

Trash-E: The Autonomous Litter Picker Upper

Group: 26

Senior Design 2 EEL 4915L

Members

Thomas Greco: Computer Engineering

Christian Mayo: Computer Engineering

Alex Rizk: Computer Engineering

Chrizzell Jay Sanchez: Electrical
Engineering

Contents

1.0 Executive Summary.....	1
2.0 Project Description.....	2
2.1 Project Background.....	2
2.2 Objectives	2
2.3 Requirements Specifications.....	4
2.4 Customers, Sponsors, Contributors	5
2.5 Marketing and Engineering Requirements	6
2.6 Product Features.....	6
3.0 Research.....	8
3.1 Existing Products	8
3.2 Microcontrollers.....	8
3.3 Computer Vision	17
3.4 Power	25
3.5 Locomotion and Mapping.....	36
4.0 Constraints	46
4.1 Description	46
4.2 Economic.....	46
4.3 Environmental.....	46
4.4 Social	46
4.5 Sustainability	46
4.6 Ethical.....	47
4.7 Time	47
4.8 Safety	47
4.9 Manufacturing	48
5.0 Standards	49
5.1 Lithium-Ion Battery Safety Standards.....	49
5.2 Standard SystemC [®] Language Reference Manual Standard.....	49
5.3 Software and Systems Engineering – Software Testing.....	50
5.4 Programming language – C Standard	51

5.5 Robot Systems – Safety Requirements Standard	52
6.0 System Design	54
6.1 Software Design	54
6.2 Hardware Design.....	76
6.3 Bill of Materials (BOM).....	113
7.0 Prototyping, Build, Test, Evaluation Plans	114
7.1 Prototyping	114
7.2 Computer Vision Testing.....	115
7.3 Hardware Testing Plans	117
7.4 Evaluation	121
7.5 Hardware Component Testing.....	125
8.0 Project Operation.....	130
8.1 Installation, Configuration, and Operation.....	130
9.0 Administrative Content.....	132
9.1 Milestones.....	132
9.2 Budget and Finance.....	133
10.0 Project Summary and Conclusion	134
11.0 Appendices.....	135
10.1 Bibliography	135

List of Figures

Figure 1: House of Quality	6
Figure 2: Example Transmission Using UART (Courtesy of SparkFun)	9
Figure 3: Benchmarks for Different Cortex-M Processors (Courtesy of ST)	11
Figure 4: Object Detection Using Computer Vision (Courtesy of TowardsDataScience.com).....	17
Figure 5: Traditional Learning (Courtesy of TowardsDataScience.com).....	18
Figure 6: Convolutional Neural Network Architecture (Courtesy of TowardsDataScience.com)..	20
Figure 7: Available Models (Courtesy of Tensorflow).....	22
Figure 8: The TensorRT Flow (Courtesy of Nvidia).....	24
Figure 9: TensorRT Optimization Performance Graph (Courtesy of Nvidia).....	24
Figure 10: Minicomputer Deep Learning Benchmarks (Courtesy of Nvidia)	25
Figure 11: Battery Charging from a DC Power Supply	31
Figure 12: Battery Pack Charging with Buck/Boost Converter	31
Figure 13: Linear Voltage Regulator (Courtesy Rohm)	34
Figure 14: Standard Voltage Regulator	34
Figure 15: Typical Switching Voltage Regulator Circuit of a LM5017	35
Figure 16: Comparison between no SLAM and SLAM (Courtesy of MathWorks)	37
Figure 17: Flow of SLAM process (Courtesy of MathWorks)	38
Figure 18: Example of pose graph optimization (Courtesy of MathWorks)	40
Figure 19: Robot Path Using VLSAM. (courtesy of Gianmarco Chumbe/CNET)	41
Figure 20: Robot Path Using SLAM with a Lidar Sensor (courtesy of Gianmarco Chumbe/CNET)	42
Figure 21: Use of Maximum, Restricted, and Operating Space	53
Figure 22: Nvidia GPU Optimized Models (Courtesy of Nvidia)	57
Figure 23: Object Detection Model Performance on Jetson Nano (Courtesy of Nvidia).....	60
Figure 25: Object Detection Bounding Box Coordinates	63
Figure 26: Object Detection Bounding Boxes (Courtesy of Algorithmia)	64
Figure 27: Detection Bounding Box Displacement from Image Center	65
Figure 28: MCU Software Flowchart.....	68
Figure 29: Three Wheel Design Created in TinkerCAD	72
Figure 30: Project Illustration by Alex Rizk	76
Figure 31: Prototype designs for textured gripper and soft robotics gripper for the purpose of picking up litter	78
Figure 32: Flexible gripper at rest (Courtesy Layershift).....	80
Figure 33: Flexible Gripper Actuating (Courtesy Layershift)	81
Figure 34: 3D Model of the Soft Portion of the Gripper	82
Figure 35: 3D Model Dimensions of the Soft Portion of the Gripper	82
Figure 36: Top-down view schematic for trash bucket.....	84
Figure 37: Side view schematic for bucket	84
Figure 38: Illustration of modular arm pieces.....	85
Figure 39: Illustration of modular arm as one piece.....	86
Figure 40: Final Arm Design	87
Figure 41: 3D model of the solid component of hybrid gripper	90
Figure 42: Dimensions of hybrid gripper	90

Figure 43: Final Design for Trash-E	92
Figure 44: Full design of the hybrid gripper.....	93
Figure 45: Wiring Diagram for A4988 Motor Driver (Courtesy Polulu)	94
Figure 46: 12V to 5V Buck TPS52903RPJ Voltage Regulator.....	98
Figure 47: 12V to 3.3V TPS52903RPJ Voltage Regulator	98
Figure 48: 12V to 5V Buck SC4524F Voltage Regulator Circuit.....	100
Figure 49: 5V to 3.3V Buck SC4524F Voltage Regulator Circuit.....	101
Figure 50: 12V to 5V Buck NR110E Voltage Regulator Circuit	101
Figure 51: Voltage Regulator PCB	103
Figure 52: Microcontroller PCB Schematic	106
Figure 53: Design 1 PCB Layout.....	109
Figure 54: Design 2 PCB Layout.....	109
Figure 55: Top Side of Trash-E MCU Breakout.....	111
Figure 56: Bottom Side of Trash-E MCU Breakout.....	111
Figure 57: Front of Final MCU PCB.....	112
Figure 58: Back of Final MCU PCB.....	112
Figure 59: Trash-E Hardware Block Diagram	114
Figure 60: Electrical Connection Layout	124
Figure 61: 12-5V Voltage Regulator Breadboard	125
Figure 62: 12-5V Voltage Regulator Breadboard	125
Figure 63: Test Setup for Ultrasonic Sensor	128
Figure 64: 60cm Output (Left) and 30cm Output (Right)	128
Figure 65: 15cm Output (Left) and 5cm Output (Right)	129

List of Tables

Table 1: Requirements Specifications.....	4
Table 2: Project Features	7
Table 3: Microcontroller Comparison.....	10
Table 4: Components Requiring Power	25
Table 5: Possible Battery Types (Courtesy Radek Jarema)	27
Table 6: Battery Specifications.....	28
Table 7: INR18650 Li-Ion Batteries Specifications	29
Table 8: LP616594 Li-Poly Batteries Specifications.....	30
Table 9: Lidar Sensors	42
Table 10: UART Baud Rate Register Configuration	67
Table 11: PWM Register Configuration	69
Table 12: Battery Options	95
Table 13: BMS Board Options	97
Table 14: Possible Voltage Regulators	99
Table 15: Components Needed for Regulator Circuits	102
Table 16: Possible Solar Panels	104
Table 17: Conflicting Pin Functionalities.....	106
Table 18: Microcontroller Schematic Components	107
Table 19: PCB Manufacturing Costs.....	109
Table 20: Bill of Materials	113
Table 21: Components used in breadboard testing	126
Table 22: Input and Output Voltage of 12 - 5 V Step Down Regulator.....	126
Table 23: Input and Output Voltage of 5 - 3.3 V Step Down Regulator.....	127
Table 24: Ultrasonic Sensor Testing Results	128
Table 25: Fall Milestones	132
Table 26: Spring Milestones.....	133

1.0 Executive Summary

The only form of automated commercial vacuum which the average consumer can buy are Roomba®-style devices. The main issue with these devices is the size of objects that can be picked up. Typically, the size of a candy wrapper, small cereal, or dust is the maximum that these autonomous vacuums can grab. These autonomous vacuums also have trouble when they encounter things that can tangle their wheels such as pet or human hair. Our robot will be able to pick up larger objects using an arm and pincer mechanic.

In this project we created Trash-E, an autonomous robot that detects litter and picks it up utilizing computer vision. It helps to help automate the process of cleaning up a venue such as a ballroom, sporting event field, or even a backyard. Trash-E differs from a Roomba®-style device because Trash-e is not an autonomous vacuum. Trash-E instead uses an arm to pick up objects such as cups that were thrown on the ground. The purpose of the project is to create a device that reduces the manpower required to do a tedious task such as cleaning up after an event. The robot is not meant to replace Roomba®-style devices, as they perform a different function. The autonomous vacuums are meant for home use.

Our robot may be useful in any event with a large gathering of people such as a concert, football game, or party. In these events the last thing on attendee's minds is their trash and it is not uncommon for the venues to be filled with garbage after the event ends. These events have teams of workers who are dedicated to cleaning up all the trash that is left over. With larger venues, cleaning up may require upwards of 100 people.

The most important part of Trash-E is the ability to detect objects that we determine are trash. For this project, we limited the scope by determining trash only as red solo cups at first. This allows us to have a set goal in mind, rather than finding random objects and determining if it is litter. By limiting the description of trash, we made sure the robot excelled at detecting that object and fine tuned any mechanical movements.

On the hardware side, Trash-E can properly pick up these red solo cups. The entire system is powered by a single rechargeable battery pack. The voltage from the battery pack is stepped down to meet the needs of the different components on Trash-E such as the microcontroller, and the different motors that need to be driven. While the battery pack can be charged with a power supply. We implemented diodes into the circuit so that the current does not flow from the battery to the panels.

2.0 Project Description

2.1 Project Background

Litter is an increasingly difficult problem to address as the world progresses. Large amounts of money is spent to pay individuals to pick up leftover objects after many different events such as tailgates, and parties. Due to the chaotic nature of those situations, it is extremely difficult to govern and manage each individual and ensure every piece of litter is brought to a place to be disposed of. Therefore, venues choose to clean up after the event rather than take preventative measures. This leads to another problem that once litter reaches a certain size, it is too large to pick up multiple at one time. One must bend down, grab the litter, then put it in a receptacle for storage until they may properly dispose of it.

A robot does not tire from doing repetitive tasks for hours on end, making it a perfect fit for this situation. Robots that pick objects up have an arm apparatus, pincers to grip the objects, and a place to store them. They're driven by different motors which are controlled either manually or autonomously. This robot must also utilize a way to move about an area autonomously and detect a cup using computer vision.

To be autonomous, it needs to have a power source attached to it. One of the most common options is a battery. This either needs to be recharged or replaced, but there's no way to make the robot run continuously without stopping at some point.

One feature that was implemented is a receptacle that is attached to the robot for ease of storage. This eliminates the need to go back to a predetermined spot to drop the object off.

2.2 Objectives

2.2.1 Motivation

The motivation for this project was to not only utilize the knowledge we have acquired over the course of our academic careers into a single project, but to challenge ourselves and build upon those skills while learning new ones. The team dynamic makes it more realistic in terms of what we will be facing when we exit our undergraduate career and choose to pursue a career in industry or research. The topics that are covered in this project like embedded system design and programming, computer vision and machine learning, and power system engineering are things that each member expressed interest contributing to at least one of, if not more. To put more marketable skills on our resume, we wanted to do a full system design that utilizes multiple different aspects of the engineering design process so we may realize a product from start to finish.

2.2.2 Goals

The goal of this project is to create a functional robot that can:

- Move on its own.
- See an object on the floor.
- Determine if the object is litter.
- Pick up the litter.
- Map out an unknown environment.
- Navigate through the environment.
- Dispose of the litter into the bin on its chassis.

2.2.3 Definition of Litter

Litter is a broad term, and to create a robot that can pick up all litter would not fit within the scope of the project. According to Merriam-Webster, litter is defined as “trash, wastepaper, or garbage lying scattered about”. With such a generic definition, many different objects could fit into that. Examples of some common items that can be referred to as litter include: candy wrappers, cans, cups, and bottles. With the word “garbage” being in the definition also, this makes litter even more ambiguous. The famous saying “one man’s trash is another man’s treasure” truly comes into effect here. Because of this uncertainty, we defined what the robot will see as litter, trash, and garbage. To start off, we used 16-ounce Red SOLO cups. These cups are 3 x 3 x 5 inches in size. We used these cups because they are very popular at large events and often are discarded on the ground at events such as college parties. The cups are a distinct red color and have a distinct shape, which aided in easing the process of identifying the cups.

2.2.4 Definition of Obstacle

In our use case, an obstacle is anything that can impede the pathing of the robot. Anything greater than the size of a Red Solo Cup, 3 x 3 x 5 inches, was classified as an obstacle to avoid. We used wheels big enough to roll over anything smaller than the size of a Red Solo Cup if necessary.

Other obstacles we needed to consider are obstacles that are too big for the camera to see the whole object. For example, a wall did throw off the obstacle detection, and thus needed to be considered when programming.

2.3 Requirements Specifications

* “The system” refers to Trash-E and all equipment

Table 1: Requirements Specifications

Description	Value	Unit
Maximum amount of trash picked up at once	1	-
Maximum weight of trash picked up at once	2	lbs
Maximum size of trash	3x3x5	in ³
The system shall be able to detect Red Solo Cups	-	-
The system shall be able to be moved by a human	-	-
Maximum time to pick up one piece of trash	30	sec
Maximum speed of robot	5	mph
Maximum height of robot	13	in
Maximum width of robot	13	in
Maximum length of robot	20	in
Maximum weight of robot	10	lbs
Maximum rotation of robot	360	degrees
Range of motion of the robot arm	0-100	degrees
Maximum range of motion of the gripper	0-150	degrees
Max cost of robot	400	USD
The robot will have ample heat dissipation to protect the components	-	-
Minimum power supplied to circuitry	12	V
Minimum voltage supplied by voltage regulator	3.3	V
Maximum voltage supplied to microcontroller	3.63	V
Minimum voltage supplied to microcontroller	3.15	V
Maximum voltage supplied by voltage regulator	5	V
Minimum battery life	1	hr

The battery shall be rechargeable	-	-
Maximum size of bucket on the robot	8	L
The battery pack shall be capable of charging through a US wall plug.	-	-
Battery pack maximum current discharge	12	A
Maximum battery pack weight	300	g
Maximum number of batteries	6	-
Minimum battery capacity	1250	mAh

2.4 Customers, Sponsors, Contributors

For this project there are no customers. There is no intention of selling the result of this product on any market, to any company, or to any individual. This project also has no sponsor. Each member of the group individually contributed to the expenses of the project. The design of this project was thought of by the members of this group. The decision to do this project was unanimously voted by everyone in the group. Each member has their own section that was agreed upon at the start of the project.

2.5 Marketing and Engineering Requirements

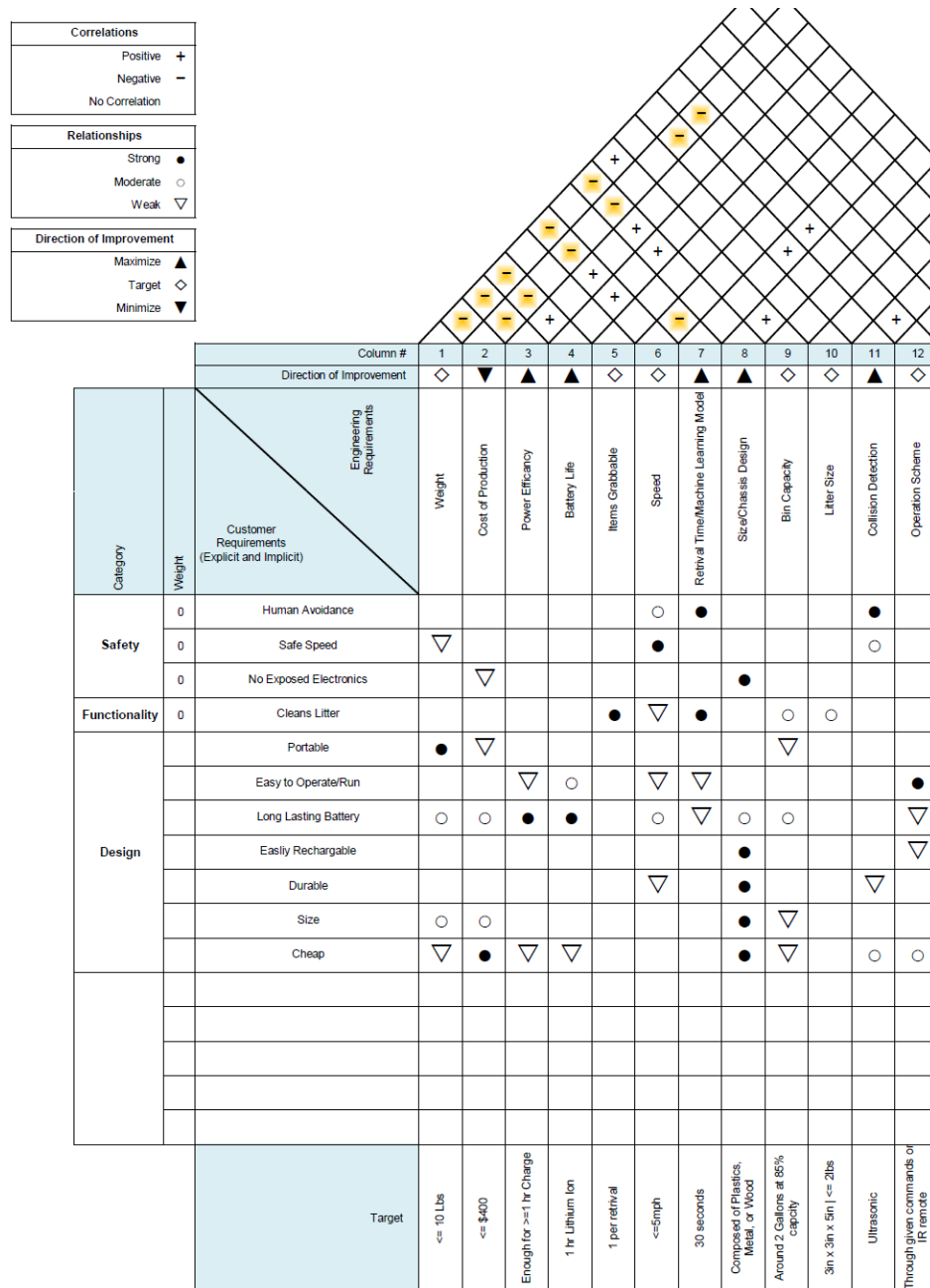


Figure 1: House of Quality

2.6 Product Features

The features table is split up between initial, primary, secondary, and stretch goals. The initial and primary goals are what we wanted to make sure the robot does. The secondary are goals that we hoped to accomplish, and the stretch goals are goals that are unnecessary but might have been added if we had the time and budget.

Table 2: Project Features

Description	Feature Type
Identify Red Solo Cup	Initial Feature
Move Around Freely with SLAM	Initial Feature
Able To Pick Up Litter	Primary Feature
Move Arm To Litter And Bin Properly	Primary Feature
Identify Aluminum Can	Primary Feature
Identify And Avoid Obstacles	Primary Feature
Store Litter In Bin	Primary Feature
Solar Panels	Secondary Feature
IR Remote Control	Secondary Feature
Sense When Bin Is Full	Secondary Feature
Function For 1 Hour	Secondary Feature
Return Home When Bin Is Full	Stretch Goal
Maneuver Through Hard Terrain	Stretch Goal
Visual And Audio Cue When An Object Is Picked Up	Stretch Goal
Empty Bin Into Large Trash Bag	Stretch Goal
Smart Arm	Stretch Goal

3.0 Research

3.1 Existing Products

A general consumer's options for a device that completes this task autonomously were very limited. Based on our research, there was only a Roomba® style device that could be purchased. One problem with this style of robot was the size of objects it could pick up. Roomba®s were designed with the intention of replacing vacuum cleaners that companies such as Dyson® and Bissell® manufacture. This means their goal was to pick up very small objects like crumbs, dirt, dust, and other similar things that could be collected without the use of the vacuum wand.

Another problem with the Roomba® was the capacity of the container it stores litter in. Even if a venue only had litter that was capable of being picked up by it, the container would get full too quickly and would require a person to empty it too often.

3.2 Microcontrollers

A microcontroller was used to control the movement of Trash-E. The Jetson Nano, which is responsible for computer vision and object detection, sends information regarding the position of the cup with a serial communication protocol using GPIO pins on both the microcontroller and Nano. Microcontrollers were an optimal choice to accomplish this task due to their versatility and low cost. Since they had many operations built-in like pulse-width modulation and analog-to-digital converters, the overall PCB design was simpler since we didn't have to implement these circuits ourselves.

