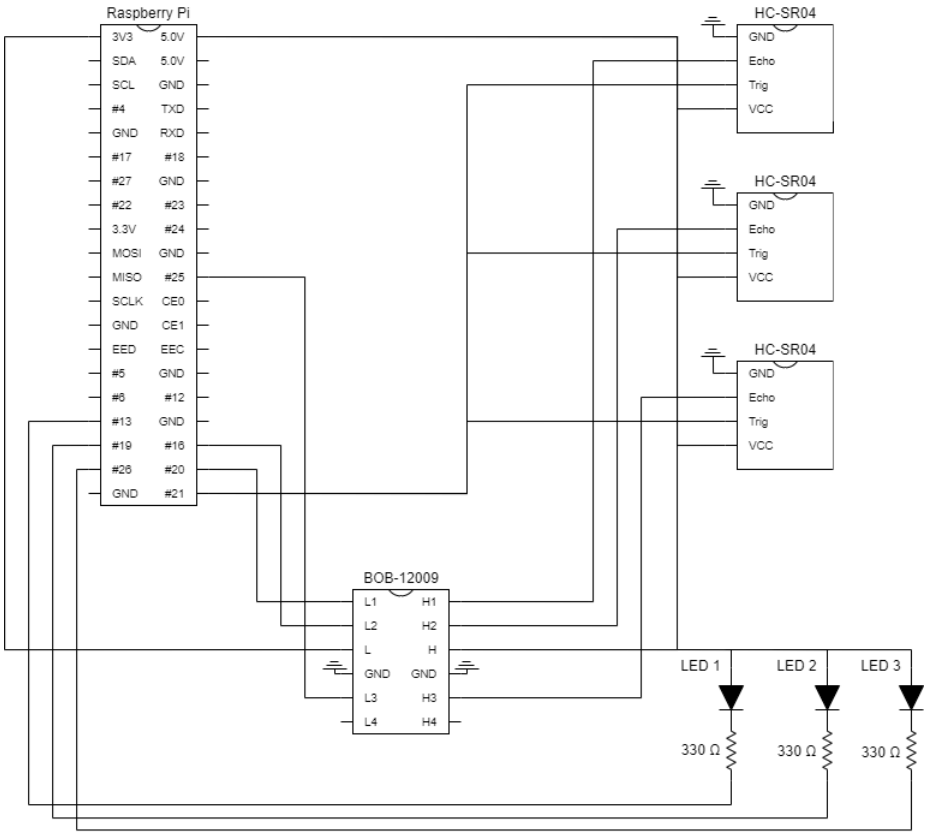


Fig. 66: Trash Level Detection Circuit With 3 Distance Sensors. Created using Diagrams [7].

Figure 66 details the circuit for the trash level detection when it was envisioned to use three ultrasonic sensors. This circuit is similar to the current design, seen in Figure 9, but with two extra ultrasonic sensors. As described in the test results section, this circuit was not able to accurately run within the physical can due to large amounts of noise interference between the three sensors.

B. Software

1) startV1 - startV6:

startV1.py was the first iteration of our code to detect trash, take an image, run it though the

TensorFlow image classification model, and set the appropriate output pins for the servo code to sample. It uses the camera to detect motion and has a threshold of one changed frame with at least a 50% difference. This motion detection was too sensitive and did not work reliably. The GPIOs are set using PIGPIO. The image classification was also very slow with the TensorFlow model.

startV2.py adjusted the motion detection so that it has a threshold of 3 changed frames in a row with at least 60% difference. This was a good improvement, but still unreliable.

startV3.py changed the GPIO library from PIGPIO to GPIOZERO because the PIGPIO daemon only allows one program to use it at a time and this clashed with the servo code.

startV4.py replaced the TensorFlow model with the TFLite model and this resulted in a dramatic improvement in speed. This version also adjusted the motion detection so that the threshold is 4 changed frames with at least a 50% difference. This led to the greatest improvement to motion detection, but it was still prone to errors.

startV5.py changed the motion detection from using the camera to using a distance sensor to measure the distance across the tray. If the distance is less than 15 cm then an item is detected inside the tray.

startV6.py uses an updated TFlite model with some more images added to the training set. This led to an improvement on sorting accuracy, especially for trash, which was underperforming.