3.2.1 Communication Protocols

3.2.1.1 Universal Asynchronous Receiver/Transmitter (UART)

UART was the simplest of the three. Using a maximum of two pins per device, receiving and transmitting could be achieved between two devices easily. Since this protocol is asynchronous, it does not need a pin for a clock signal which freed up a pin on our microcontroller that could be used for other functions. While this method was very straightforward, it had major drawbacks. Due to the asynchronicity of UART, data could be transmitted whenever it wants and both devices must be listening constantly. Another drawback was that only two devices could communicate at once.

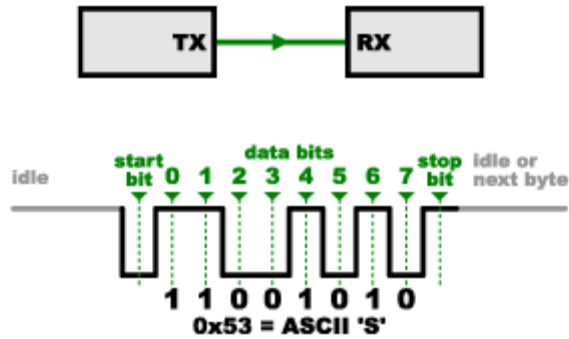


Figure 2: Example Transmission Using UART (Courtesy of SparkFun)

With this information in mind, UART was a good choice to communicate between the microcontroller and the Jetson Nano. We minimized the number of pins utilized since data would only be sent from the Nano to the microcontroller. Since the communication would always be happening due to the asynchronicity, more processing power would be consumed on both the Jetson Nano and the microcontroller. Figure 2 illustrates how we would send data from the Jetson Nano to the microcontroller using a low start bit, a high stop bit, and no parity bits. On the other hand, it would be troublesome to utilize this protocol between the microcontroller and the peripherals. A separate UART would need to be created for each microcontroller and peripheral, resulting in excess pins being used.

3.2.1.2 Serial Peripheral Interface (SPI)

Unlike UART, SPI is a synchronous protocol meaning it utilized a clock signal to communicate between two devices. SPI also offered multiple peripheral capabilities by utilizing a chip select signal per device. With the addition of the clock signal and chip select signals, this could greatly increase the number of pins needed to implement this protocol which wasn't ideal for a project with many different peripherals.

SPI could be used for the communication between the Nano and microcontroller but was not needed due to the one-way transmission between the devices and would be wasting pins. It is very helpful for the communication between the microcontroller and the peripherals. Having a dedicated way to talk to multiple destinations from one source was very beneficial even with the extra pin cost. This method also allowed for one way communication to movement peripherals such as continuous servos, while having two-way communication between others like the precision servos all in the same system. If pin space became a problem, the daisy-chaining method could be a potential solution to reduce the amount of pins used.

3.2.1.3 Inter-Integrated Circuit (I2C)

I2C is a synchronous communication like I2C but only uses two pins, much like the UART. I2C offers the advantages of both UART and SPI but falls short in speed.

For a single frame of data, UART and SPI can accomplish this task in one total frame, whereas I2C accomplishes it in two. This method was certainly viable for the communication between the Jetson Nano and microcontroller, but again was unnecessary since there was only one controller and one device. I2C is very useful for the communication between the microcontroller and peripherals because of the low number of pins used, as well as the two-way communication. Many boards, like the TM4C1232H6PMI7, have an I2C interface which would help us implement this protocol.

3.2.2 STMicroelectronics (STM) vs. Texas Instruments (TI)

	STM32G0B1KCT6	STM32L151CCT6J	STM32L071CBT6	TM4C1233H6PZI	TM4C1232H6PMI7	STM32G0B1KET6N
Price	\$5.62	\$5.44	\$5.01	\$8.66	\$7.14	\$6.30
Core Processor	ARM Cortex-M0+	ARM Cortex-M3	ARM Cortex-M0+	ARM Cortex-M4F	ARM Cortex-M4F	ARM Cortex-M0+
Operating Voltage	1.7V – 3.6V	1.8V – 3.6V	1.8V – 3.6V	1.08V – 3.63V	1.08V – 3.63V	1.7 – 3.6V
Core Size (Bit)	32	32	32	32	32	32
Speed (MHz)	64	32	32	80	80	64
# of I/O pins	30	37	40	69	49	29
Program Memory (kB)	256	256	128	256	256	512
CoreMark [®] /MHz	2.46	3.34	2.46	3.42	3.42	2.46
Mounting Type	SMD	SMD	SMD	SMD	SMD	SMD
Package	LQFP	LQFP	LQFP	LQFP	LQFP	LQFP

Table 3: Microcontroller Comparison

To determine the candidates in Table 3, we decided to narrow the manufacturers to two companies. Even though two microcontrollers could utilize the same core processor, there are many differences in how companies design their microcontrollers and expect the user to interact with them. For this project we decided to research microcontrollers made by TI and STM.

TI was a company that we became familiar with through our academic program at UCF, making it a great choice to pick a microcontroller from. We previously utilized the MSP430FR6989 to learn common embedded practices which allowed us to get accustomed to the recommended IDE, as well as utilized their syntax and processes to accomplish the basics. We didn't want to reuse the MSP430FR6989 for this project so we can learn more about what different TI microcontrollers have to offer, while not straying too far from our current knowledge base.

STM was another company that has a good reputation through word of mouth and forums on the internet. With many different microcontrollers that are specific to embedded applications, they also have plenty of development boards that we can utilize to prototype our system with before ordering a custom PCB. None of us had worked with an STM microcontroller or their software. This made STM microcontrollers perfect for us to research and compare to the more familiar TI.

3.2.3 ARM Cortex-M

ARM Cortex-M was a 32-bit Reduced Instruction Set Computer (RISC) processor core which was optimized for low-cost, energy efficient integrated circuits in many embedded applications[1]. With the huge popularity of this instruction set, we wanted to pick a microcontroller that utilizes this technology.

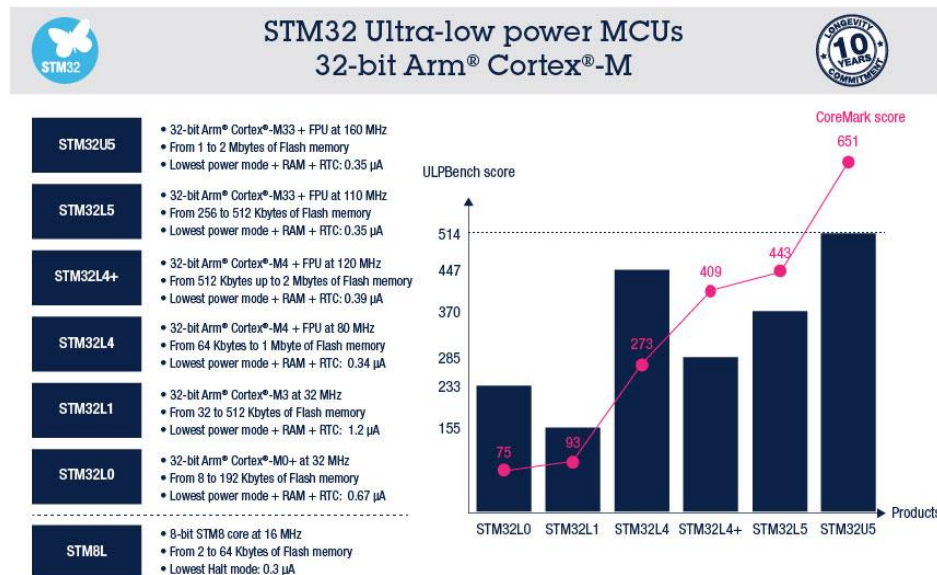


Figure 3: Benchmarks for Different Cortex-M Processors (Courtesy of ST)

3.2.3.1 CoreMark and ULPBench Analysis

To determine how well a processor performed, we looked at the ULPBench score as well as the CoreMark score. ULPBench (Ultra Low Power Bench) determines how energy efficient a particular microcontroller is. CoreMark tests the functionality of a specific processor core. Table 3 has the CoreMark score divided by the frequency it was running at to get a more accurate representation of the score as this considers how many instructions the processor can execute in a second. [2][3] Figure 3 displays the ULPBench scores in the bars and the CoreMark score on the line for each STM32 variation. While this information wasn't for the specific microcontrollers in our table, each Cortex-M in the figure utilized the same specs as the microcontrollers we were researching. We investigated the official EEMBC benchmark table for these scores, the company in charge of maintaining the benchmarks these scores are made from but couldn't find anything regarding the microcontrollers we picked. This was because scores do not have to be submitted, but also can only be submitted by members or licensees of EEMBC [4]. Therefore, this was the closest information we could acquire without buying each microprocessor and conducting the benchmarks ourselves.

Due to product availability, we considered three different Cortex-M variations: Cortex-M0+, Cortex-M3, and Cortex-M4. With M0+ and M3 being run at much lower processor speeds, we compared them more closely to each other than with M4. We saw that M0+ had a significantly lower CoreMark score than M3, it also had a larger ULPBench score, indicating that the processor could potentially be more energy efficient. When looking at M4, we saw that the CoreMark and ULPBench scores were drastically higher than the other two, making the architecture more enticing. This was due to "The combination of high-efficiency signal processing functionality with the low-power, low cost and ease-of-use benefits of the Cortex-M family". [5] Based on these results, Cortex-M4 was very enticing for us to choose.

3.2.4 Core Size

Core size of the microcontroller's processor indicated how many bits of information could be passed into the data bus and processed in one clock cycle. The higher the core size, the more bits could be processed and the larger the value could be for one variable, but also required more storage even for variables with few amounts of bits used. We decided to go with a 32-bit core size since many languages utilize this size and we were familiar with programming languages that have 32-bit variables standard.

3.2.5 Core Speed

Core speed tells how many clock cycles happen in one second. Generally, the higher the core speed means that more instructions could be computed in one second although this is not always true since it was based on the instruction set

used. All the microcontrollers we investigated had a minimum core speed of 32 MHz which would be adequate for Trash-E's application.

3.2.6 I/O Pins

These pins would be used to connect to other devices/peripherals and communicate between them. We had to be careful that we didn't choose a microcontroller with too few pins since we would not be able to implement all planned functions of Trash-E. We also wanted to have some excess pins in case we could implement some of our stretch goals later. For this reason, we decided the microcontroller should at least have 40 I/O pins.

3.2.7 Program Memory

The amount of FLASH storage in the microcontroller was critical to the decision-making process. This storage was the area where our code would reside. The more complex the application, the more lines of code we wrote which, in turn, increased the size of our file. If we got a microcontroller that had too little storage space, we would have to either buy a new microcontroller that had more storage or increase the storage capacity by adding an external storage device of some sort. Without having any code written it was extremely difficult to determine how much is "too little". Two different algorithms could achieve the same thing but take two different approaches. If one approach was poorly optimized or has more lines of code, that option will be larger and could go over the FLASH capacity. With our inexperience in the subject we decided on 256kB as the minimum FLASH storage.

3.2.8 Mounting Type

Since the PCB would be soldered ourselves, the mounting type of our components had a big influence on our decision. There are two types of components we can choose from: Through-hole (TH) and Surface Mount (SM).

3.2.8.1 Through-Hole

Since this mounting type utilizes pins that go through the board and the components being relatively large, it's easier to solder and can normally be done using only a soldering iron and solder. It would be easy to verify the solder process went well and there's no solder bridging due to the spacing between pins being greater than SM components. They are also compatible with solderless breadboards which is a huge advantage. Prototyping on breadboards will allow us to attach the components directly into the breadboard, exchange parts, and alter our design without needing to solder and desolder each component. With the larger sizes of TH components, they take up more space and increase the total PCB size. A disadvantage to microcontrollers specifically is that as the microcontroller increases in complexity and adds more features, more pins are needed, and TH is no longer viable. This restricts the complexity of circuits if TH is

being used for a microcontroller. We wanted to utilize TH for all the components to make the soldering process simpler but it wasn't viable for the scope of our project.

3.2.8.2 Surface Mount

SM utilizes pads instead of holes for the component to connect to. SM components are much smaller than a TH component which greatly reduces total PCB cost and size. Size can potentially be further reduced, or functionality can be increased for the same size, by mounting components to both sides of the PCB. Soldering SM components by hand can be very challenging compared to TH since the pads are very close together. Verifying the quality of the completed solder is also more difficult and will require the use of a microscope to ensure there is no solder bridging and the pins are making direct contact with the pads. To ensure costs of the PCB and components stay down, we will be utilizing SM technology for our microcontroller and any other basic components as much as we can.

3.2.9 Package

The package refers to the way a component connects with the PCB. For the microcontroller, there were three different SM package variations: Quad Flat Package (QFP), Ball Grid Array (BGA), and Quad Flat No-Lead (QFN).

3.2.9.1 QFP

The closest to a TH component, QFP has little leads coming out of the chip that allows it to sit directly on the pads. Due to the leads being visible, this makes it easier for an individual to solder by hand and was a highly sought-after package for us. The problem with this package was that as the number of pins on a chip increases, the size of the leads and pads, as well as the space between them, decreases. We determined that a total pin size of around 64 pins is ideal to reasonably be able to solder this by hand. This package also helped us keep utility costs down as it could be soldered with a regular soldering iron. This package is also very prone to mistakes since we will be using a soldering iron. If we are not careful, we could damage the board itself and have to restart on a new one.

3.2.9.2 QFN

This package is very similar to the QFP but instead of the leads coming out of the chip, the leads are tucked under the chip, and has a metal pad that acts as a heat sink in the center. This could be more beneficial for our PCB than the QFP due to this extra heat dissipation from the center pad. Without having leads extending from the chip, this makes it more difficult to solder. An easier solution is to coat the pads with a flux paste and drag the solder across. The chemical interaction between the flux and solder will allow the solder to fall into place on the pads. This is still not ideal for us but is doable while also keeping costs low.

3.2.9.3 BGA

BGA was the least optimal for us to use and we steered clear of when looking for potential microcontrollers. Because the connections are on the bottom of the board, more can be placed than the other two packages. The more connections mean more placement of solder balls and doing this by hand can take a long time and requires high precision. To solder them, a hot air solder gun or reflow oven is needed which we do not have and will increase our costs since we would have to purchase a hot air solder gun. Because of this, we decided QFP or QFN are the packages we need to restrict our search to.

3.2.10 Price Per Unit

We wanted to select a microcontroller that fit the above needs/preferences, while also keeping costs to a minimum since the price per unit of the microcontroller could break our budget. With it being our first time getting PCBs printed and soldering components onto them for a project, we had to account for mistakes. This meant we ordered multiple PCBs and microcontrollers to verify our prototype still worked after soldering on the components. With this in mind, we set our absolute max price to \$10 per microcontroller.

3.2.11 TM4C1232H6PMI7

With the considerations of sections 3.2.1 through 3.2.10, we decided to select Texas Instruments' TM4C1232H6PMI7. One key factor to this decision was the familiarity of the software IDE would allow us to move quickly while developing and not spend unnecessary time relearning the basics. Another key factor was the cost. We want to keep everything under budget and this microcontroller allows us to buy multiple in case one gets damaged and needs to be replaced. Most of the following information comes from the TM4C1232H6PMI7 datasheet and was not our work.

3.2.11.1 JTAG

Since we weren't using a development board, we needed to implement a way to program the microcontroller. "The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic." [6] JTAG allowed us to flash our code onto the microcontroller with an IEEE backed protocol. Only four pins were used on the microcontroller due to this protocol. Those pins are: TCK, TMS, TDI, and TDO. Using serial transmission, we could send data to the microcontroller with TDI with the TCK clock signal for controlling the speed.

3.2.11.2 Clock Signal

The TM4C1232H6PMI7 has multiple options for clock signals that can be used in the microcontroller. The Main Oscillator (MOSC) provided a very accurate clock source by utilizing an external crystal oscillator. The microcontroller supported crystal oscillators with frequencies between 5 and 25 MHz. While the speed and accuracy was enticing, using the MOSC will introduce more required capacitors on the PCB, increasing size and decreasing space on the PCB while increasing production costs. Due to this reasoning we selected the Precision Internal Oscillator (PIOSC).

The PIOSC was a clock source that was integrated onto the chip and used by default. There was no required use of external parts or crystals for it to function. It provided a 16-MHz clock source to the chip with a +/- 3% accuracy due to temperature. The internal clock was implemented using resistors and capacitors which made it less accurate than a crystal oscillator due to the increased temperature of the components. While the max speed of the PIOSC was 36% slower than the MOSC and less accurate, it would be more than sufficient for Trash-E which only needed to generate signals for servo/stepper motors and ultrasonic sensors.

3.2.11.3 PWM Generation

To generate the PWM signals for the motors and sensors, multiple timers need to be used which the microcontroller is in no short supply of. "The TM4C1232H6PM General-Purpose Timer Module (GPTM) contains six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM blocks. Each 16/32-bit GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or concatenated to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Each 32/64-bit Wide GPTM block provides 32-bit timers for Timer A and Timer B that can be concatenated to operate as a 64-bit timer." [6] With twelve GPTM blocks and two timers per block, we had the potential to generate twenty-four PWM signals. This was plenty to implement our original design as well as accommodate for any stretch goals that can be implemented in the future

3.2.11.4 UART Interfaces

The TM4C1232H6PMI7 also had eight separate UART interfaces that were fully programmable. With baud rate generation of 5 Mbps for regular speed and 10 Mbps for high speed, there was no problem sending data between the microcontroller and Jetson Nano. There's also separate transmit and receive FIFOs that reduced the CPU interrupt service loading and have programmable length. The interface also gave us full control over the serial communication characteristics such as the amount of data bits, either one or two stop bit, and even, odd, or no parity bit.

3.2.11.5 Sleep Modes

Arguably the most important feature that this and many other microcontrollers offer was the Low Power Modes, or in the TM4C1232H6PMI7's case, Sleep Modes. By entering a Sleep Mode, power consumption was kept to a minimum. Depending on what functions are needed to keep running during sleep, either Sleep mode or Deep-sleep mode can be chosen. Sleep mode only stops the processor clock while Deep-sleep mode stops the system clock as well as switches off the Phase Locked Loop (PLL) and Flash memory. Since Deep-sleep mode turns off not only the processor clock but the system clock, PLL and flash, we wanted to use Sleep mode. This mode will allow us to keep Trash-E moving while there is nothing to process from the Jetson Nano. In comparison to other microcontrollers, like the MSP430 family which has four different Low Power Modes, the power conservation options of this microcontroller were limited but was sufficient for our power needs since we couldn't go into too deep of sleep with Trash-E doing continuous movement at almost all times. It's also important to note that the deeper sleep modes might reduce the power consumed by the microcontroller, but also increase the amount of time required to sleep and wake.

3.3 Computer Vision

3.3.1 Computer Vision Overview

For computer vision to work, a lot needed to get done before it was used for cases like Trash-E identifying trash in real time. Computer vision required a lot of data and would need to use machine learning techniques to accomplish it. It needed to analyze a lot of data and learn from it until it could make certain distinctions in images and ultimately recognize what it needed to find in an image or video. Figure 4 is an example of what a computer sees after identifying objects via computer vision.



Figure 4: Object Detection Using Computer Vision (Courtesy of TowardsDataScience.com)

To accomplish computer vision, we needed certain algorithms that could learn from given inputs and produce an outcome on its own in a way that the human brain would. The best way to accomplish this was to use a form of machine learning called deep learning.

3.3.2 Machine Learning

Machine Learning was the term that refers to a machine becoming capable of learning from a large data set and performing actions based on what the computer has learned and the input data it was receiving. There are many types of learning that machine learning can be done in.

Supervised learning datasets are labelled manually prior to being given to a machine learning model for training. These datasets also include the expected output that the model will use to become very accurate at predicting when it comes to new input data. Unsupervised learning uses datasets that are not labelled and have no specified structure. The model will learn on its own and make classifications based on the data.

Semi-supervised learning is an approach that combines a small portion of labelled input data along with a large amount of data that is not labelled that the model will use to learn. This is typically used when there isn't a lot of labelled data available or having a complete set of labelled data is too challenging or expensive but still want to use some amount of labelled data. This type of learning can achieve better performance and accuracy than its supervised counterpart.

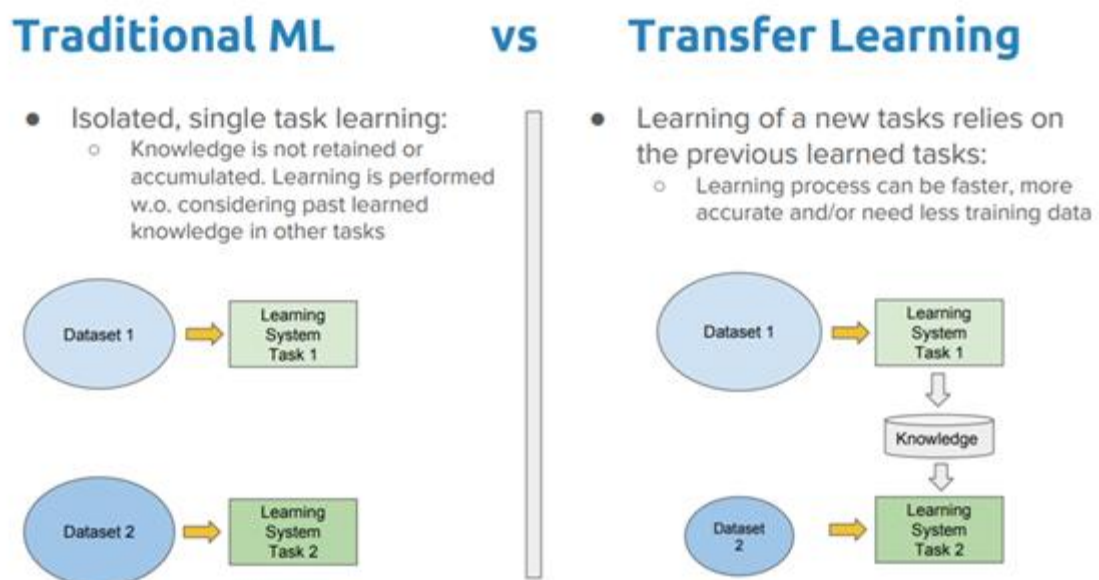


Figure 5: Traditional Learning (Courtesy of TowardsDataScience.com)

Transfer learning involves using a pretrained model and using the existing knowledge from previously learned tasks and applying that knowledge to a new task that is related as shown in Figure 5. If we had trained a previous model for object detection on cars, we could use that model's knowledge to learn how to detect trucks and other forms of vehicles and make learning faster.

3.3.3 Deep Learning

Deep learning is a subset of machine learning that uses neural networks to learn large amounts of data through lots of training.

Neural Networks are the brain of the AI and are used extensively in deep learning. These networks are meant to simulate the way that humans learn with the brain. Our brain has neurons that make connections and so does the neural network that we use for machine learning. All the neurons in a neural network are interconnected and are organized into multiple layers. These layers consist of the input layer, the hidden layers, and the output layer. The input layer of our neural network receives information to learn via our input data. The next few hidden layers in between the input and output layers are where most of the work and training is done in a neural network.

In these layers many mathematical computations are performed on our input data. The connections between all the neurons in these layers have weights which determine the importance of the input value and the strength of the connection. Each of these goes through an activation function which in simple terms standardizes the outputs of the neurons. The number of hidden layers you could have in a neural network is arbitrary. These are the layers where people usually spend time tweaking and testing this area by increasing and decreasing the number of hidden layers and number of neurons in each layer. With deep learning, the neural networks have more than one hidden layer, which is where the deep term comes from.

Once these layers have been passed and we reach the output layer, there is a loss function that determines how wrong our network's output was from the real output data. We want that function to be as close to zero as possible to get the most accurate output from our deep neural network. To improve accuracy and reduce loss, we use optimization algorithms called gradient descent and backpropagation which find the minimum of a function and in this case that is the loss. It allows a deep neural network to change its weights automatically in incremental steps after each iteration of training to achieve a loss as close to zero as possible.

Deep learning networks can be supervised but they can also be unsupervised as well. Unsupervised training is where deep learning typically shines. With deep learning, data preprocessing can be mostly eliminated. Deep learning algorithms can process unstructured data and automate feature extraction. For example,

feature extraction for computer vision can be a very challenging task to do manually and involves a lot of manual work. Images must be put through many processes to extract features such as edges, colors, and brightness. A deep learning network can be given a set of images and it will be able to determine important features that allow it to distinguish objects from each other. Through algorithm processes and training with the neural network it can learn from the data and become very accurate. This allows it to make accurate predictions based on new input data.

3.3.4 Convolutional Neural Networks

There are many types of neural networks out there. For our purposes of this project, the most tried and true deep learning algorithm for computer vision was a convolutional neural network. A convolutional neural network should be the best algorithm to implement for Trash-E to detect trash objects using image recognition. Convolutional Neural Networks (CNN) are deep learning algorithms that can take an input image and then learn various patterns and features about that image and make decisions about the image. Figure 6 is a diagram of a convolutional neural network showing its general architecture.

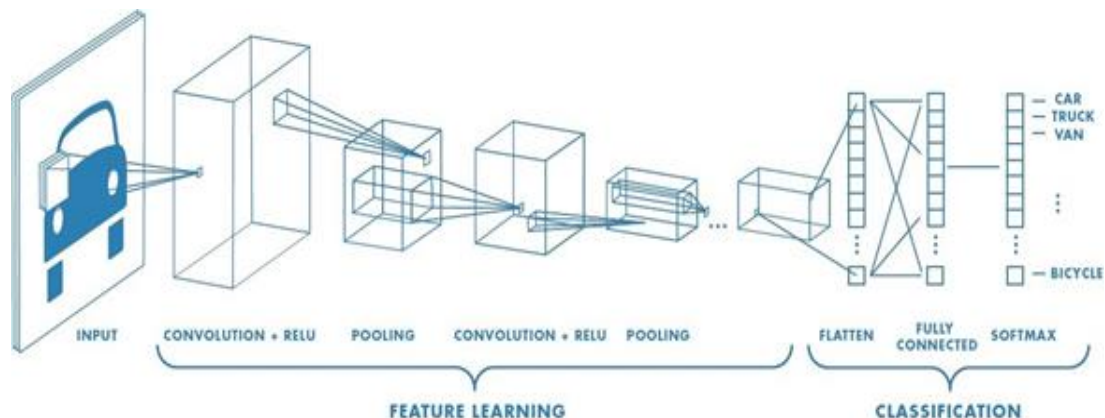


Figure 6: Convolutional Neural Network Architecture (Courtesy of TowardsDataScience.com)

The first step in the convolution neural network was the convolution layer. An image was fed into a CNN in the form of a matrix with pixel values. In the convolution layer a kernel/filter, which was a square matrix of a certain size, was typically used to hover over the original image in a certain number of shifts and strides. Each time the filter is over a new section of the image matrix, a matrix multiplication is performed and produces a new value for the image matrix. Depending on the kernel/filter values, different types of high-level features can be extracted from the input image. A CNN can capture spatial and temporal dependencies in an image through these filters. The main objective of the convolution was to extract high level features from the input image such as edges, color, gradient orientation and more. There can be more than one convolutional layer.

Following convolutions, the pooling layer follows which was responsible for further extracting dominant features and reducing the spatial size of the convolved feature by using a smaller kernel and certain techniques. This reduces computational power that is needed further into the network. There were two types of pooling techniques: average pooling and max pooling. Max pooling returns the maximum value of the image matrix that the pooling kernel was on. Average pooling returns the average of all the values of the image currently in the pooling kernel.

Now, the output is flattened into a column vector and fed into a feed-forward neural network and backpropagation is applied in every training epoch. This layer is referred to as the fully connected layer and is where the network learns the many features we have extracted from the image. After a certain number of epochs, the model can distinguish between features of the input image and classify them using the SoftMax activation function. This function would produce multiple probabilities ranging from 0 to 1 for all the classes our CNN was trained to find and output what the image likely contains.

3.3.5 Convolutional Neural Network Architectures

There are many CNN architectures out there. Some of the notable ones are: LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, and ZFNet. Each are different takes and variations on the general CNN architecture shown earlier. For Trash-E, we had many options available to choose for the CNN architecture and involved more experimentation and fine tuning to decide which architecture would return the best results. There are also many pre-trained state of the art models available on Google's TensorFlow GitHub that were trained on the COCO 2017 dataset. Some of the most popular ones that we could use are SSD MobileNet V2 and SSD ResNet. Each of these architectures have configurations for certain image sizes such as 320x320, 640x640, etc.

Model name	Speed (ms)	COCO mAP	Outputs
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes
Faster R-CNN ResNet50 V1 800x1333	65	31.6	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes

Figure 7: Available Models (Courtesy of Tensorflow)

A higher resolution architecture would take more computational power and won't be as fast as a lower resolution one but offers better mean average precision as shown in Figure 7, where the speed in milliseconds is on the left and the mean average precision is on the right. This was a tradeoff we had to decide on for Trash-E's computer vision implementation.

3.3.6 Programming Languages for Machine Learning

We had many options for the programming language we could use for writing Trash-E's software. Several programming languages are used for AI and machine learning nowadays. Some of the popular ones are Python, C/C++, and Java. The most popular of these languages for AI and Machine Learning is by far Python. For computer vision and robotics, the most popular languages to use are C/C++ and Python.

Python is an interpreted high level programming language, so it is less performant than a compiled language since the code is executed line by line. Python is one of the most supported machine learning languages out there. It is a relatively simple language in comparison to C/C++ and Java. Python has an extensive number of tools and libraries for machine learning such as TensorFlow, scikit-learn, PyTorch, and Keras to name a few. These libraries support computer vision and deep

learning allowing the ability to easily create convolutional neural networks and train them.

C/C++ are very much used in embedded and robotics programming. C/C++ are compiled languages and have very high performance. The areas they're used most for in AI are gaming and robot locomotion. They're not as simple as python when it comes to building new machine learning applications and getting what you want quickly. However, they are favored when control, high performance and efficiency is needed. C/C++ have some libraries for machine learning and computer vision such as MLPACK, SHARK, and OpenCV.

Java is less popular for embedded and robotics and is used more for desktop and enterprise applications. Java does come with a decent amount of machine learning libraries such as TensorFlow, Deep Java Library, Kubeflow, and Java-ML.

3.3.7 Libraries and Tools for Machine Learning and Computer Vision

There are many libraries available for machine learning and computer vision. We decided to investigate the most popular libraries since they have the most support and have everything we would need for Trash-E.

3.3.7.1 TensorFlow

TensorFlow was a machine learning open-source library by Google. It allowed users to build and train machine learning models using high level Keras APIs. It was a more general machine learning library for python but still offered functions that can be used for computer vision.

3.3.7.2 PyTorch

PyTorch was an open-source machine learning library for Python developed by Facebook. PyTorch could work for both Python and C++. It provides tensor computing with acceleration via a graphics processing unit or GPU. It also offered the ability to create deep neural networks.

3.3.7.3 TensorRT

TensorRT is an SDK by Nvidia that optimizes inference performance of models for Nvidia GPUs. It is used to optimize trained models from a machine learning library so that it runs faster and more efficiently on an Nvidia Jetson Nano. When a model is finished training and ready to be deployed to a Jetson Nano, first the model or in other words graph is frozen. This essentially saves the model. Once the model graph is frozen, it can be optimized by TensorRT as shown in Figure 8.

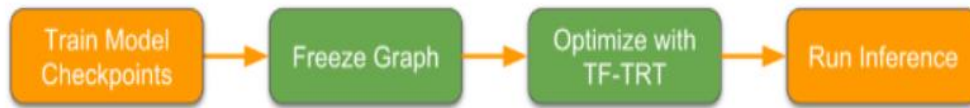


Figure 8: The TensorRT Flow (Courtesy of Nvidia)

TensorRT parsed the model and applied optimizations to the graph where it is able to. When it detects a compatible subgraph, TensorRT replaces it with a TensorRT optimized node. First, layers within the TensorFlow graph that have unused output are destroyed so that unnecessary computation is avoided. Next, convolution, bias, and ReLU layers were merged to form a single layer. Further optimizations include layer aggregation which also improves performance. Most importantly the overall original computation of the graph or model is unchanged but it is restructured to optimally perform operations more efficiently and faster as shown in Figure 9. The examples shown involve using TensorFlow but the same can be done for models created using PyTorch.

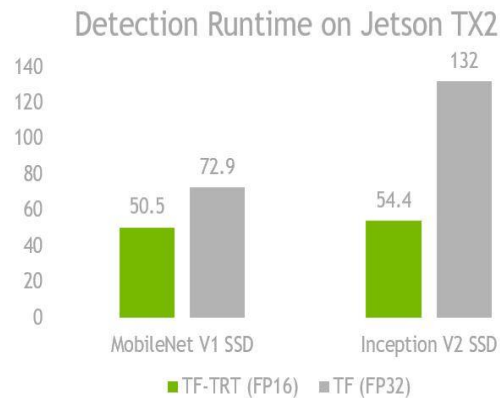
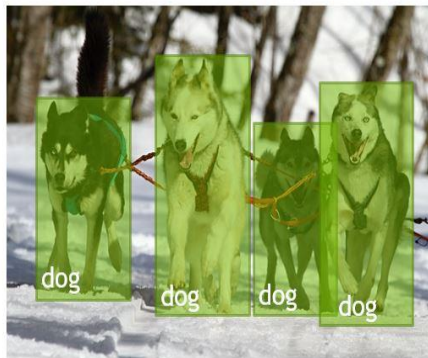


Figure 9: TensorRT Optimization Performance Graph (Courtesy of Nvidia)

3.3.8 Hardware Options for Machine Learning

Our robot uses computer vision in real-time. To process the data and algorithms we needed for computer vision and deep learning we needed a minicomputer on our robot capable of handling these tasks. These tasks included object detection, and classification. There are several out on the market such as the Nvidia Jetson Nano, Raspberry Pi 3, and Google Edge TPU. Figure 10 shows benchmark comparisons for these products. The Nvidia Jetson Nano appears to drastically outperform the other two boards in this comparison and while using the object detection model architectures that we previously stated in our research on convolutional neural network architecture.

Model	Application	Framework	NVIDIA Jetson Nano	Raspberry Pi 3	Raspberry Pi 3 + Intel Neural Compute Stick 2	Google Edge TPU Dev Board
ResNet-50 (224×224)	Classification	TensorFlow	36 FPS	1.4 FPS	16 FPS	DNR
MobileNet-v2 (300×300)	Classification	TensorFlow	64 FPS	2.5 FPS	30 FPS	130 FPS
SSD ResNet-18 (960×544)	Object Detection	TensorFlow	5 FPS	DNR	DNR	DNR
SSD ResNet-18 (480×272)	Object Detection	TensorFlow	16 FPS	DNR	DNR	DNR
SSD ResNet-18 (300×300)	Object Detection	TensorFlow	18 FPS	DNR	DNR	DNR
SSD Mobilenet-V2 (960×544)	Object Detection	TensorFlow	8 FPS	DNR	1.8 FPS	DNR
SSD Mobilenet-V2 (480×272)	Object Detection	TensorFlow	27 FPS	DNR	7 FPS	DNR

Figure 10: Minicomputer Deep Learning Benchmarks (Courtesy of Nvidia)

3.4 Power

3.4.1 Power Supply

In order to determine the power supply necessary for the robot, the requirements of the components must be looked at. In Table 4, the possible components that needed power are listed along with their required specifications.

From table 4, most of the components should be able to be powered by a normal battery bank that is often used for phones. This is because most of these components are able to be powered through the USB port on a computer, which normally has a maximum supply of 5V, 0.5A. The maximum current and maximum power columns are the maximum, and thus the devices will not be drawing that much on regular use. The Blink Mini claimed that it requires wall power on the amazon website, however with further investigation and after testing it, it should be able to be powered through a power supply. Some of the other cameras did not have readily available datasheets and without purchasing it, it was hard to find the requirements to power them.

Table 4: Components Requiring Power

Component	Voltage Requirement	Maximum Current	Maximum Power
Jetson Nano	4.75V	4A	19W
Arduino Uno	7-12V	50mA	.35-.6W
Raspberry Pi 3	5V	2.5A	12.5W

Motor Driver	8-35V	1A	8-35W
MG996R 55g Metal Gear Torque Digital Servo Motor	5V	3A	15W
Stepper Motor	12V	1.2A	14.4W
AREBI Spy Camera Wireless Hidden WiFi Mini Camera HD	4.2V	300mA	1.26W
Blink Mini	100-240V	.15A	15-36W
Logitech C270 HD Webcam	5V	1A	5W
NexiGo N60 USB Computer Camera	5V	1A	5W
HC-SR04 ultrasonic sensor	5V	15mA	.075W

There were a few options for powering Trash-E. The first option and the least likely one will be a readily made battery bank that normally is used as backup batteries for phones. This option is the least likely, as we want to integrate our own design, from the batteries to the voltage regulation. The next few options that can be used are readily made batteries such as Li-Ion (Lithium Ion) batteries, Li-Poly (Lithium Polymer) batteries, or NiMH (Nickel-Metal Hydride) batteries. These three batteries are the best options because while there are many more, these are inexpensive, and are not difficult to find or charge.

Table 5: Possible Battery Types (Courtesy Radek Jarema)

Battery type	NiMH	Li-Ion/Li-Poly high energy type	Li-Ion/Li-Poly high current type	Li-Ion/Li-Poly high safety type
Chemistry	NiMH	LiCoO ₂ LiNiMnCoO ₂	LiMn ₂ O ₄	LiFePO ₄
Nominal cell voltage	1.2V	3.6 - 3.7V	3.7 - 3.8V	3.2 - 3.3V
Operating cell voltage range	1.0 - 1.4V	3.0 - 4.2V	3.0 - 4.2V	2.5 - 3.65V
Max voltage for charging	1.4 - 1.6 V	4.2 - 4.3V	4.1 - 4.2V	3.65V
Max charging current*	0.1C or 1C (only with $\Delta V/ \Delta T$)	~1C	up to 3C	up to 4C
Max discharging current	1C - 30C	1C - 2C	5C - 30C	1C - 40C
Charging method	CC with timer or $\Delta V/ \Delta T$ (faster)	2 stages: CC and then CV	2 stages: CC and then CV	2 stages: CC and then CV
Specific energy	40 - 120Wh/kg	150 - 250Wh/kg	100 - 150Wh/kg	90 - 120Wh/kg
Specific power	100 - 1000 W/kg	100 - 400W/kg	400 - 5000 W/kg	200 - 7000 W/kg
Internal series resistance for a single cell	5-50 m Ω (for 18650 size)	15-100 m Ω (for 18650 size)	10-50 m Ω (for 18650 size)	6-60 m Ω (for 18650 size)

The above table showcases the batteries that were outlined previously. From this table, it can be noted that the Li-Ion/Li-Poly batteries may be the best for our use case because they have a much higher voltage than the NiMH batteries, though NiMH have a better power-to-weight ratio and are safer than the Lithium batteries. The “C” in the current rows of the table is the capacity divided by hour. To understand the significance of “2C”, take an example battery capacity such as 4000mAh, and multiply by 2C and the result is 8000mA. Once all the parts were fully determined, we had find the battery necessary to last 1 hour based off battery consumption of the different parts.

The batteries can also come in different constructions such as cylindrical, prismatic, or in a pouch. The Li-Ion batteries can come in cylinders or prismatic, but they require metal enclosures, whereas Li-Poly batteries can come in the previous or in a pouch as well. There are many cylindrical Li-Ion batteries, ranging in different diameters and lengths such as 14500, also known as AAs. These cylindrical batteries are a likely option, as battery packs can be constructed from them that we can use to power the Trash-E. The prismatic cells are unlikely to be useful in our application. Li-Poly pouches are another possibility, as they can be stacked and are often used for RC cars, drones, and other high-power applications. However, the Li-Poly pouches must be secured safely within the robot and there must not be any sharp objects inside, because the Li-Poly batteries are much easier to be pierced than the Li-Ion ones.

Since our parts needed to have a maximum of 12V, we aimed to create a 12V battery pack to power Trash-E. Using voltage regulators, we stepped down the voltage to necessary voltages when needed such as for Jetson Nano.

In the end, we ended up using 3 14.8V nominal 2600mAh Lithium-Ion battery packs. These batteries were donated to us by Smart Charging Technologies to be used in our project. These batteries worked perfectly in our use. To power the Jetson Nano and Lidar unit, we used an external battery pack, as permitted by Dr. Richie, as the USB-C controllers that we needed were all out of stock at the time of ordering.

3.4.2 Battery Requirements

Table 6: Battery Specifications

Requirement	Specification
Size	Less than 200cm ³
Weight	Less than 1kg
Nominal Voltage	10-12V
Maximum Discharge Current	10A
Capacity	Greater than 1250mAh

In table 6, the specifications of the battery can be found. The size of our battery pack needed to be small enough to fit on Trash-E while minimizing the space it took. We hoped that this size and weight would be sufficient. To find the capacity and maximum discharge current, we used the following equation.

$$\text{Maximum Discharge Current} = \frac{\sum P_{MAX\ COMP}}{V_{MIN}} = \frac{99.835\ W}{10\ V} = 9.985A = 10A$$

Thus, we need batteries that can handle 10A discharge current. To calculate the capacity, we use the following equation. We found the average A by adding the average current of the components together. We wanted to power the robot for at least 1 hour.

$$Capacity = A_{AVG} * h * 1000 = 1.2465 * 1 * 1000 = 1246.5mAh$$

3.4.3 Battery Options

Between the three batteries: NiMH, Li-Ion, Li-Poly, we examined which one was the best. Since we will be requiring a lot of energy, Lithium based cells are a better option than NiMH.