C. Image Classification Models

1) Creating A Model From Scratch: Before we decided to implement transfer learning for our model, we created a model from scratch. Due to having a fairly small data set, we were not able to achieve satisfactory results from this model. The results from training this model can be seen in Figure 67.

2) Transfer Learning With a TensorFlow Model: As seen in the Test Plans and Results section, we also created a regular TensorFlow model. While this was successful when tested on a computer or laptop, it proved to be very slow on the Raspberry Pi. This was due to the large size of the model. Since the Raspberry Pi has a smaller processor, it took over a minute to load the model and then classify the image. The results for this can be seen in Figure 48. We eventually converted our model to a TensorFlow Lite model for our final prototype which resulted in a great performance improvement.

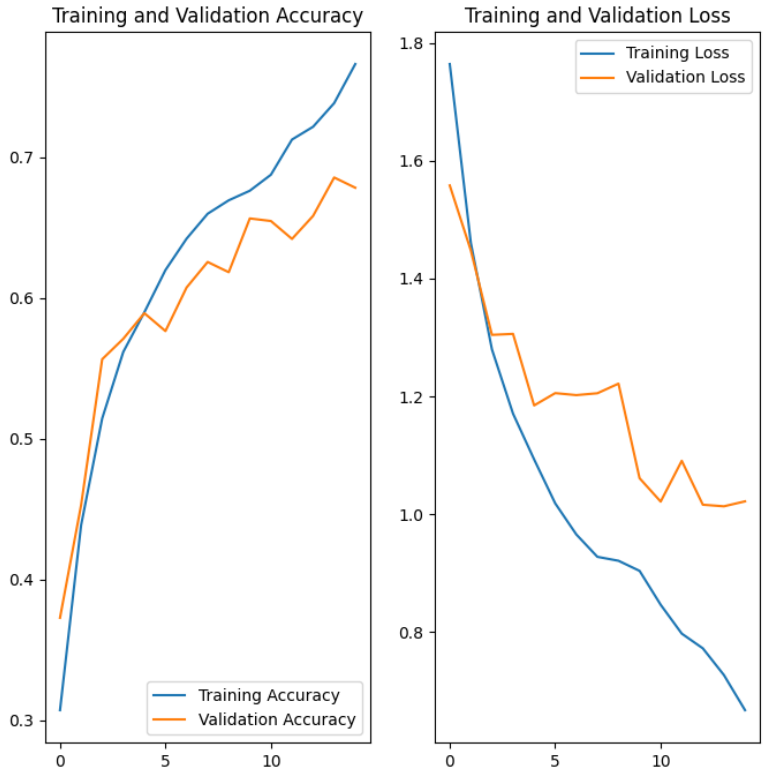


Fig. 67: Training and Validation Accuracy/Loss With Model Made From Scratch

XI. Appendix 5 - Reviews

A. *Justyn Duthler*

Reflecting back on the entire process for this project, there are many things that went well. As a team we we were able to effectively work together with no issues throughout the two quarters. Our communication was clear, everyone provided assistance to each other when needed, everyone met deadlines we set for ourselves, and we were able to effectively divide the tasks for the project with each person's skill set in mind. Another aspect of our process that went well was our focus on making the prototype work effectively. When an implementation of a certain part of the design proved to not meet our goals, such as the original image classification neural network or the motion detection software, we would come together as a team to develop a new design to test. However, if I were to do this project again there are many things that I would change. While our effort to make the prototype work effectively proved to be a great asset, it took a lot of the time away from creating a solid design. If we were to do this project again, I would allocate more time in developing the design. Such as designing custom PCBs, gathering a larger data set for our image classification and training the neural network with said larger data set, and spending more time researching alternative methods to meet our goals. Our design is comprised of hardware that we are familiar with through our courses but I believe we could come up with a more efficient and accurate design by expanding our knowledge with new hardware. Overall, we met a lot of our initial goals for this project and provided a design that proves our concept works. With further research and development, I believe SimpleSort has the potential to become a product with wide use.