Possible Li-Ion batteries that we can consider are the INR18650-35E 3500mAh batteries made by Samsung. The plan was to have 6 batteries, with 3 in series and those two rows in parallel. The table below shows the specifications of the batteries in this configuration. The batteries meet all of the criteria set in Table 7. The batteries are also less than \$50 in total.

Table 7: INR18650 Li-Ion Batteries Specifications

Specification	Detail
Dimensions	67 x 57 x 38 mm
V_{NOM}	10.8V (3.6V _{NOM} 4.2V _{MAX} Single Cell)
V_{MIN}	9V
Max Discharge Current	16A
Volume	145 cm ³
Weight	300g
Capacity	7000mAh

The Li-Poly batteries that we found are the LP616594 4700mAh batteries. The configuration was 3 batteries in series. Table 8 shows the battery specifications. The Li-Poly batteries in this configuration do not meet the maximum discharge current that we specified, but it should be sufficient because the robot will not be running at maximum current. Otherwise, the Li-Poly batteries meet the rest of the requirements.

Table 8: LP616594 Li-Poly Batteries Specifications

Specification	Detail
Dimensions	94 x 65 x 19 mm
V_{NOM}	10.8V
V_{MIN}	9V
Max Discharge Current	9.5A
Volume	113 cm ³
Weight	236g
Capacity	4700mAh

Out of the three batteries we considered, the Li-Ion batteries seem to be the best option. They met all the requirements set out for the robot.

3.4.4 Recharging

The batteries needed to be recharged. We can have two methods to recharge the batteries. The first method would be to use a DC power supply to fully charge the battery pack. With a DC power supply, we can limit the current flowing into the battery from the power supply. Figure 11 showcases a Li-Poly battery charging from a power supply. While it is a different battery from what we will be using, the concept is still the same. Another method would be to use a separate battery such as a 12V lead acid battery and use a Buck/Boost converter to step the voltage down so the batteries can charge properly. Figure 12 shows a possible configuration.



Figure 11: Battery Charging from a DC Power Supply

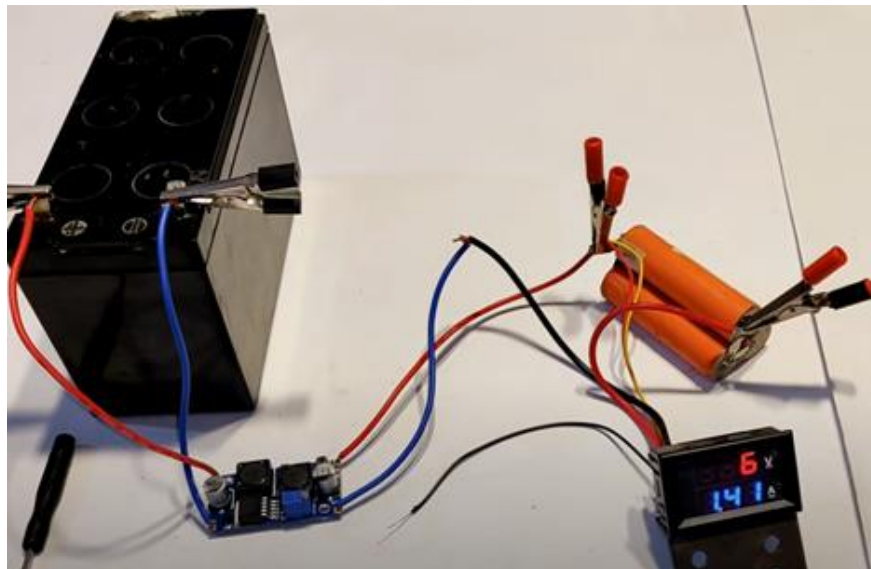


Figure 12: Battery Pack Charging with Buck/Boost Converter

Another option would be to design an AC to DC converter so that we can plug an AC adapter into the robot. The input AC adapter will be converted to DC and stepped down to a safe voltage so the batteries can safely charge.

3.4.5 Photovoltaic Cells

As a secondary goal, we hoped to implement photovoltaic (PV) cells, or solar panels that can passively regenerate the batteries while Trash-E is operating. Things we had consider for the photovoltaic cells were the size of the panels as compared to the size of the robot, the placement of the panels, and the recharge rate. Depending on the size of Trash-E, the implementation of photovoltaic cells may not be worthwhile, as they may not be able to provide any substantial energy at all.

The sun radiates photons, which contain varying energy, which varies based on the wavelength of sunlight. When the photons hit a PV cell, the cell attempts to absorb the photons, although all the photons are not fully absorbed, as some are reflected off. Once there is enough photon energy absorbed within the semiconductor material, the electrons inside the cell are free to move. When enough electrons have moved to the front of the PV cell, a voltage potential will have been created. Once the cell is connected to a load, such as a light bulb, electricity will flow. We can use this to help charge the battery while it is operating outside. We will have to make sure that the charge from the battery does not flow to the solar cells while the cells are not charging.

Photovoltaics have many advantages and disadvantages. PV cells are good for the environment because their energy generation releases no carbon emissions. This is beneficial for our robot because Trash-E will be able to work outdoors. Thus, we hope to be able to run Trash-E for longer periods of time when outside on a sunny day. Because PV cells have no mechanical parts, there will be little to no maintenance regarding them once they have been implemented.

3.4.5.1 Monocrystalline Silicon Cell

Monocrystalline Silicon Cells are generally more efficient than other types of PV cells. While they are more efficient, they also are much more expensive. As a result, we did not plan on using these types of cells, but they were considered in the preliminary stages. If we find low-cost Monocrystalline PV cells, we may consider using them.

3.4.5.2 Polycrystalline Silicon Cell

Polycrystalline Silicon Cells are the cells that we were most likely to use. They are cheap and abundant since they are the most popular types of photovoltaic cells. Section 6.2.9 further covers the types of solar cells that we will consider using in our robot design.

3.4.5.3 Thin Film Cells

It was highly unlikely that we will use Thin Film Cells because the flexibility and thinness of these cells are not necessary for our application. Furthermore, since they are generally less efficient than the two previous cells, it is unhelpful to our design. In addition to the previous, some thin cells contain rare or toxic elements. These elements would be detrimental to our design because they would add unnecessary dangers to our robot. To make up for these dangers we may need to add potting to our circuits, which would add too much complexity that we do not need.

3.4.5.4 Miscellaneous Cells

These cells such as high efficiency cells were out of the scope of this project because they are either exceptionally expensive or are still a new technology that is not fully developed.

3.4.6 Voltage Regulator

3.4.6.1 Linear Voltage Regulators

Each component within Trash-E requires certain voltages to operate correctly. By supplying 12V, each component should be able to be powered, either at 12V or stepped down to the required voltage. Certain components such as the ultrasonic sensor or the motors controlled by the Arduino will be taking power from the Arduino, thus it will not need to be stepped down. However, the Jetson Nano used 5V to power, thus the 12V needed to be stepped down. The purpose of a voltage regulator was to keep a constant voltage output regardless of input voltage or current draw from the load. There are two types of voltage regulators, linear and switching, each with their own benefits and drawbacks. A linear regulator, which can be found in Figure 13 below, is a simple circuit with low noise and few parts necessary externally. It uses the control circuit to monitor and change the output voltage. Linear voltage regulators are slower at changing the output voltage if there is a large change in the input voltage because it is using a feedback loop to control the output voltage. In our application there should not be huge drops or rises in our input voltage, so this should not be a problem. The linear voltage regulator often has poor efficiency between the input and output voltage conversion. The linear voltage regulator can also only be used as a buck converter. This is not an issue in our use case, as we only want to step down our voltage. It can also get hot easily, so temperature must be considered. This is because when stepping down voltage, the excess power must go somewhere, and thus the voltage regulator expels it as heat. To calculate the power loss, the following equation is used. Figure 13 shows a basic Linear Voltage Regulator.

$$P = (V_{IN} - V_{OUT}) * I_{LOAD}$$

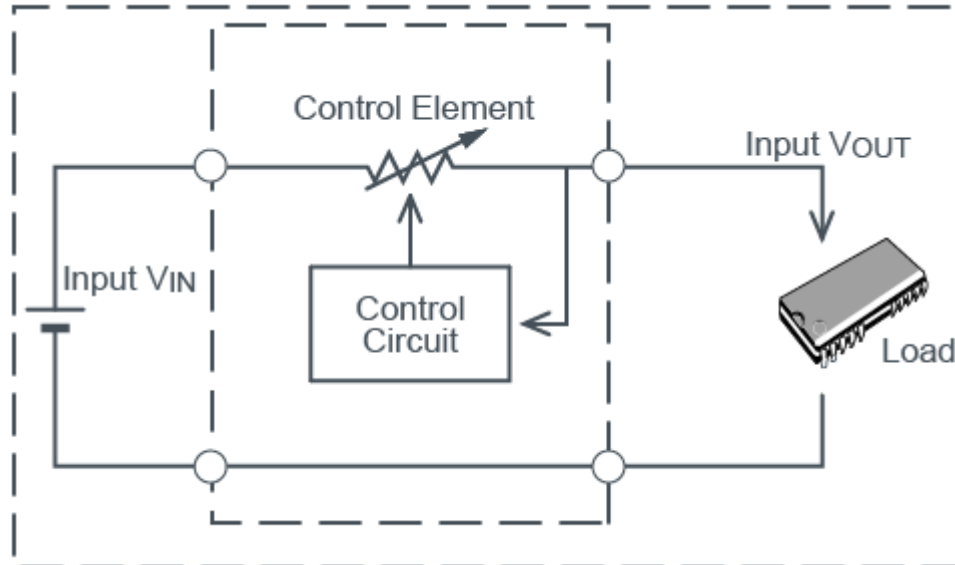


Figure 13: Linear Voltage Regulator (Courtesy Rohm)

3.4.6.1.1 Standard Voltage Regulator

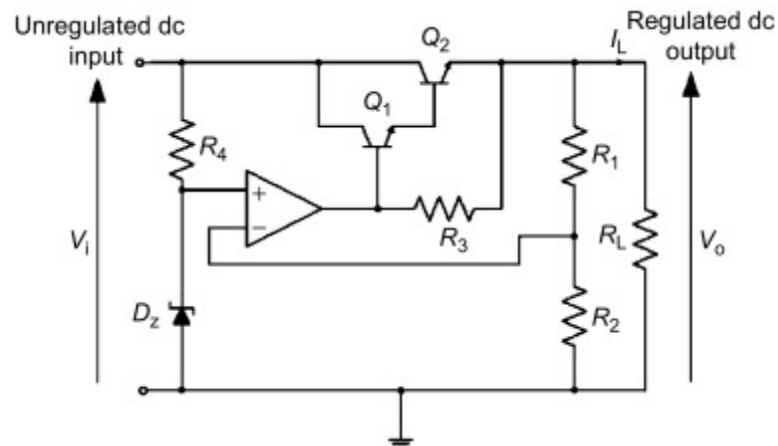


Figure 14: Standard Voltage Regulator

The standard voltage regulator in Figure 14, was a basic configuration using a Darlington pair of transistors. The Standard voltage regulator circuit could be replaced with an Integrated Circuit of a regulator. These options are later discussed in Section 6.28. Standard voltage regulators can have large voltage drops depending on the device specifications. Voltage dropouts for a standard regulator can vary between 1V and 2V. In cases where a regulator needs to drop from 120V to 12V, 1V voltage dropout is not a big issue. However, in a case where a voltage needs to be converted from 3.6V to 3.3V, the voltage dropout is a third of the input voltage. The output voltage must be less than the input voltage minus the dropout voltage, otherwise the regulator will be unable to function.

$$V_{OUT} < V_{IN} - V_{DROPOUT}$$

3.4.6.1.2 Low Dropout Regulator

Low Dropout Regulators (LDO) regulators are useful when needing small voltage drop amounts. Some LDO regulators can have drops of 100mV, this is useful for cases between 3.3V and 3V. Since we are dropping down 12V to 5V, and possibly 12V to 3.3V or 5V to 3.3V, we will not be needing a LDO regulator.

3.4.6.2 Switching Voltage Regulators

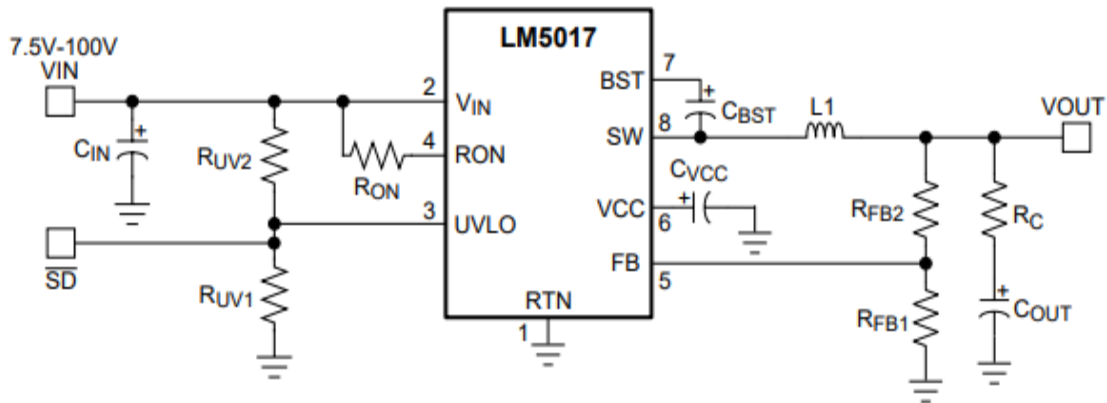


Figure 15: Typical Switching Voltage Regulator Circuit of a LM5017

A switching voltage regulator is a type of regulator that allows for both buck and boost of a voltage. This is possible by having a switching element that the circuit uses to change input power into a pulse. The voltage is smoothed out with the use of different capacitors and inductors, FETs, and other components. Figure 15 shows a typical circuit of the switching voltage regulator. It is more complex than the linear voltage regulator in Figure 13. The output power is set to the desired voltage, and once it reaches it, the switch is turned off. When the switch is off, there is no input power being supplied. By continuously doing this process at high speeds, the efficiency between the output voltage and input voltage is much higher than a linear regulator, which also results in lower temperatures because there is not as much power being dissipated as heat. Although the switching voltage regulator has these advantages, there are also several disadvantages such as the regulator being much more complex and requiring more external parts.

Within Trash-E we will not be requiring precise voltage drops with tight margins, we also want to keep costs down. As a result, we ended up using a Standard Linear Voltage Regulator circuit, as that was sufficient. In the final design, we used the lm1084-ADJ linear regulators made by TI.

3.5 Locomotion and Mapping

3.5.1 Different Types of Movement

There are a few options to how we approached maneuvering Trash-E around on its own. As a recap, Trash-E needs to be able to maneuver any area to find trash, assuming it was physically capable of traversing the area given its physical limitations. Given Trash-E's physical limitations, Trash-E should be able to maneuver on any flat surface. However, this area or environment that Trash-E was placed into was not always familiar and can be entirely new every time. So, there were certain methods in our design that allow Trash-E to maneuver around an unknown environment and accomplish its tasks while being autonomous or without manual control via a controller.

One way of detecting obstacles in the robot's path was using bumper sensors. These are the least useful in terms of Trash-E since we have no way of determining where it is, where it's been, and where it will go. It is essentially random movement that is up to chance and the layout of the environment and can't guarantee every spot will be reached. The robot can also get stuck in an area depending on the configuration of the environment and the movement capabilities of the robot itself.

Facing some of the same challenges as the bumper sensors, this approach involves using ultrasonic sensors on all sides of Trash-E that will detect incoming obstacles for it to keep moving and avoid obstacles. If the sensors don't detect anything close to the chassis, then the robot would continue to move forward until a sensor has detected an obstacle that is too close. If the sensor detects an object in front, Trash-E will maneuver to the right or left depending if the sensors on those sides do not detect any obstacles. In the case where Trash-E leads itself into a dead end in which case the sensors on the front, left and right are triggered, then Trash-E would reverse out of that spot. However, there are a couple of downsides to this approach. There is a chance that Trash-E could get stuck in a loop and not be able to explore the whole area that we want it to clean. Another downside is that the movement would be completely random. This is bad because it would lead to a lot of repeated work since Trash-E would not know if it had traversed the area, it is currently traversing. This could lead to a much longer runtime to complete its cleaning task.

This approach we considered is a very simple approach to this problem but has some major downfalls. This approach involves Trash-E solely being guided by the computer vision algorithm and the objects that the camera will detect. In any case, when the robot detects an object of interest then it will approach it accordingly and then pick it up. Once it completed its task, it will then spin until it detects another object of interest. This method would have worked fine if all the objects were always right next to each other, however, this will usually not be the case. If there were objects scattered across a room, there is a chance that Trash-E would not

be able to detect an object either because it is too far away or there is an obstacle obstructing the view of that object.

Another way was with a camera. Visual detection is better than the bumpers since we can gather points of interest in the surrounding environment. It also allows us to keep track of the robot's position by using landmarks. The issue with visual detection is it is more sensitive to light and can throw off the calculations or blind the robot entirely.

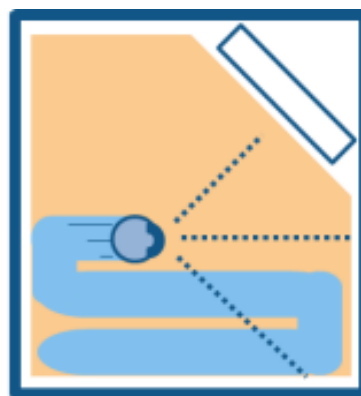
One way to fix this issue was using a lidar sensor. These sensors are similar to an ultrasonic sensor except it uses light waves. "A typical lidar sensor emits pulsated light waves into the surrounding environment. These pulses bounce off surrounding objects and return to the sensor. The sensor uses the time it took for each pulse to return to the sensor to calculate the distance it traveled". [https://velodynelidar.com/what-is-lidar/] Sending out these pulses many times per second in a complete circle around the sensor achieves a real-time map of the immediate environment. This information can then be processed by an algorithm that makes a graphical representation of the surrounding area. lidar is a very valuable technology when generating detailed maps of the environment around a robot.

3.5.2 Simultaneous Localization and Mapping (SLAM)

The third approach was a fully autonomous approach and method that utilizes both localization and mapping which is called simultaneous localization and mapping. SLAM is a method that allows autonomous robots and vehicles to build a map of its surroundings and localize itself within that map at the same time. With this method an autonomous robot can use these algorithms to map out an unknown environment. At the same time, the robot is able to know where it is in the environment. With this map information, robots can use path planning and obstacle avoidance.



Without SLAM:
Cleaning a room randomly.



With SLAM:
Cleaning while understanding the room's layout.

Figure 16: Comparison between no SLAM and SLAM (Courtesy of MathWorks)

A popular example of simultaneous localization and mapping in the real world was the home robot vacuum. These robot vacuums can navigate the floor of a home autonomously and figure out a path that can ensure it will vacuum every part of the floor. At the same time, these robot vacuums are in new environments and can create a map of the floorplan so that the path taken is the most efficient and avoids any obstacles or obstructions. In Figure 16, we see the difference SLAM makes for the robot vacuums. This is very similar to the comparison of approach two and three for Trash-E. With our second approach we would move very randomly while in the third approach utilizing SLAM, our movement would be more defined and efficient for time and battery life.

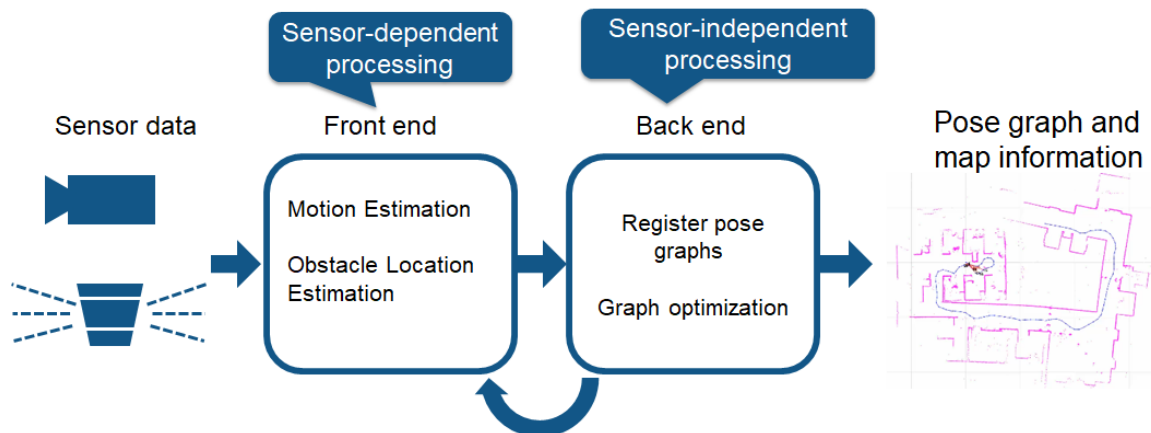


Figure 17: Flow of SLAM process (Courtesy of MathWorks)

There are two things that needed to be done to achieve simultaneous localization and mapping. This includes the front end and back-end processing. The front-end processing is where sensor signal processing is done. The back-end processing is where pose-graph optimization is done. In Figure 17 we can see the flow of the SLAM process, which goes from taking in sensor data to frontend processing, backend processing, and finally we get our map. SLAM uses different methods of gathering data such as from a camera, lidar device, odometer, wheel revolutions, or other imaging sensors to determine the amount of movement needed and its location in the map which is called localization.

There are also different versions of SLAM that have been developed. Two that we considered for Trash-E were visual SLAM and LiDAR SLAM. Visual SLAM primarily uses images acquired from cameras and other image sensors. The cameras or sensors that are used can range from complex to simple and can be very expensive or inexpensive depending on the needs of accuracy. The next popular SLAM is LiDAR SLAM which uses a LiDAR device that is a laser sensor that have a 360 view. This method of SLAM is significantly more precise than visual SLAM.

For Trash-E it was best to use LiDAR SLAM as this method is much more accurate and efficient. Therefore, we would need to place a LiDAR device on Trash-E that

will use 360 laser scans to detect the area around it. Trash-E will maneuver along an area and simultaneously create the map of it and localizing itself as well within that map to plan its path.

We had a couple of options for implementing SLAM on Trash-E. One was to use MATLAB's robotic systems tools that offer capabilities for implementing SLAM. MATLAB offers toolkits for implementing the SLAM onto Trash-E as well as for what comes after which is path planning for autonomous driving over that map. The downside with MATLAB however was that it is not free and in fact very expensive. We would need to spend approximately hundreds to use the toolkits from MATLAB on top of paying to use MATLAB in the first place. This is out of our budget range and doesn't make sense for us to use. However, we could use MATLAB on the computers located on UCF and work on SLAM at the UCF facilities. But in terms of personal use and working on the project at home, MATLAB is not the most feasible option.

Another option was to use an open-source SLAM library called BreezySLAM developed by Professor Simon Levy from Washington and Lee University. This package library gives us the ability to implement a LiDAR based SLAM on our robot at no cost and with Python or C++. Compared to using MATLAB, using this open-source library makes sense financially and allows us to develop at home. However, being that it is an open-source package created by an individual, it might not have all the features that are available on MATLAB for SLAM.

The final option was to use ROS (Robot Operating System). This software package offers an open-source library for general robotics in Python. Code can be written in C++ or Python. We decided to choose this option since there is LiDAR capabilities and we can integrate it with our Python script easily.

3.5.2.1 SLAM Implementation

When we initially place Trash-E in a new location, we needed to map the area that it will autonomously drive through. This is where SLAM would come into play. Trash-E will have a LiDAR on its body that will do a scan of the objects surrounding it to sense the distances and angles of obstacles around it. If needed, we can use the wheels on the body to determine the distance that has been driven by Trash-E and how it has turned. We can do this by calculating the rotations of our continuous servo motors that control the wheels with the circumference of them to get the distance traveled.

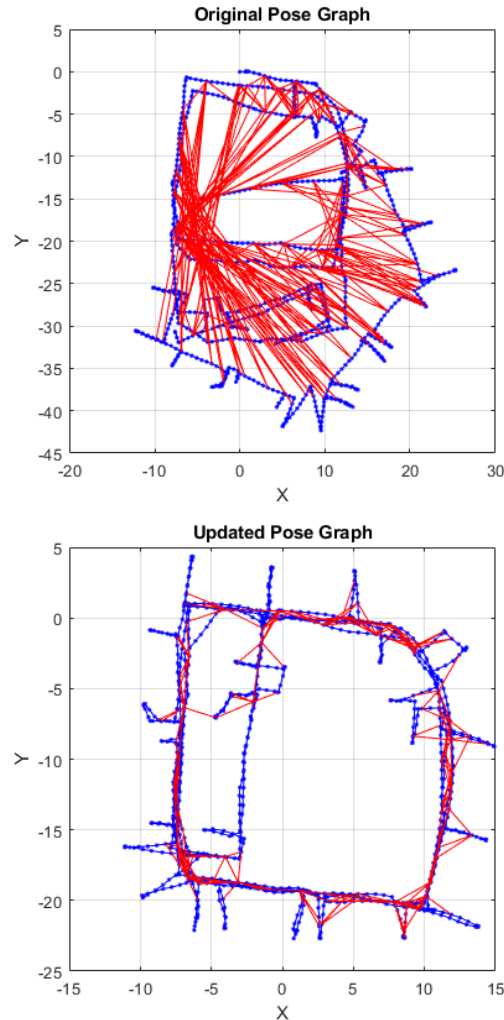


Figure 18: Example of pose graph optimization (Courtesy of MathWorks)

During this process we created our map with pose graph optimization. Pose graph optimization helps fix the errors of positions and distances when using SLAM to create the map. Without pose graph optimization our map would likely look very inaccurate and have a lot of errors when relying solely on the sensors. Pose graph optimization will use nodes of poses (positions on the map) and constraints between the nodes which we can call edges. At some point in the process of SLAM, we will detect the same features once again with the sensors. This means that we can likely close the loop of our map traversal between these two nodes. Our pose graph optimization algorithm will pull these two nodes as close as possible until the features they detected match. During this process all other nodes' edges in the map experience "tension" and are pulled simultaneously as the original two nodes are being pulled together. After this is done the errors in the map have been mostly corrected and the is much more accurate than before the pose graph optimization. An example of this is optimization is shown in Figure 18 where you can see the original map had many errors and very uncertain but after pose graph optimization the map is a lot cleaner and has less errors.

Once we have the map for our area, we can make a binary occupancy grid out of it which we can use to determine an optimal path to traverse this area/grid while avoiding any obstacles. For this we use many shortest path algorithms and search algorithms. One that we can use is the A* search algorithm which is more efficient than normal graph search algorithms. We also use RRT and RRT* (Rapidly Exploring Random Trees) algorithms which are sampling based search methods.

With the map and path planning made Trash-E continuously and autonomously navigated the area which it has mapped using its path planning on the grid. While navigating autonomously through the area, our computer vision looks for the trash objects in its view. If while traversing Trash-E detects a trash object, priority is given to the computer vision algorithm so that it can control the movement of it towards that piece of trash so that it can complete its task of picking it up and placing it in its bin. After it has been placed in it's been, control is given back, and it continuously keeps searching the area for more trash.

3.5.3 Visual vs. Lidar

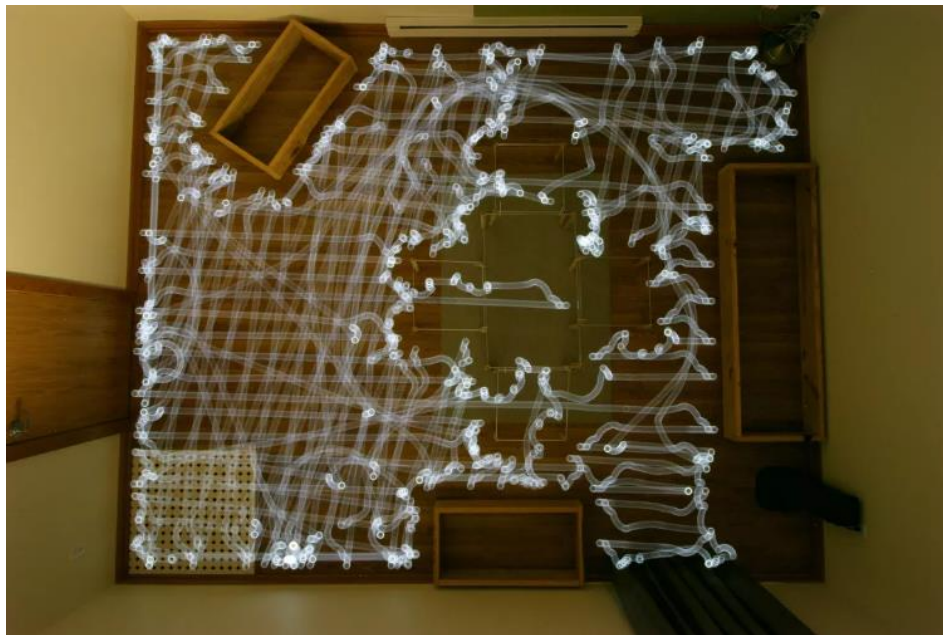


Figure 19: Robot Path Using VLSAM. (courtesy of Gianmarco Chumbe/CNET)

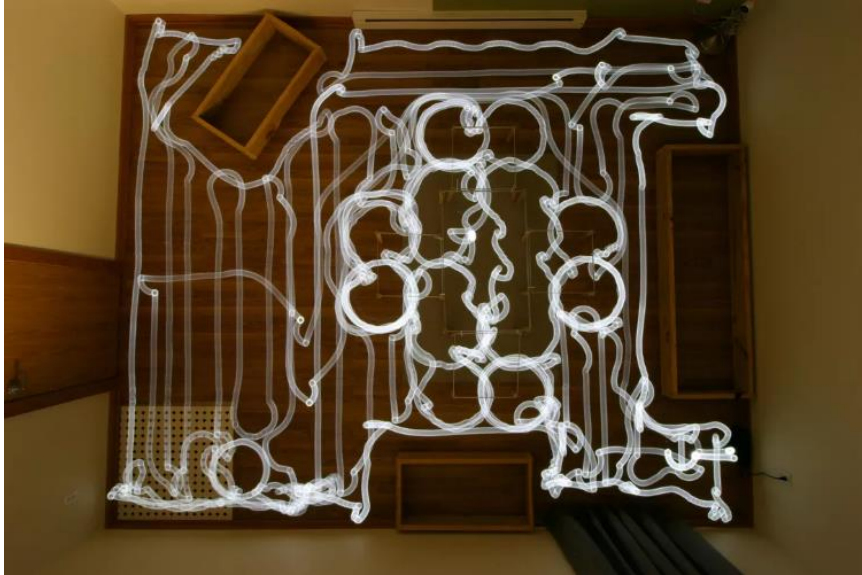


Figure 20: Robot Path Using SLAM with a Lidar Sensor (courtesy of Gianmarco Chumbe/CNET)

While utilizing visual locomotion can still complete the task, it is not very efficient as it has a lot of retracing, which can be seen above in Figure 19, due to only collecting points from in front of the robot at any given time. Lidar allowed the same situation to be optimized and be more efficient which is why we wanted to utilize this technology. The highest efficiency and least amount of extra movement is necessary for Trash-E to keep power consumption to a minimum. When compared to VSLAM, Figure 20 shows significantly less retracing and a more optimized path.

3.5.4 Lidar Options

Table 9: Lidar Sensors

	Slamtec A1M8	MakerFocus YDLIDAR X2L	EAI YDLIDAR X4	getSurreal XV Lidar	Hokuyo URG-04LX-UG01
Price	\$100	\$70	\$80	\$150	\$975
Availability	In Stock	In Stock	In Stock	In Stock	In Stock
Weight	370g	126g	180g	370g	160g
Input Power	5V	5V	5V	5V	5V
Power Consumption	500mA	500mA	500mA	500mA	500mA
Laser Safety	Class 1	Class 1	Class 1	Class 1	Class 1
Scan Radius	12m	8m	10m	6m	5.6m
Laser Range Scanning	360°	360°	360°	360°	240°
Accuracy	+/- 3.5%	+/- 3.5%	+/- 3.5%	+/- 3.5%	+/- 3%

Scan Rate	5.5 Hz (10Hz Max)	6 Hz	7 Hz	5 Hz	10 Hz
Ambient Illuminance	No Informatio n	No Information	No Informatio n	No Informatio n	Florescen t Max: 6,000 Lux
Communicatio n	UART/US B	UART	UART	UART	Mini USB 2.0

3.5.4.1 Price

Since we were choosing to do a lidar based SLAM, we purchased a sensor to gather the necessary data. Due to the nature of the technology, lidar can become quite expensive very quickly. Our first choice of sensor was the Hokuyo URG-04LX-UG01 due to it being tested by others using the Python library BreezySLAM that we will be using and knowing that there are no compatibility issues. The issue with this sensor for our use case is the price of \$975 which is over double our total budget. Even though this is a very good sensor for education and researchers, we won't be able to use it. Instead, we did some searching for lower end, or hobbyist, sensors. The prices range from \$70-\$150 which are more realistic for our robot.

3.5.4.2 Weight

Weight was a large consideration for our robot when choosing hardware parts. The more the robot weighed, the more energy we need to expend to move it which depletes our batteries quicker. The goal for our sensor was to be as compact and lightweight as possible. The MakerFocus YDLIDAR X2L boasts its impressive weight of only 126g, while the Slamtec A1M8 is more than double at 370g, along with the getSurreal XVLidar. We wanted to keep our sensor below 200g as to minimize the total weight of our robot to prolong the up time we can pick up trash.

3.5.4.3 Power Specifications

Since lidar sensors spin around to send the laser signals, they need something that can accomplish this task. All the sensors we looked at utilize DC motors. Each DC motor has an input power of 5V and consumes a maximum current of 500mA while the motor is running. We will be able to supply the required 5V using the voltage regulator boards that were designed for Trash-E. The idle currents varied between them in the range of 200-300mA, but we used the worst case where the motor is running all the time to spin the sensor. This gave us an accurate measurement of how long the robot can run at the minimum.

3.5.4.4 Laser Safety

Working with lasers of any kind requires certain safety specifications. Different safety procedures and PPE are required with different classes of laser safety.

According to the Environment, Health and Safety, Class 1 lasers are “eye-safe under all operating conditions. A Class 1 laser is safe for use under all reasonably anticipated conditions of use; in other words, it is not expected that the MPE can be exceeded”. [X] They also describe a Class 1 Product as “a laser product or device which may include lasers of a higher class whose beams are confined within a suitable enclosure so that access to laser radiation is physically prevented. Such products do not require a laser warning label on the exterior”. [X] The lidar sensors we chose utilize an enclosure to ensure the laser radiation is physically prevented, making them a Class 1 Product.

3.5.4.5 Scan Radius

This specification of the sensor indicates how far from the center of the sensor the laser can accurately measure. The lowest radius in our price range was 6m. Given our robot was used indoors, this minimum radius was sufficient. At any given point we were be within 6m of a wall for our robot to locate an obstacle of some sort, whether it be a wall or an unknown object.

3.5.4.6 Laser Range Scanning

Creating a full map of the environment was essential for Trash-E to traverse efficiently. If the map was not generated completely, there would be discrepancies with the path it should take since the information was not there. To generate a full and complete map, we needed the lidar sensor to be able to fully spin 360°. This will omit the possibility that obstacles weren’t detected due to the robot having not faced that direction. All the sensors we considered were capable of rotating 360°.

3.5.4.7 Accuracy

Measurements of the environment also needed to be accurate so the robot may traverse efficiently. Out of all the sensors we investigated, the lowest accuracy range is +/- 3% and the average was +/- 3.5%. With the 3% variance being a higher costing sensor, we must stick with the 3.5% variance. This did not affect our real-time measurements greatly and was sufficient for the environment mapping since the robot should not be going that close to walls and obstacles to begin with.

3.5.4.8 Scan Rate

The scan rate determines how many full scans of the environment can be completed during one second. Given that our robot will be moving relatively slow to other autonomous mobile robots, like quadcopters, the scan rate was not too important for our decision making. Since we also used the Jetson Nano for our computation, the extra information that we would gain from having a higher scan rate would not have been properly utilized like it would if we were using a higher-powered CPU.

3.5.4.9 Ambient Illuminance

This specification was a very important one, although unfortunately most of the sensors did not have information regarding it on the manufacturer's websites or on their datasheets. Ambient illuminance tells us the brightness conditions that the lasers will work in without error for different lighting types. The only option that tells us this information is the Hokuyo URG-04LX-UG01 which was no longer under consideration. For this sensor, the maximum florescent max is 6,000 lux. According to Green Business Light UK[1], "the lux of artificial indoor lighting, however, is typically 1,000 lux or below...". They also point out that the lux of direct sunlight is a minimum of 32,000 lux and the minimum of ambient daylight is 10,000 lux. Given this information, even if the maximum ambient illuminance for the sensors that had no information is only 25% of the Hokuyo sensor, the robot will have a maximum of 1,500 lux and is still above the 1,000 typical lux. Sunlight is the only cause for concern when it comes to light. The robot will still be able to successfully complete the environment map without error if the shades were drawn to keep the sun out.

3.5.4.10 Communication

All sensors utilize UART at the minimum which was sufficient for one way communication from the sensor to the Jetson Nano. At least one sensor utilized USB but this is not necessary to use as there was enough GPIO pins for us to use with the Jetson Nano.

3.5.4.11 Conclusion

Given all the considerations above, we chose the MakerFocus YDLIDAR X2L. The price point kept us within our budget and the weight kept power usage low. The other features that this sensor is lower in than the competitors are negligible for our use case.

4.0 Constraints

4.1 Description

This section covers the design constraints for Trash-E: Autonomous Litter Picker Upper, and the associated standards with its design.

4.2 Economic

As the whole team consists of a group of students, it is to be expected that our capacity to spend resources is relatively small. Therefore, in order to complete the design in a cost efficient and on time manner the cost was no more than four hundred dollars. The four hundred dollars allocated was split amongst the four members of the group, which allows the burden of purchasing parts to be evenly distributed. Being mindful of what each member has spent will ensure that no conflict is had between the members. To reinforce this, a bill of materials with who funded each part was implemented.

4.3 Environmental

Terrain was immediately taken into consideration when brainstorming ideas for the robot. Having a robot operating in several different kinds of terrains would take away from our focus and consume more of our time and resources. For Trash-E, we have it operate on flat terrain such as tile floors, carpet, cement, or anything similar. With two wheels and a swivel wheel in a triangular configuration, this would not allow us to travel in any rough terrain. Implementing a design for rough terrain would be impractical for its intended use as well.

4.4 Social

The design of the robot was mainly built around functionality as it is not intended to be used for anything other than picking up trash/litter. Appearance was a secondary focus when designing the robot. Therefore, in exchange for soft and aesthetic looks we could focus on functionality for the sake of development. Take, for example, iRobot®'s Roomba® with its sleek and functional design that allows it to do its job while not being an eyesore.

4.5 Sustainability

In the case of sustainability, this robot would be fully sustainable as it will be 3D printed with polyethylene terephthalate glycol (PETG). The production of plastic is at an all-time high in the twenty-first century damaging the Oceans and polluting the environment around us. Many 3D plastics can be easily recycled back into their unused filament string form leading to less plastic ending up in the environment. Manufacturing metal parts specific to the project may lead to further trash in the

garbage heap when the robot is no longer operating or is not needed. To combat the issue of pollution, a shredder designed for cutting plastic along with a filament extruder should allow for the reuse of this plastic material.

4.6 Ethical

Concerning the job security of janitors and people whose job is to clean up after public events or parties. This robot is designed in such a way that it should not put people in this occupation at risk as it was never intended to perform the entirety of their jobs in the first place.

4.7 Time

For this project we had two semesters to design, build, test, and present. This greatly impacts the sophistication of our robot as we had to make sure we could implement our idea in about 3 to 4 months. This also affected the number of secondary and stretch features we could achieve. If something went wrong or took longer than expected, we had to re-evaluate what we could get done in the remaining time. To combat this, work was done in the first semester in making sure everything is researched and planned out to allow for a smooth development process. Documentation as well as research and some testing were done by December 2021. Having most of the research, testing, and documentation done by then gave enough time to allow for slight mistakes for the completion of the project by April 2022.

4.8 Safety

This robot is to autonomously operate in areas where human traffic is taking place; there must be multiple fail-safes as well as safety practices to prevent harm during operation. In the case of collision, Trash-E is plastic and light weight. The max voltage supplied to the motors is capped to ensure that the robot moves at a specific low speed. The gripper arm is designed in a way to avoid sharp edges and protrusion which also applies for the chassis design. With the use of ultrasonic, collision detection was implemented to prevent the collision case mentioned previously. Electronics for the robot are all grounded and hidden from direct contact from users. Stored inside the chassis, the electronics will be contained in a separate compartment to give easy access to developers and to protect users from the robot. Finally, an option for manual shutdown was included in the case of an unforeseen action during runtime or if the robot is found to not be operating properly. Further elaboration on safety can be viewed in the standards section of the document.

4.9 Manufacturing

Since this robot is designed to operate indoors, operation outdoors is outside of the project scope. Furthermore, manufacturing costs were saved since we do not need to deal with outdoor elements. Despite this, the robot is still reliable and durable enough to operate for a single hour. We were limited to the following methods for making parts for our robot:

1. 3D Printing: Using the 3D printer in the Innovation Lab or using a printer that a group member owns. Manufacturing of parts this way allows for greater creativity and less time for production. Using Autodesk can make for short work of chassis designs and moving components for the robot. This option is also relatively cheap if we provide our own plastic or use the schools.
2. Purchasing wood: Going to a home improvement store and getting the required wood to build into what we need. We would also need access to a workshop that is provided by UCF. This would be one of the cheapest options, but it also requires the skills for woodwork as well as being able to design the specific measurements for schematics. Replacing parts would be cheap but would take some time, like 3D printing, to reproduce.
3. Purchasing metal: This approach would have been the most expensive and difficult due to not having the correct tools. Similarly, we would be able to use the workshop provided by UCF, but working with metal is the hardest and, potentially, the most dangerous option. Replacing broken components on the chassis would take a great amount of time to do, but the likelihood of it happening in the first place would be low. Weight also must be taken into consideration, as metal would be the heaviest out of all of the three options.