B. Kevin Ryan

Overall, this project went very well. I feel like we worked really well as a team with everyone contributing to the project. We met all our goals of having a functional prototype with a sorting accuracy of over 80%. In my opinion the biggest drawback to this project was having to do everything remote. This did not provide too much of an issue for the software aspects of the project since we could easily collaborate through github. However, I think this created issues with the hardware and prototype construction. I think it would have been more effective if classes were in person and we could have one prototype stored in a campus lab room that we could all meet together to work on in person. Instead, we had to worry about shipping six identical copies of each part so that we could all build a prototype or test specific parts. I think this cost more time and resources than it would have if we had a singular prototype we were all in close proximity to. This would have also allowed us to spend 6 times as much money on this prototype that would have made it easier for us to increase the size/scalability as well as spend time and money on creating custom PCBs for the prototype. Despite the issues involved with working on a capstone during a global pandemic, I am very pleased with the end result of our project and I think it has solid potential to become a marketable product.

C. Shirley Phuong

These past two quarters went very well considering we had to work remotely the entire time. Despite doing everything online, we were able to create a working physical prototype with a satisfactory sorting accuracy. In addition to this, everyone in the group put in the time and effort to make everything go smoothly. Going into this, I did not have high expectations for the image classification portion of our project; none of us had any experience working with it before. However, my expectations were exceeded and we were able to create a successful model for our product. This part of the project is definitely something I felt went very well. If I were given the chance to redo this project, I would prefer to do it in person with the rest of the group. I would also like to spend extra time on the can design since we were mostly focused on functionality. Overall, I think our group did a great job with the time we were given and we were able to create a possibly marketable product that is beneficial for the environment.

D. Trevor Carleton

Looking back at this project it was very successful. Going into winter quarter we faced challenges that we and this class hadn't experienced before. We had to work online with people to complete a complex engineering project. This didn't create a significant issue when creating software but it was a problem when we began building the physical design. We all had to get our own parts shipped out and this took a significant amount of time. Once we were able to get the parts we made quick progress. We were able to surpass our 80% accuracy in trash sorting and our sorting mechanism could consistently place items where the software intended. If we were to do the project again I would prefer to do it in person, as it would be much easier to communicate and to create the physical design. I would also like to use upgraded hardware to give the trash can more strength and a better look. Our budget restricted us from getting some of these upgraded parts. Overall, the process and final design went very well.

E. Kevin Spence

I believe that this project was very successful. We accomplished our main goal of having a prototype that could automatically sort items placed in the bin with greater than 80% accuracy. The sorting mechanism we designed was always consistent, and the accuracy of the image classification system exceeded our original goals, despite shipping delays for parts and everyone in the team having to work remotely. I think as a team we did a good job dividing labor, and meeting deadlines. If given more time I think the areas we would look to improve on the project would be the image classification accuracy, because the more accurate the better. Another area where we could improve would be getting parts that were more tailored to meet our specific needs such as a wide angle camera, rather than the standard Raspberry Pi camera. Ultimately I enjoyed working with each of my team members, and I am proud that we were able to provide a proof of concept with a consistent functional prototype.

F. Brevan Chun

Perhaps the most ambitious yet successful aspect of this project was the image recognition. It slightly exceeded expectations in terms of classification accuracy and greatly exceeded expectations in terms of runtime. Perhaps one of the most significant weaknesses in the design of our prototype is its relatively compact size. A compact design allowed us to keep parts small, cheap, and manageable but the resulting sorting mechanism was constrained to smaller items. Additionally, it meant the camera mounting location had to be closer than ideal. A design accommodating for larger items could have allowed us to mount the image recognition camera in a more optimal position, giving us clearer pictures that would more closely match our training data set and result in even better item classification accuracy. In regards to the project as a whole, we met or exceeded nearly all of our initial goals and we worked as effectively as a team throughout the process - an excellent experience overall.

XII. Appendix 6 - Image Classification Accuracy Spreadsheet

0	Item	Predicted Class	Actual Class	Match?	Cardboard	Compost	Glass	Metal	Paper	Plastic	Landfill
1	apple	compost	compost	TRUE	-2.39	**-0.06**	-3.05	-3.43	-6.67	-2.4	-2.38
2	plastic water bottle	recycling	recycling	TRUE	-5.67	-7.21	1.18	-2.82	-4	**3.35**	-2.41
3	granola bar wrapper	trash	trash	TRUE	-4.57	-5	-4.72	-2.86	-5.65	-2.8	**2.07**
4	black soda can	recycling	recycling	TRUE	-7.16	-3.59	**0.37**	0.016	-4.55	-2.82	-5.44
5	gorilla glue bottle	recycling	recycling	TRUE	-7.74	-1.48	**-0.16**	-2.51	-1.89	-3.4	-3.31
6	orange peel	compost	compost	TRUE	-8.06	**5.11**	-6.28	-4.61	-4.68	-6.01	-1
7	square chocolate wrapper	trash	trash	TRUE	-3.89	-0.99	-5.87	-2.52	-0.03	-3.64	**0.39**
8	plastic wrapper	recycling	recycling	TRUE	-2.37	-3.33	-1.44	-1.49	**0.28**	-3.17	-1.65
9	crumpled yellow post-it	recycling	compost	FALSE	-5.01	**2.48**	-5.08	-5.19	-2.77	-4.27	-1.28
10	1oz. blue doritos bag	trash	trash	TRUE	-2.46	-0.97	-8.11	-1.24	-1.74	-6.43	**-0.53**
11	pink aluminum can	recycling	recycling	TRUE	-5.81	-7.16	-3.1	**2.47**	-7.39	-3.16	-1.52
12	small tomatoes	compost	compost	TRUE	-2.19	**0.19**	-6.31	-5.06	-6.73	-2.33	-2.14
13	bitten apple	compost	compost	TRUE	-4.39	**1.65**	-5.44	-7.38	-4.84	-1.25	-1.65
14	bar wrapper	trash	trash	TRUE	-4.21	-3.32	-7.71	-4.37	-2.52	-1.46	**0.82**
15	tea wrapper	trash	trash	TRUE	-3.73	-6.21	-3.84	-5.2	-1.31	0.06	**0.98**
16	crumpled plastic bottle	recycling	recycling	TRUE	-5.94	-5.36	-2.71	-7.06	-4.95	**-0.65**	-1.73
17	onion quarter	compost	recycling	FALSE	-4.23	-1.93	-2.34	-7.23	**0.93**	-3.12	-1.22
18	receipt (flat/folded)	recycling	recycling	TRUE	-3.43	-1.39	-3.72	-3.55	**2.19**	-2.18	-4.34
19	receipt (crumpled)	compost	recycling	FALSE	-10.57	**1.13**	-4.28	-4.22	-3.63	-4.97	-2.2
20	glass cup	compost	recycling	FALSE	-6.44	**-0.14**	-2.34	-4.25	-6.5	-4.08	-3.6
21	banana	compost	compost	TRUE	-8.11	**-1.56**	-4.8	-3.24	-9.45	-3.68	-7.1
22	kitkat wrapper	trash	trash	TRUE	-3.88	-6.08	-7.1	-1.22	-3.2	-3.06	**1.71**
23	tortilla chip	compost	compost	TRUE	-10.16	**2.67**	-5.3	-5.69	-5.54	-7.41	-1.66
24	crumpled blue tape	trash	trash	TRUE	-8.63	0.75	-7.44	-5.22	-3.44	-8.26	**0.99**
25	ketchup packet	trash	trash	TRUE	-5.84	-1.92	-5.57	-3.73	-3.06	-4.26	**-0.55**
26	tissue	compost	trash	FALSE	-13.05	**-0.56**	-2.92	-6.25	-4.44	-3.31	-1.07
27	red napkin	compost	trash	FALSE	-2.1	**1.46**	-5.75	-6.66	-3.61	-7.03	-1.6
28	tan napkin	recycling	trash	FALSE	**-1.62**	-2.09	-6.86	-7.35	-4.23	-6.07	-1.76
29	raspberry pi box	recycling	recycling	TRUE	-0.54	-6.71	-7.17	-6.82	**0.69**	-3.59	-2.5
30	corn	compost	compost	TRUE	-5.53	**2.11**	-8.52	-6.42	-4.74	-4.59	0.48
31	apple peel	compost	compost	TRUE	-7.63	**5.72**	-9.07	-9.18	-6.9	-9.91	-5.06
32	blue plastic bottle	recycling	recycling	TRUE	-3.26	-3.62	-3.12	-2.45	**-2.39**	-4.07	-6.43
33	apple slice	compost	compost	TRUE	-9.77	**2.21**	-7.12	-6.44	-5.22	-7.16	-2.43
34	french fries	compost	compost	TRUE	-5.69	**3.21**	-8.49	-8.05	-2.77	-9.66	-2.27
35	id card	recycling	recycling	TRUE	-5.36	-3.55	-4.99	-1.59	-1.78	**-1.04**	-1.5
36	clear plastic wrapper	recycling	recycling	TRUE	-12.54	-1.93	-4.11	-2.93	-7.28	**0.07**	-2.65
37	index card	recycling	recycling	TRUE	-3.32	-3.8	-6.86	-4.59	**0.78**	-5.72	-2.2
38	blue doritos bag (back)	trash	trash	TRUE	-4.71	-4.2	-11.11	-5.01	-4.09	-4.25	**-1.4**