5.0 Standards

5.1 Lithium-Ion Battery Safety Standards

Several international/universal industry practices are used in upholding standards in lithium-ion battery safety, this also applies to regular lead acid batteries. Organizations such as IEEE, ICE, IECCEE, U.S based OSHA, NRT, ANSI, ISO/IEC, UN/DOT, UL, and many more have thoroughly created many standards for the regulation of lithium-ion batteries; all have the facilities and equipment to do so. For this project we followed IEC 62133 (International Electrotechnical Commission) as they create non-profit standards internationally. The scope in this standard states that “IEC 62133 specifies requirements and tests for the safe operation of portable sealed secondary lithium cells and batteries containing non-acid electrolyte, under intended use and reasonably foreseeable misuse” (IEC). Several testing procedures, as well as maintenance requirements are recommended by this standard.

5.2 Standard SystemC ® Language Reference Manual Standard

The standard *IEEE 1666 2012* lays out clear definitions on how to go about with syntax and proper procedure on developing certain aspects of C language including C++. Seeing as this project involves heavy usage of both software and hardware as well as the communication between the two, for example, serial data transmission from the Nvidia Jetson to the microcontroller that interprets data sent and acts out instructions. In other words, it provides an extensive list of core language class definitions, predefined channel class definitions, system C data types, system C utilities, terminology, and simulation semantics. All of those listed previously are in use for the development and documentation of our project.

As a side note, this standard uses the words “shall” [16], “should” [16], “may” [16], and “can” [16] that carry their own significance. Shall being the most important meaning that what is requested in the standard is mandatory. Should is mainly a recommendation and nothing more. Finally, can is used to imply that something is possible, or within the scope of operation. For example, shall is used when dealing with function definition and side effects in section 3.3.2 in the standard. The use of such is explicitly used throughout the standard like in this case: “Such functions shall not have any side-effects that would contradict the behavior explicitly mandated by this standard.” [16]. They clearly define what should be done throughout the standard with similar cases to the previous quote. Not following the advice indicated by the word “shall” [16] will typically cause issues within the C system you are designing.

Using this standard will “provide a C++-based standard for designers and architects who need to address complex systems that are a hybrid between hardware and software” [16]. Therefore, it increased our options as well as

provided us with guidance for the solutions to our complex software and hardware systems.

5.3 Software and Systems Engineering – Software Testing

ISO/IEC/IEEE 29119 is a standard that takes multiple other standards and compiles them into one coherent standard. Developed by the International Organization for Standardization and the International Electrotechnical Commission, they attempt to make a worldwide standardization that could be adapted to work internationally.

Mainly, the concepts applied from this standard are the testing process, as well as the testing techniques. Outlining the testing process consists of splitting up and organizing several processes that need to be done. In our case, with four team members, we organized the software and hardware processes by assigning specific tasks to each member.

Test management was undoubtedly the largest section as it is paramount that proper testing procedures are taken to ensure quality code and operation. First, coming up with a plan on how to execute a testing plan until its completion; this can consist of several techniques to accomplish this with liberty to adjust plans to meet design goals. For example, the test plan is created, testing starts and is monitored while providing needed updates, and finally test completion. In the test completion phase, plans for further improvements or maintenance can be made, but it also serves as a final check if guidelines were followed before the release of the product.

Testing techniques include specification-based testing, structure-based testing, as well as experience-based testing. Each comes with its own approach to the testing process, and with their respective advantages and disadvantages. Specification-based testing is more about using previous information gathered such as documentation gathered from part manufacturers about specifications for operation. Furthermore, there are several ways to implement structure-based testing as there are many sources to pull from. Structure-based testing can pull from outside datasets, sample codes, models, and documentation to achieve design goals during testing; there are several ways to go about this. Experience-based testing relies on the testers previously gathered experience and knowledge. The tester in this case could create a model to aid in the structuring and development of code and the software, though it is slightly limited due to the fact of not relying on outside sources. This technique is also not as predictable since each developer/tester has varying skills in different fields. In this case, direct debugging and testing of code is utilized to develop features which would take more time, but it can lead to creative solutions. Errors encountered will also depend on the tester's ability to predict the operation of written code as well as the ability to create unit tests to cover many different input cases.

Conforming to the standard as well as its shown practices will make it so that the structure of the software we developed applies to this standard. Although *ISO/IEC/IEEE 29119* provides many options for test management and testing techniques, many of them were excluded from this robot's development as several of them provide enough guidance for a small development group. The testing techniques as well as the organizational procedures were taken into consideration. With the combination of those two principles, proper planning, and adequate testing of each testing block ensured proper adherence to the standard.

5.4 Programming language – C Standard

International standard *ISO/IEC 9899* intends to specify: “the representation of C programs”, “the syntax and constraints of the C language”, “the semantic rules for interpreting C programs”, “the representation of input data to be processed by C programs”, “the representation of output data produced by C programs”, and “the restriction and limits imposed by a conforming implementation of C” [19]. This is what we used for much of the embedded software design for the project.

Within the standard document, the language section includes notation, concepts, conversion, lexical elements, (constant) expressions, declarations, statements and blocks, external definitions, preprocessing directives, and future language directions. Typical concepts for programming languages like syntax, data type identifiers, and more are included under annex A. Following this is further information on several libraries that aid with specific operations, for example, the `math.h` or `string.h` libraries that were very useful during development. Before the creation of the C standards, a large amount of functionality may have been difficult to know about.

Previously mentioned aspects of the standard aided in the development of drivers on the embedded side of Trash-E: Autonomous Litter Picker Upper. Communication between the Nvidia Jetson and the microcontroller used a serial communication protocol dealing with parsing through strings. For example, the Nvidia Jetson sends a compressed string of instructions to the microcontroller for motor control; the way to interpret this string is to make an algorithm and use functions given to us by libraries to do so. We are doing a Python to C conversion as well, so data types will also be important to keep track of. While testing we had to keep in mind what data type is being sent. An example of this would be sending a string from python and having to use a conversion to make it a long int or unsigned float for interpretation. Pulse width modulation also needed to be calculated for the motor controls, so the math library was of great use for handling the work required. Without the use of this standard, much of the functionality for the embedded/hardware side would be mediocre in efficiency at best. As the saying goes, “don't reinvent the wheel”. Having a fast and efficient system is essential to operating in real time to improve the responsiveness of the robot. Coincidentally, next to assembly, C language is one of the fastest languages to choose from.

5.5 Robot Systems – Safety Requirements Standard

The scope for the standard *ANSI/RIA R15.06* states that “this safety standard applies to the manufacture, remanufacture, rebuild, installation, safeguarding, maintenance, testing and start-up, and training requirements for industrial robots and robot systems” [20]. A general overview of the sections provided by the standard: definitions, hazards to personnel, actuating controls, Installation of robot systems, safeguarding personnel, safeguarding devices, and maintenance. The purpose of this standard in our case was to avoid potential setbacks and damage in the development of Trash-E. It also provides several proposals for certain designs for power systems and other features. Some of these suggestions include designing reliable circuitry for controls, robot stopping circuits and emergency stop, grounding requirements, and much more.

To avoid any injury to users of the robot, we are following several of the pieces of advice given by *ANSI/RIA R15.06*. Circuitry that was implemented for the power systems needs to be safe to avoid electrical shocks and shorting of the system. Section 6.10 encourages the grounding of any electrical system within the robot; this also includes limiting access to the electronics during use. Since Trash-E will be interacting with its surrounding environment, it was important to include an emergency stop for unforeseen actions by the robot. Developing reliable circuitry with minimal interference from outside sources was also essential to the robot’s operation because if the motor controls and power supply DC to DC converter are not reliable unforeseen operations may happen.

In the case of our alternate design of having an articulate arm, rather than a stiff one-directional one, *ANSI/RIA R15.06* also provides safety standards for this. The standard suggests defining a maximum, restricted, and operating space for articulate robot arms. Shown in Figure 21 below is a graphical representation of this concept.

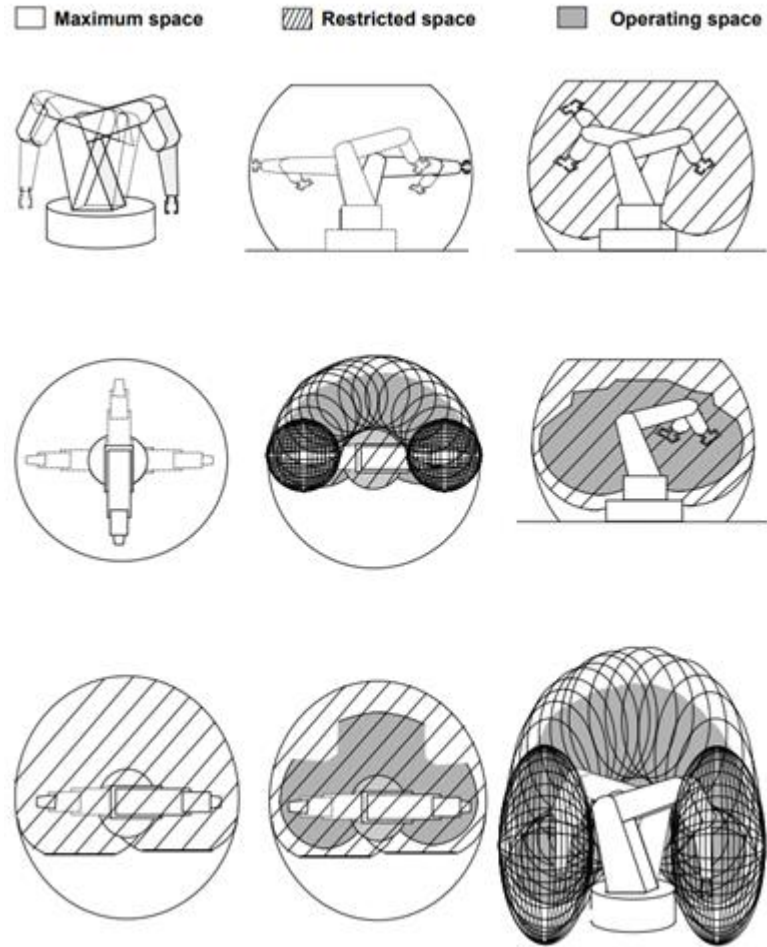


Figure 21: Use of Maximum, Restricted, and Operating Space

Maximum space, as the name suggests, the complete reach of the articulate arm to show the potential reach of it. This is useful in the case of estimating the safe range a user or bystander can be in during operation. Restricted space indicates the areas where the articulate arm should not move to. In the case of Trash-E, the arm has a 0-degree arc of restriction behind, because if it would rotate into that range, it would hit the electronics as well as the chassis. The operating space in front of it, a 150-degree arc, is its operating space to pick up trash. For throwing away the trash after it had been picked up, we used the case of dynamic restricted space for the arm to be able to bend into the restricted zone and drop the garbage into the garbage can. The definition for dynamic restricted space is as follows: “the safeguarding interlocking logic may be such that the restricted space is redefined as the robot performs its tasks” [20]. The stiff-arm design does not require as many restrictions due to its two-dimensional plane of movement.

6.0 System Design

6.1 Software Design

In this section we will be going over how the Trash-E software was designed. The following subsections will split the information for the software design on the three main software platforms. These will involve design explanations of the various features and algorithms that were used on the microcontroller, Jetson Nano, and computer vision.

6.1.1 Software Overview

Trash-E can maneuver and spot trash and pick it up on its own. Trash-E can roam on its own until it finds a piece of trash. Trash-E is constantly using computer vision to be able to recognize these trash items and maneuver its way towards each object. Once a trash item has been detected, Trash-E moves towards it until it gets a certain distance close to it. Trash-E has an arm with pincers that it will use to pick up the trash. It decides how it will approach the pickup of the item then it can grab the item and put it into its trash bin.

6.1.2 Computer Vision

6.1.2.1 Functionality

Trash-E has a camera connected to its main computer, the Jetson Nano. A camera is used to capture images of what is in front of Trash-E. The camera captures a live video feed, and our computer vision software analyzes each frame. In each frame, our algorithm performs object detection and looks for the trash items of interest. If there are multiple items in front of Trash-E that are of interest, the software will decide to follow the item that is closest to Trash-E. The Jetson Nano is powerful enough to process the images so that Trash-E can scan and identify trash objects in its view in real time. Once Trash-E has decided which trash object is detected it moves towards that object. The software sends data to the microcontroller through serial communication in order to determine a PWM signal that controls Trash-E's direction of movement. Once Trash-E is close enough to the object, the software stops accelerating Trash-E and brings it to a complete stop so that it can now pick up the trash. Once the item is in Trash-E's bin, the computer vision software initiates once and again and begins to look for an object. If there is no object in view, Trash-E utilizes ROS navigation until a new cup is spotted.

6.1.2.2 Development Environment and Platforms

6.1.2.2.1 Programming Language for Computer Vision

We are primarily using Python to develop the computer vision and machine learning software that is used on Trash-E. Although not as performant as languages like C++ due to the nature of it being compiled at runtime, Python is a very powerful language, has a lot of machine learning libraries, and the code syntax is very easy to read.

With Python we worry less about the syntax of the code and its semantics since it has no types, pointers, and everything is handled for us in easily readable code while still having all the powerful data structures and object-oriented functionalities that C++ provides. This left all the focus on the implementation of our machine learning and computer vision algorithms for our application. If we had gone with C++ as our primary language, there would have been a lot of nuances in the language that we would have to deal with aside from the machine learning and computer vision implementation. C++ has features like types and pointers that could possibly complicate our program that are handled behind the scenes in Python. However, C++ is a faster language and is very popular for robotics programming.

Python is already supported with many machine learning and computer vision libraries that we can use as well as many other frameworks, and extensions that provide a lot of functionality. Python is platform independent meaning that we could develop and implement our computer vision code on our desktop using the Windows operating system and pass on our code to a Linux operating system and it would still work as intended. This made the implementation of what Trash-E needs a lot simpler and more efficient.

6.1.2.2.2 Development Environment

For this project we used an IDE to develop our code for computer vision and machine learning such as PyCharm. Using an IDE like PyCharm gave us the best development environment to be able to write our code with efficient debuggers. PyCharm is also directed towards use for writing code in Python and offers a lot of support for doing so. We developed code on a Windows operating system since it is the current operating system that we own and the libraries and tools we needed to develop our machine learning computer vision platform are mostly cross-platform in terms of the operating system.

The computer vision software that we developed was transferred and run on the Nvidia Jetson Nano. The Jetson Nano runs on a Linux-based operating system and has a RAM capacity of 4GB. Our software is run on this system as an inference model meaning it does not train on the Jetson Nano. The training of the machine learning model was done on a desktop PC using a dedicated GPU to process the training. This saved lots of time for development and eases the stress on the

Jetson Nano RAM and processing power. The machine learning object detection model was also optimized by a proprietary Nvidia tool called TensorRT that optimized our model so that it can perform only necessary computations and save performance on the Jetson Nano while still ensuring the same results as if it were unoptimized.

6.1.2.2.3 Libraries

There are many ways to go about creating computer vision applications using the extensive number of libraries that are available to use. Using these libraries allowed us to use state of the art algorithms and architectures already created by AI researchers. This cuts down the development time and overall gives us a more efficient product. Python is supported by many machine learning libraries as well as some that are tailored toward computer vision.

One route that we could have chosen for designing our machine learning software is the more recent PyTorch library. This is Meta's open source machine learning framework that was released in 2016 and is relatively new to the industry and is primarily used by researchers and people who want a more pythonic framework. In comparison to TensorFlow, PyTorch is a bit simpler to read and is more intuitive since it is more like python making it easier to learn than TensorFlow. This allows people to make quicker prototypes and get projects going much quicker too. Debugging with PyTorch is also simpler to do with common tools compared to TensorFlow which requires you to use another tool for debugging. Using PyTorch would involve similar processes as if we were using TensorFlow. We could use a pretrained model such as Faster R-CNN that is offered by PyTorch and attempt to leverage it for transfer learning to be able to train it to our custom dataset a lot quicker and potentially get better accuracy than training from scratch. If we ultimately decided to write our own CNN architecture instead of using a researched model during prototyping, PyTorch would make building our neural network a lot simpler than on TensorFlow. This is because PyTorch uses dynamic computational graphs which allows us to change behavior of our model at runtime which makes optimizing models much easier. Since PyTorch is newer than TensorFlow, that means it is not as extensive. However, for our use case in designing this computer vision application that is not an issue.

On the other hand, the other route we could have chosen to develop our computer vision software is TensorFlow. This is the most popular machine learning framework in the industry for production. The reason we would want to use TensorFlow is because it will allow us to use a bigger selection of famous state-of-the-art pretrained models and give us the option to train our own custom object detection models using our custom objects just like we would be able to with PyTorch. TensorFlow is backed by Google and has frequent updates and support. It is also open source, so we don't have to spend any money in order to use it, just like PyTorch. TensorFlow is packed with many built-in functions for machine learning that would make training and testing our models a lot quicker and testing more efficient than from scratch. TensorFlow also includes TensorBoard

integration which could help us in fine tuning our model by showing us our testing evaluation metrics. One downside for TensorFlow is that in order to train our model on a graphics processing unit, or GPU, it must be manufactured by Nvidia and is only supported with Python. However, this will not be a problem for us since we will be using an Nvidia GPU that we already own to train our model. TensorFlow also has a much steeper learning curve than PyTorch.

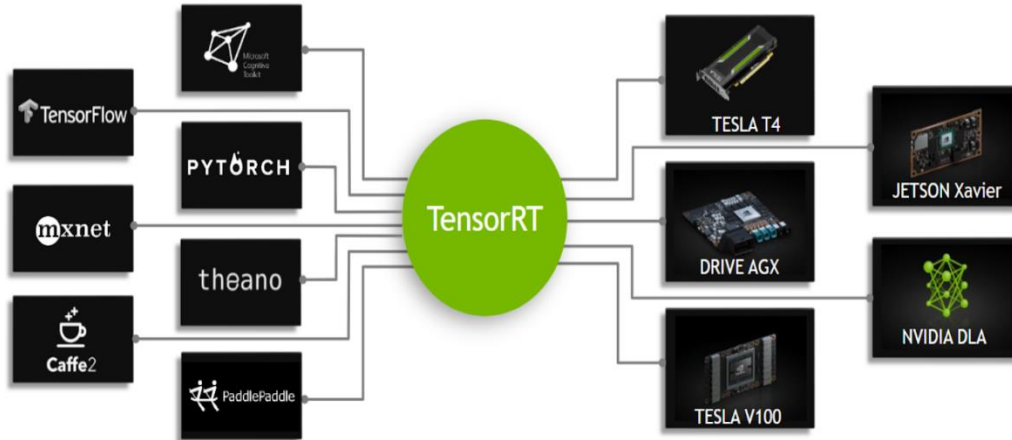


Figure 22: Nvidia GPU Optimized Models (Courtesy of Nvidia)

Both PyTorch and TensorFlow are supported by Nvidia TensorRT. This is an SDK by Nvidia that is written in C++ that takes a trained model from libraries like TensorFlow and PyTorch and converts to a more optimized model for Nvidia GPUs as shown in Figure 22 while maintaining the same results which includes the Jetson Nanos GPU. This drastically increased performance on the Jetson Nano while using our inference model on it.

There are a couple of options of how we could run the model on our Jetson Nano. With PyTorch we could convert the model to ONNX format and convert it straight to TensorRT to run on the Jetson Nano. With TensorFlow, we could either use the Tensorflow-TensorRT integration on TensorFlow and be able to still use unsupported parts of the model on TensorFlow during runtime while everything in the model that is compatible will be using TensorRT converted computations. The last option would offer the best performance for the Jetson Nano which is to convert the TensorFlow model to UFF format and convert it straight to a TensorRT model.

Our initial decision on which library to use was TensorFlow. Later on during prototyping we ran into issues using TensorFlow on the Jetson Nano, so we decided to switch to PyTorch. These issues included not being able to run the model well and it would not convert properly to TensorFlow-TensorRT or TensorRT.

Other than the machine learning library, the most important library that was used for the object detection software was Nvidia's jetson inference library. This library is made specifically for the Nvidia Jetson platform and runs on the Jetson boards.

It provides all the necessary dependencies and packages needed to get started doing object detection and image detection. It is an inference and real time deep neural network vision library that offers many APIs to train a model using PyTorch, converting our model to ONNX and TensorRT, and run inference of our model. It also provides the data that we need to analyze our detection boxes. The jetson inference library also offers its own labeling software that made capturing images and labeling our dataset simple.