39	toothbrush	trash	trash	TRUE	-5.44	-1.74	-4.41	-2.92	-6.13	-6.81	**-0.21**
40	lettuce leaf	compost	compost	TRUE	-3.34	**0.211**	-3.35	-4.39	-6.27	-9.02	-1.68
41	spam can	recycling	recycling	TRUE	-4.2	-2.33	-8.09	**-1.12**	-4.22	-2.82	-2.29
42	small toothpaste	trash	trash	TRUE	-6.82	-2.99	-1.44	-2.97	-2.67	-3.18	**-0.9**
43	chocolate wrapper	trash	trash	TRUE	-10.98	-1.31	-4.77	-2.54	-7.41	-3.74	**-0.82**
44	latex glove	trash	trash	TRUE	-3.64	-1.15	-7.51	-6.56	-3.23	-10.75	**-0.77**
45	plastic bag	trash	trash	TRUE	-6.54	-0.74	-1.63	-6.31	-3.47	-4.24	**1.32**
46	sauce packet	trash	trash	TRUE	-6.5	-2.62	-3.15	-2.4	-4.77	-5.1	**-1.8**
47	red netting	compost	trash	FALSE	-4.34	**1.12**	-6.37	-5.92	-7.47	-8.57	-4.41
48	lemon	compost	compost	TRUE	-7.76	**1.69**	-6.64	-5.84	-6.54	-5.26	-1.96
49	pink flower	compost	compost	TRUE	-5.65	**0.79**	-5.64	-4.37	-4.52	-8.67	-3.16
50	purple flower	compost	compost	TRUE	-8.1	**2.31**	-8.57	-5.77	-5.18	-6.91	-4.43
51	bouquet of flowers	compost	compost	TRUE	-5.38	**2.11**	-8.7	-6.28	-5.85	-10.28	-3.59
52	envelope	recycling	recycling	TRUE	-2.17	-4.76	-4.12	-3.22	**2.47**	-1.28	-1.99
53	mini pumpkin	compost	compost	TRUE	-6.42	**1.57**	-4.07	-5.1	-6.88	-2.49	-1.7
54	cardboard packaging	recycling	recycling	TRUE	-0.86	-3.99	-4.49	-1.91	**0.93**	-2.17	-2.45
55	orange plastic	recycling	recycling	TRUE	-6.18	-2.85	-2.16	-2.74	**-0.67**	-3.2	-2.21
56	plain cardboard	recycling	recycling	TRUE	**1.72**	-3	-5.68	-5.22	-0.87	-5.61	-1.33
57	black zipties	recycling	recycling	TRUE	-6.75	-1.18	-3.91	**1.73**	-3.75	-7.82	-5.44
58	Kirkland Signature water bottle	recycling	recycling	TRUE	**-1.46**	-3.08	-7.24	-6.46	-5.33	-3.25	-2.69
59	cardboard sleeve	recycling	compost	FALSE	-4.49	**0.08**	-1.49	-1.61	-2.15	-2.85	-4.72
60	wine bottle	recycling	recycling	TRUE	-1.62	-0.63	**1.06**	-3.84	-2.84	-5.42	-2.8
61	used tissues	compost	compost	TRUE	-2.17	**1.59**	-10.64	-8.47	-2.59	-6.81	-0.86
62	surgical mask	trash	recycling	FALSE	**1.76**	-1.18	-7.98	-4.93	1.74	-8.52	-4.52
63	lemon wedge	compost	compost	TRUE	-0.902	**1.95**	-5.58	-5.15	-7.17	-2.84	-2.94
64	nectarine pit	compost	compost	TRUE	-8.47	**5.68**	-8.98	-8.69	-8.62	-5.06	-3.64
65	chapstick tube	trash	trash	TRUE	-2.597	-2.349	-3.153	-4.418	-2.053	-1.982	**-0.706**
66	chamois butter tube	recycling	trash	FALSE	-3.230	-5.613	**-1.750**	-6.704	-2.491	-3.090	-3.104
67	lottery ticket	recycling	recycling	TRUE	-2.071	-3.329	-7.793	-4.863	**0.540**	-0.991	0.121
68	pretzel chip	compost	compost	TRUE	-5.785	**3.429**	-13.457	-8.110	-6.893	-7.727	0.530
69	empty hummus container	recycling	trash	FALSE	-7.794	-0.800	-0.799	-3.915	-9.380	**0.725**	-1.834
70	pen	recycling	recycling	TRUE	**-0.474**	-3.450	-2.413	-5.869	-3.244	-6.320	-0.966
71	hand santizer bottle	trash	trash	TRUE	-4.182	-3.651	-3.072	-3.868	-3.588	-0.276	**0.586**
72	ear buds	compost	recycling	FALSE	-1.140	**0.197**	-6.857	-6.182	-2.639	-6.644	-3.082
73	sunglasses	recycling	recycling	TRUE	-3.778	-2.193	**-1.901**	-3.164	-2.433	-6.442	**-1.901**
74	Cliff Gel wrapper	trash	trash	TRUE	-5.234	-2.643	-3.877	-5.196	-3.466	-4.246	**-0.920**
75	baby carrots	compost	compost	TRUE	-2.876	**3.554**	-6.443	-3.325	-3.234	-5.403	-1.544
76	serum packaging box	recycling	recycling	TRUE	-1.64	-4.97	-3.33	-3.23	**-0.85**	-1.61	-2.68

Class	# of items	correct matches	accuracy
Overall (76)	76	63	82.89%
Compost (23)	23	21	91.30%
Recycling (29)	29	24	82.76%
Trash (24)	24	18	75.00%

Fig. 68: Image Classification Accuracy Spreadsheet Results

REFERENCES

[1] 2021. URL: https://www.recycleaway.com.

[2] 2021. URL: https://www.trashcanswarehouse.com.

[3] 2021. URL: https://www.kicad.org/.

[4] *America Recycles Day*. 2019. URL: https://www.epa.gov/americarecycles/america-recycles-day (visited on 05/27/2021).

[5] AutoDesk. *Fusion 360*. 2021. URL: https://www.autodesk.com/products/fusion-360/.

[6] *Bin-e*. URL: https://www.bine.world/ (visited on 06/02/2021).

[7] *Diagram Software and Flowchart Maker*. URL: https://app.diagrams.net (visited on 05/15/2021).

[8] Justyn Duthler et al. *Code Repository*. 2021. URL: https://github.com/JustynDuthler/SimpleSort.

[9] Sarah Frost. *CompostNet*. 2019. URL: https://github.com/sarahmfrost/compostnet.

[10] *Raspberry Pi 4*. 2021. URL: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[11] *RASPBERRY PI CAMERA MODULE*. 2021. URL: https://www.digikey.com/en/products/detail/raspberry-pi/913-2664/6152810.

[12] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

[13] *SER0038*. 2021. URL: https://www.digikey.com/en/products/detail/dfrobot/SER0038/7087149/.

[14] *SparkFun Logic Level Converter - Bi-Directional*. 2021. URL: https://www.sparkfun.com/products/12009.

[15] *TrashBot*. URL: https://cleanrobotics.com/trashbot/ (visited on 06/02/2021).

[16] *Ultrasonic Distance Sensor - HC-SR04*. 2021. URL: https://www.sparkfun.com/products/15569.