Other libraries that were used to assist the main computer vision implementation are PySerial. PySerial is a python library that encapsulates the access for the serial port. It provides backends to python that will automatically be configured for the platform we will make a serial connection to. The serial connection we make uses UART between the microcontroller and Jetson Nano. With this library we initialize that connection so that the computer vision software can guide Trash-E to the trash items. Computer vision models can give results in bounding boxes. The coordinates from these boxes are sent to the microcontroller via serial connection on PySerial so that it can guide Trash-E's movements with a PWM signal.

6.1.2.3 Object Detection Model Architecture

The algorithm that we needed to achieve good object detection is a convolutional neural network. As mentioned before, both Pytorch and TensorFlow offer many state-of-the-art pretrained models that are based on convolutional neural networks and have very good results in object detection. These models are already trained with very good weights which allowed us to leverage transfer learning for quicker and efficient training on new objects.

We originally used a pretrained model from the TensorFlow Model Zoo located on their GitHub [21]. These models are high quality researched architectures or models that are provided by Google's TensorFlow GitHub. However, since we had to make a switch to using PyTorch instead, we used the pretrained models that are offered by PyTorch. Using a pretrained model gives us the advantage of utilizing transfer learning and could ultimately make our training more efficient and more accurate in the end. Leveraging the technique of transfer learning will enable us to train the final layers with our new images and classes in much less time since the weights from previous training are already very good and it will be accurate at detecting our new classes of objects as well. These models also offer the option to be trained from scratch. The TensorFlow model zoo offered a lot of options for pretrained models. Some of the object detection choices we have include SSD MobileNet v2, SSD ResNet50 v1, Faster R-CNN ResNet, and Mask R-CNN. The PyTorch model selection had similar choices.

Mask R-CNN offers normal object detection such that it uses detection boxes. However, a main key feature that differentiates this model from the others is that it offers masks as well. These masks essentially wrap the object instead of just placing a box around the detected object. This feature, however, comes with a

heavy cost. The performance runtime of this model can take as long as 301 milliseconds (ms). This number could likely be worse if we were to run it on the Jetson Nano. This model also has a high mAP accuracy but comes with a great performance cost. For the purposes of our project this model would be overkill, and the masks are an unnecessary feature and would be extremely inefficient for Trash-E.

The Faster R-CNN is another high accuracy model but has better performance than the mask R-CNN. This model only uses detection boxes for identifying objects and has a much better average runtime of the range 53 - 236 ms depending on the choice of resolution and the ResNet architecture.

SSD ResNet50 v1 performs very similarly to the Faster R-CNN models. At the lowest resolution it offers a faster runtime and also gives a slightly higher average accuracy for the model. However, when we compare this model with the SSD MobileNet v2, it is surpassed in speed.

The SSD MobileNet v2 model option offers us the best performing runtime for object detection. At the lowest resolution it offers object detection in 19 ms, surpassing all the previously mentioned models. However, this increase in speed does come with a tradeoff of accuracy. The accuracy for the fastest version of MobileNet is averaged at 20.2 mAP. Compared to all the previous models this is the lowest average accuracy.

After considering all our options for the object detection models available in the TensorFlow Model Zoo and the PyTorch models offered on the jetson inference library, the model we used is the SSD MobileNet v1 model. This is a single stage object detection model. We trained the model to be able to detect our own custom objects only, which in our case are cups. However, this custom dataset can be easily expanded upon in the future.

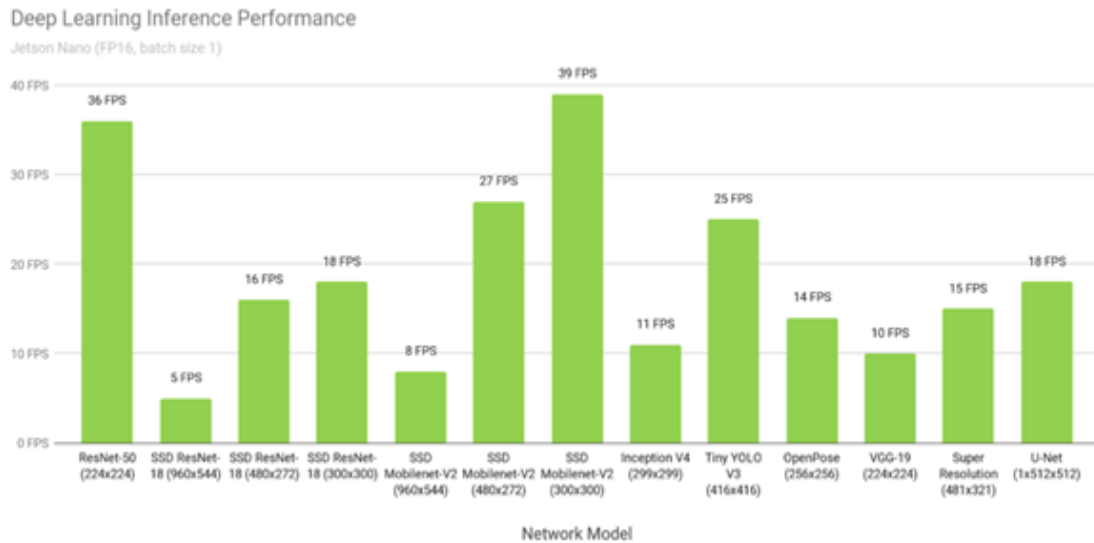


Figure 23: Object Detection Model Performance on Jetson Nano (Courtesy of Nvidia)

The SSD MobileNet v1 is similar to its MobileNet v2 successor. There are a couple of reasons for choosing this specific model, one being that it had very high performance compared to the other models as shown in Figure 23, which is very important for our application since we are running the inference model on a Jetson Nano in real time.

Since our model will be running on the Jetson Nano, we should optimize for performance and efficiency for the sake of hardware capabilities so therefore we are going with a lower resolution model. The RAM that is available on our Nvidia Jetson Nano is 4GB, the main operating system on the Jetson Nano already consumes a decent portion of the RAM so it would be critical that our design uses a performant algorithm with lower computational cost so that we don't run into issues of crashing or overheating our Jetson Nano which would render our product useless. Our dataset and objective do not require the most precise and most accurate model to detect trash objects. Therefore, it is best to go with performance over accuracy for our convolutional neural network architecture or object detection model. The SSD MobileNet V1 model runs very well on the Jetson Nano.

6.1.2.4 Creating The Dataset

The first step in every computer vision task is to gather the data that we will use to train our model for computer vision. For our project, we needed to gather photos of all the objects that we want Trash-E to detect. We took pictures using the camera that we used on Trash-E. It mattered to use the specific camera since it has a 160 degree FOV which altered the shapes of the cups when compared to images taken with a normal FOV.

There are techniques involving scripts that we could have written to automatically take pictures from a webcam attached to our desktop computer. This script can automatically take pictures for the appropriate classes. The script would loop through each class that we want to detect and take a certain number of pictures accordingly until we've completed taking pictures of all classes. This method would be great if we were only creating computer vision software that would only run from a webcam on top of a desktop inside a normal room, such as signing into your account by face detection or placing filters on peoples' online chat backgrounds. However, for our project, this was not an ideal approach since Trash-E operates in many different conditions and not just one room in place.

The approach we went with was to manually take these photos for our dataset. These photos needed to show the items in many orientations, lighting conditions, colors, and distances. It is important that the pictures are done in all these ways so that our algorithm can recognize them from any position in any condition that Trash-E can operate in. If we did not take these steps to consider the different conditions in our images, then we would get a very inaccurate model with many false positives as well as false negatives.

We needed at least 100 photos per class to have a decent training on it. Luckily with transfer learning, the accuracy will be good even with the new small dataset since the pretrained models' weights were already very good. We ended up gathering approximately 500 images of the cup class.

Once we gathered these images, we labeled the data based on the object it is and what we want our network to classify it as. We used a labeling software from the jetson inference library that allowed us to easily select the objects in our images that we want to label and properly annotate them and organize them into their appropriate files and formats.

We also needed to create a label map for our dataset. This map data structure holds all the labels for our dataset including the label name as well as a unique identifier for that label in the map.

6.1.2.5 Training The Object Detection Model

We trained our model using the PyTorch library for Python. The training was done on the Jetson Nano using the GPU. Training with a dedicated GPU on a desktop made the training process complete quicker than it would on the Jetson Nano but due to using the jetson inference library, it limited us to using the Jetpack software that was loaded onto the Jetson Nano which isn't available on desktop.

The jetson inference library offered APIs for PyTorch, which we utilized to easily perform functions on our model. The jetson inference library includes a training script designed for the specific model that we chose. This script is modifiable, and we can change some parameters. One of the parameters we changed a bit is the number of epochs the model trained for, or the amount of training iterations it

makes. If we felt that the accuracy was not good, we would train it for longer or increase the number of images in our dataset for training.

Once our model was trained, we used the jetson inference library to access a camera connected to our Jetson Nano and feed live video to our object detection model. Then our object detection model examined each frame and detected our custom objects that are in them. Using the jetson inference library, we were able to draw boxes around the cups in each frame to show the location of the cup on the image that the model predicted as well as return the values of the coordinates in the detection boxes that will come as a list of bounding boxes.

6.1.2.6 Running our Trained Model on Jetson Nano

Once we were satisfied with the accuracy of our model, we converted the PyTorch model into ONNX format using an API from the jetson inference library. We needed the ONNX format because that is the universal model format that Nvidia TensorRT uses to create a TensorRT engine. Nvidia offers the TensorRT software that allowed our model to run efficiently on the Jetson Nano. After converting the model into a ONNX format, we called an API for TensorRT object detection that built our optimized TensorRT graph which creates a TensorRT execution engine. This allowed for better performance on the inference graph than just using the converted graph. With this TensorRT optimized model we ran our trained object detection model on our Jetson Nano more efficiently than just the native PyTorch or ONNX model. Using TensorRT we were able to achieve frame rates of around 30 frames per second.

6.1.2.7 Serial Communication with Microcontroller

One of the most important aspects of our software design is the connection between the computer vision software and the microcontrollers software. The microcontroller oversees maneuvering Trash-E as well as controlling the arm in which it will use to pick up trash. However, there is no way that the microcontroller would know where a trash object is on its own. Therefore, the computer vision software must gather necessary data that it can send to the microcontroller via serial communication so that it can determine what to do. The main three problems we needed to solve for Trash-E's vision and maneuverability decision making are:

1. How will Trash-E know which trash object to approach in a scene?
2. How will Trash-E know how to maneuver itself to that particular trash object?
3. How will Trash-E send that information between the Jetson Nano computer vision software and the microcontrollers software which controls all the motors?

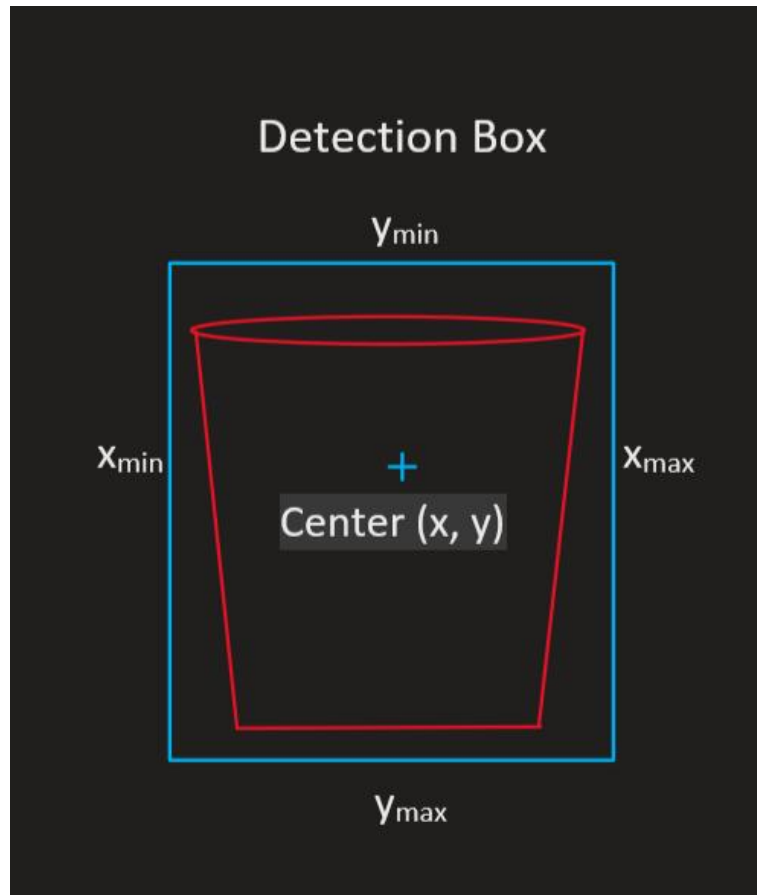


Figure 24: Object Detection Bounding Box Coordinates

The jetson inference library already offers bounding box coordinates that surround all the objects of interest in the algorithm's view. These coordinates in each bounding box include the center x , and y coordinates as shown in Figure 24. This essentially gives us the coordinates in the image that this object was found in and creates a tight box around it. With these coordinates we could determine many interesting bits of information that we can use to guide Trash-E.

The jetson inference library also gives us the area of each detection box. Our software sorts the areas of each object's bounding box that it has detected in decreasing order of area. In other words, the biggest area detection bounding box will be in the front of the sorted list of bounding boxes. Why is this necessary for Trash-E? With these bounding box areas sorted, we now have a way that Trash-E can decide which object in its view it should approach. A bigger detection box area means that the object is closer, and Trash-E will approach the specific object or biggest bounding box. For testing purposes, we output the direction Trash-E needs to move to pick up the closest cup in the terminal.

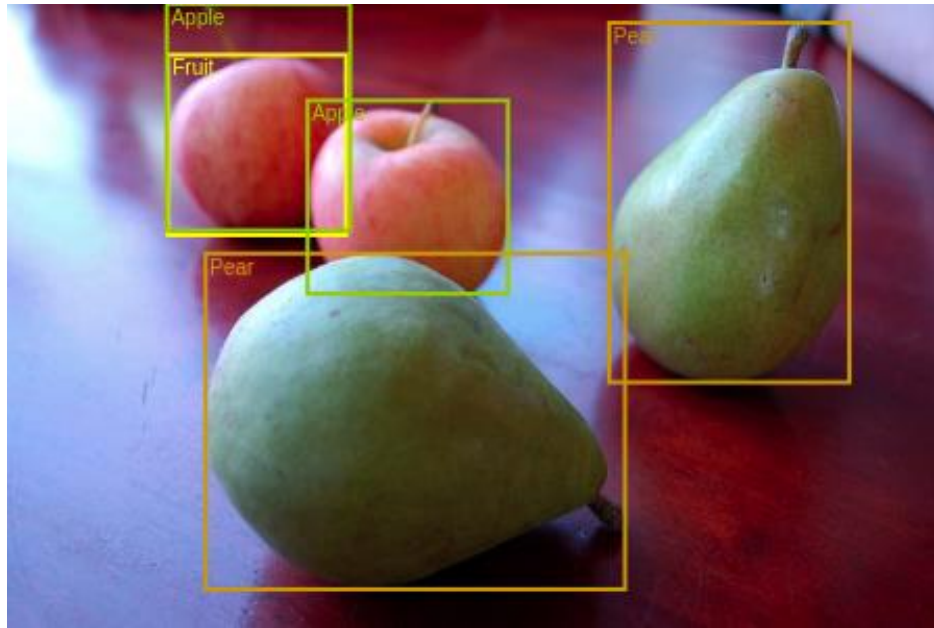


Figure 25: Object Detection Bounding Boxes (Courtesy of Algorithmia)

For example, let us say our object detection model was tracking fruits. In Figure 25 we see that the closest pear has the largest bounding box area. In this case Trash-E would approach that pear since it has the largest area and will continue to be the largest since as we get closer the area can only increase. In our real case of trash objects, which are red solo cups, they will always be of the same or very similar size. Therefore, the edge case of a huge object in the background being mistaken as the closest object compared to a smaller object that is closer would be very rare.

Trash-E must eventually stop the approach towards the object when it gets close enough for pickup. A method that we have designed to solve this problem is to have a sensor towards the front of Trash-E that detects the presence of an object in front of it. The computer vision software keeps signaling Trash-E's microcontroller to move forward until the sensor detects the object is close enough to grab with its arm for trash removal. Once at that point, the computer vision software is overridden, and the microcontroller takes control of moving the robot's arm to place the item in its trash bin and ignores incoming data from the Jetson Nano's software.

This would work well if Trash-E was always in a straight path to the object. However, that will rarely be the case, so another aspect of Trash-E's maneuvering needed to be solved. An object that is detected and is closest can be anywhere in Trash-E's camera view on the x-plane. To turn Trash-E so that it turns and moves straight towards the object, we used the coordinates to determine that maneuver.

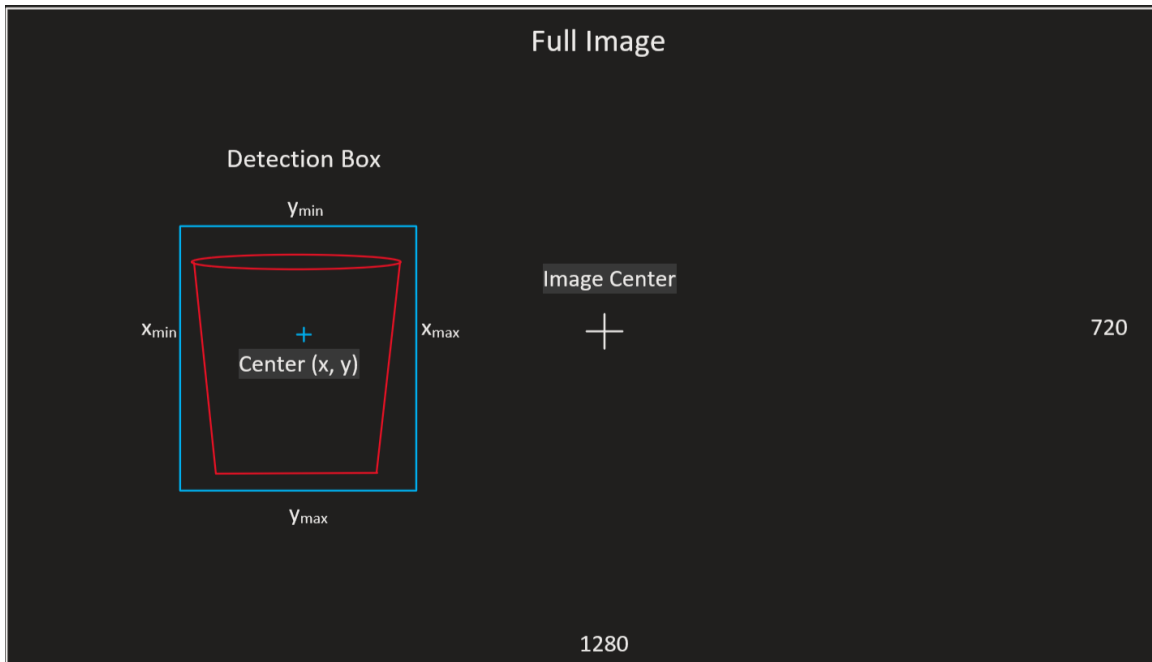


Figure 26: Detection Bounding Box Displacement from Image Center

With the coordinates from the closest bounding box, we determine the center of that bounding box. We calculate the center of the bounding box by taking the center x value. We compare that center x axis coordinate value with the center coordinate values that we determined for our entire full image. Which we decided is between 750px and 800px for our 720p image. With both the coordinates of the center of the bounding box and the center of our image we determined the displacement between these two points as shown in Figure 26 as well as the direction it is from the center origin of the image.

Since Trash-E only needs to turn right, left, or go straight, only the x-plane is of significance to us. Therefore, the displacement check between our bounding box's center and our image center only needs the x-coordinates. If the cup center x value is less than 750px, that means that the object is to the left of our origin and Trash-E should turn left. Otherwise, if the center x value is greater than 800px, then Trash-E should turn right. Trash-E will continue to turn until the image center and the bounding box's center are within the same threshold which was between 750px and 800px. The reason we don't want the Trash-E to turn until the values are exactly zero is because that would lead to extremely fidgety movement. Due to real world limitations and scenarios, Trash-E would constantly try to correct itself because it will overshoot that equilibrium. Having the small buffer threshold eliminates this mediation issue.

With these methods implemented, Trash-E can now determine which way to turn, and which object it should move towards based on the data from the object detection model. The remaining step is to design a way that we can pass this information to our microcontroller that ultimately controls these movements based on the information it receives from the Jetson Nano. Within our detection script we

have functions that send that data in a UART method to the microcontroller. The Jetson Nano never needs information from the microcontroller, so the communication will be one-directional. Through this script we initialized a connection over UART with the microcontroller. We used a library called PySerial which makes this process extremely simple. This library has support for serial support access on multiple platforms.

Once our object detection software has determined which direction Trash-E needs to turn after the previous mentioned steps in our design, we send a value over UART that the microcontroller interprets to perform the corresponding maneuvers. We only have three decisions to make on which turn to make, which are left, right, or continue forward. Therefore, we hardcoded three values to send over UART that the microcontroller can easily translate. Each value sent over UART will be a 32-bit integer. If the displacement checks return a 0 value, then the string "0" encoded in 'UTF-8' format is sent over UART which corresponds to going straight. If it returns a negative number, "2" is sent over UART which corresponds to turning left. If it returns a positive number, "1" is sent over UART which corresponds to turning right. If in this case our object detection model does not detect any objects that we are looking for, our calculations return a null value, and we will send a value of "3" over UART.

6.1.2.8 ROS Navigation with SLAM

SLAM was implemented for Trash-E to autonomously move around the environment. This was done using the ROS Navigation stack. The libraries that were used are hector_slam for generating a map of an unknown area, map_server to save and crop the generated map, amcl for determining the robot's estimated position, and move_base which creates a path to a goal and sends messages to direct the robot's movement. We chose to use ROS because it runs well on Linux distributions, can be run seamlessly with our other Python script, and is simple to set up. There is also a lot of documentation and online tutorials for help setting things up for robot navigation.

6.1.3 Microcontroller Software Design

The movement of Trash-E is controlled by the microcontroller using the Cortex-M4 processor. The main tasks of the microcontroller include: move the robot using servos on the wheels, determine when the trash is close enough to be picked up, detect obstacles utilizing buttons on the corner of the chassis, lower and raise the arm of the robot with a stepper motor to put the trash into the bucket on the chassis, and to grip the trash with the gripper using a servo. The software was written using Texas Instruments' Code Composer Studio IDE since the microcontroller is also made by TI and we are familiar with using the IDE.

6.1.3.1 Flowchart

The overall structure of the software is implemented using a finite state machine (FSM). Since the overall processes that the robot will do is linear and needs to be completed in order, the FSM is perfect for this application. The overall logic that the microcontroller software follows is shown in Figure 27 below on page 68.

6.1.3.2 Wheel movement

To move Trash-E, there are two servo motors attached to two of the three wheels. These motors are attached to the front two wheels. Having the motors attached in this way will allow us to spin the robot without moving too much forward or backward. The microcontroller will generate a PWM signal on the GPIO pin that is connected to each servo. For the wheels we chose pins 13 and 14 to generate the PWM signals. To move the robot straight, equal duty cycles are generated to keep both sides of the robot moving at the same speed. To turn left, the right servo is given a higher duty cycle. The opposite is true to turn right where the left side has a higher duty cycle.

6.1.3.3 Baud rate

The decision on whether to move straight or to turn is decided by the Jetson Nano and sent to the microcontroller using UART on pins 15 and 16. A baud rate of 9600 is chosen to send the information. A high number of bits being sent is not needed as too much information could result in the robot over-correcting or potentially under-correcting with the PWM signal generation being overwritten too quickly. To achieve this 9600 baud rate, we need to alter the registers shown in Table 10.

Table 10: UART Baud Rate Register Configuration

Name	Description	Value
UARTIBRD	UART Integer Baud-Rate Divisor	Determined by formula below
UARTFBRD	UART Fractional Baud-Rate Divisor	Determined by formula below
UARTLCRH	UART Line Control	0x00
UARTLCTL bit 5	HSE of UART Control	1'b0

The values for the integer and fraction portions of the BRD can be found using this formula:

$$BRD = \frac{UARTSysClk}{(ClkDiv * Baud Rate)}$$

The PIOSC is used for the UARTSysClk and the ClkDiv is determined by a bit in UARTCTL. This bit is set to 0 to achieve the divide by 16 we need for our regular speed operation.

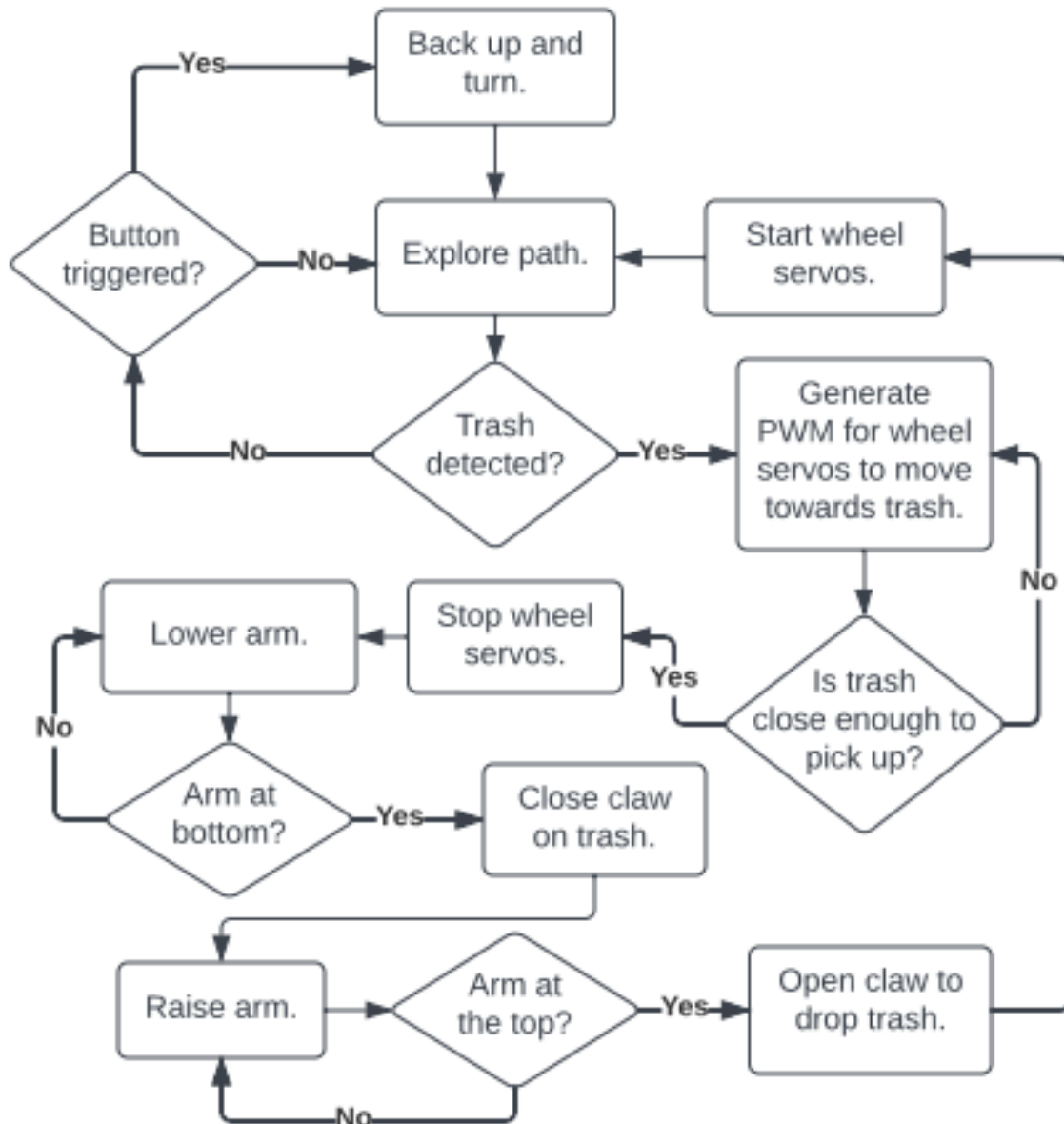


Figure 27: MCU Software Flowchart

6.1.3.4 Command handling

While the robot is in this movement state, it is continuously listening to the Jetson Nano for instructions. There are three different commands that can be sent to the microcontroller: steer left, steer right, or continue straight. With the small number

of commands we implemented, the number of bits that need to be sent from the Jetson Nano to the microcontroller can also be kept low. We utilized the lowest amount of data bits that the UART interface allows us to send, five data bits, to send a single character. If a 32-bit integer “0” is parsed from the Jetson Nano, the PWM of the two servos will not be altered and the robot will continue to move forward. If “2” is sent it means we need to turn left. If “1” is sent it means we need to turn right.

6.1.3.5 PWM Signal Generation

The PWM signals for the motors need to be generated using the timers in the microcontroller. For each servo a separate PWM needed to be used. We chose pins 13, 14, 43, 44, 61, 62, 63, and 64 for the PWM signals since they are all 32/64-bit counters which gave us the most flexibility in timing. To set the timers to PWM mode, the registers in Table 11 needed to be altered in each GPTM block for the corresponding timer. The timer period is set in GPTMTnILR and the match value is set in GPTMTnMATCHR. The signal stays high while the counter is increasing, then goes low after the value is the same as the match register. This allows us to set any duty cycle that we want and can be unique and individually controlled for each motor.

Table 11: PWM Register Configuration

Name	Description	Value
GPTMCFG	GPTM Configuration	0x4
GPTMTnMR	GPTM Timer A/B Mode	0x006
GPTMCTL bit 14	GPTM Control	1'b0
GPTMTnILR	GPTM Timer A/B Interval Load	0xFFFFFFFF
GPTMTnMATCHR	GPTM Timer A/B Match	Determined by what duty cycle is required

6.1.3.6 Wheel Stopping

As the robot gets closer to the trash that it is going to pick up, the ultrasonic sensor starts sending response signals to the microcontroller indicating how far away the object is. The sensor needs a PWM signal as input to trigger the detection, and the response is sent back to the microcontroller on a GPIO pin. Pin 43 is used to send the PWM to the sensor. The microcontroller then samples how long the response is high which indicates how far away the object is. The response is sampled on pin

44. Pin 43 is set according to Table 10. Interrupts were used to sample the signal from the ultrasonic sensor. The interrupt triggers whenever the value is high, and a flag is set to indicate the start of a reading. If the flag is set and another interrupt is enabled, the counter will be incremented. A separate interrupt occurs when the flag is set, and the incoming signal is set to low. This calculates the distance based on the counter then sets both the flag and counter to zero.

Since the Jetson Nano will be communicating with the microcontroller to direct it towards the trash, we need a way to have the robot stop when it's close enough to pick it up. The ultrasonic sensor is attached to the base of the chassis on the front, just below the arm, where it has a good view of the area in front of the robot and won't be obstructed by other pieces of equipment. Once the sensor indicates that the object is about six inches away, an interrupt occurs to set the duty cycle of the wheel servos to 0% and move the state to pick up the trash.

6.1.3.7 Grabbing Trash

A smaller servo is attached to the end of the arm to handle opening and closing the robot arm's claw mechanism. By utilizing the servo, we will always know what position the claw is in and can adjust it accordingly. This servo will be connected to pin 61 set to PWM mode. With the motor starting out at 90°, it is rotated to 0° by altering the value in the corresponding GPTMTnMATCHR register to clamp onto the trash. The process is reversed to let go of the trash at the top of the arm movement over the bucket.

6.1.3.8 Raising/Lowering the Arm

Once the trash has been picked up, the robot must move it to the bucket on its back. The microcontroller interfaces with an A4988 motor controller to drive the stepper motor. The input on the A4988 will be hooked up to pin 64 which is set up in PWM mode. The microcontroller continues to step the motor downwards until 270 steps with one-quarter micro-stepping has completed. We use a single value to move the motor up and down since the arm always has to travel a set distance up and down.

6.1.3.9 Function Descriptions

This section will describe the overall steps that happen in each state the robot can be in. There will be three states total. After the third state, the microcontroller will return to the first state which acts as an idle state.

6.1.3.9.1 Discover Trash Function

The goal of this function is to find trash that is on the ground.

- START
- Receive input from Jetson Nano

- Update PWM signals accordingly to parsed input.
- END

6.1.3.9.2 Acquire Trash Function

The goal of this function is to move Trash-E towards the trash based on the Jetson Nano's commands.

- START
- Keep moving toward trash until the ultrasonic sensor reports the trash is six inches away from the claw.
 - Read input from Jetson Nano. Change PWMs accordingly for 32-bit integer inputs.
- Set duty cycle to 0% for wheel motors.
- END

6.1.3.9.3 Move Trash to Bucket Function

The goal of this function is to move the grabbed trash to the bucket on the back of Trash-E.

- START
- Set the DIR pin connected to the motor controller low.
- Generate 25% duty cycle PWM and increment counter on each pulse to lower the arm.
- Once the counter is at the specified value, generate 0% duty PWM for the arm.
- Generate 50% duty cycle for the servo at the end of the arm until it has reached 90°.
- Set the DIR pin connected to the motor controller high.
- Generate 25% duty cycle PWM and decrement counter on each pulse until it's zero to raise the arm.
- END

6.1.4 3D Modeling Software

There are many options when it comes to choosing a piece of 3D modeling software. Software that was under consideration: Tinkercad, Solidworks, AutoCAD, FreeCAD, Fusion 360, and OpenSCAD. Each has varying features and learning curves, and some may be more appropriate for the project than others.

TinkerCAD offers a simple user interface along with fewer features than other modeling software. The number one thing about TinkerCAD is its learning curve. Testing the software itself, it walks the user through several tutorials to help them get familiar with the interface as well as how to use all its tools. Due to its simple design STL file, 3D printing can be done relatively fast. If one plans

on adding any detail to the model, it would make it more difficult. After working with this software for an hour, we created a simple model based off the initial design for the robot shown in Figure 28 below. This software is completely free.

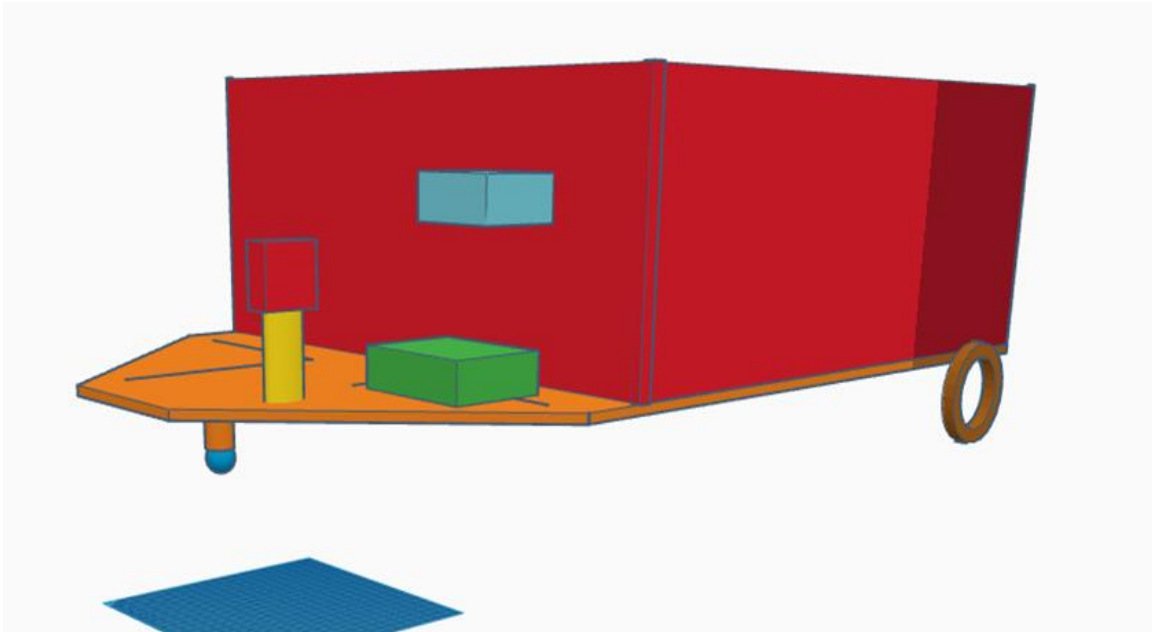


Figure 28: Three Wheel Design Created in TinkerCAD

SOLIDWORKS offers a 3D CAD software free for students while having a code given by my.cecs.ucf.edu. This truly is a professional piece of software that takes many hours of training/practice to get the hang of. It offers many tools to complete any task, and it lets the user simulate the movements of models. Despite that, the first time AUTODESK is opened, it recommends reading through hundreds of pages of documentation on how each tool works and so forth. Our group consists only of computer and electrical engineers, and no one has worked with a software on this kind of level before, so the documentation was very overwhelming. Instead, opting for a YouTube tutorial was able to give a good introduction to the software and how to make basic shapes and such, but it wasn't enough to make a robot chassis. If we had more time to train with SOLIDWORKS, we would have liked to utilize this software to design the chassis.

Being one of the most used, AutoCAD is undoubtedly one of the most powerful 3D CAD software packages on the market. Reviewing its features, we can create detailed designs while being able to collaborate with other users on the same project. It provides many visualization tools to aid in the production of 3D models. The large number of tools and features is overwhelming to a beginner with limited time and without direct guidance. This suffers from the same issue SOLIDWORKS does. In addition, a 30-day free trial is offered before a subscription is needed.

FreeCAD is a step up from TinkerCAD since it's capable of much more such as configuring settings, it has more than quadruple the number of tools, clear and descriptive UI, and it's able to turn a two-dimensional schematic into a 3D model.

In terms of complexity, it is a decent amount above TinkerCAD and a bit below both AutoCAD and SOLIDWORKS. FreeCAD is at a decent middle ground. There are plenty of videos that showcase its tools and features in a manner that covers all bases and is easy to follow. Overall, its open-source platform allows it to compete with larger paid softwares like the two previously mentioned while still being free.

Fusion 360 is another solid choice and slightly better than FreeCAD in terms of an easily navigable user interface while still maintaining the same number of features. 2D design schematics can be made and transformed into 3D models. Eagle can also be used in tandem with Fusion 360 to seamlessly integrate the CAD and PCB software to give a more accurate representation of the orientation of components after printing the Chassis. It also has great compatibility with several popular slicing products making 3D printing easier by reducing errors in the STL file. Compared to the top end software like AUTODESK and AutoCAD it has nearly the same number of features, but with a more user-friendly UI along with a package that lets you port over your PCB designs.

OpenSCAD is a nearly limitless 3D modeling software that relies mainly on a scripting language to generate models. Out of all the choices, OpenSCAD has the largest learning curve, but has much higher reward in terms of modeling. It has little to no interactive UI making it very unfriendly to new users. Much time and effort is needed to become efficient because a new language must be learned. To make matters worse, taking the time to code together models is a slow and tedious process compared to predesigned tools in other user interfaces. Although it is powerful, OpenSCAD was eliminated due to its complexity.

TinkerCAD would be the safest option as the interface and tools are minimal and easy to understand. As mentioned before, experimenting with it for the first time, we were quickly able to make a rough model of the chassis. Building the complete chassis in TinkerCAD then disassembling it into individual pieces should allow for easy printability. The build plate volume of the Prusa i3 MK3S+ is 9.84 x 8.3 x 8.3 inches (250 x 210 x 210 mm). Any pieces that were larger than this size, like the support beams on the bottom of the acrylic sheet, were split into two separate pieces and glued together.

6.1.5 3D Printing Software

Typically, 3D printing software, or slicers, can take several different kinds of 3D modeling file types such as 3MF, STL, OBJ, AMF, and sometimes unique file types that contain special information. Depending on the brand or version of a 3D printer will determine the possible software that is compatible with it. Some slicers are more capable and provide more features than others. Printer brand and the way each machine generates G-code (instructions for 3D printer) can also contribute to the quality of a print. Several 3D printers available on campus are open for use for a fee. To save money and have more convenience, we are using a team owned

Prusa i3 MK3S+ for printing parts for Trash-E. For the software our options are Cura, Simplify3D, Slic3r, KISSlicer, Tinkerine Suite, Prusa Slicer, Repetier, OctoPrint, an SelfCAD.

Cura is the most widely used 3D slicing software out on the internet. Part of its success is from its free price tag along with being open source. Of course, Cura is specifically designed for Ultimaker 3D printer users, but it offers compatibility with several others. Unlike most printer software, this software has three different stages of processing during the process. In the first phase the user can configure printing settings and decide how they're going to slice the model. The second scans the model after generating the G-code, finding any areas in need of structural support or that could potentially fail. The last stage, it's possible to watch prints progress live or remotely.

Simplify3D is one of the most powerful and far-reaching slicers. Its greatest strength would be its ability to easily implement detachable supports. This is a category most slicers struggle in. It allows for configuration of the material and support thickness which no other slicer provides. By having effective supports it ends up leaving a better surface finish due to its supports being able to cleanly detach. It ends up leaving less of a cleanup job for post processing resulting in better looking prints. Its pre-print simulations are better than Cura's because of how accurate its error detection system is. Out of all slicers, Simplify3D has the most accurate and precise calibration system for tuning retraction, infill settings, cooling fans, and brims. There is no comparison in terms of quality in any other slicer software. Despite this, there is a price point of 150 dollars making it too expensive for the scope of this project.

Slic3r, while being relatively old, created in 2011, is another open-source 3D slicer. Initially, it was released as a non-profit project, and it ended up being one of the greatest 3D model slicers on the internet due to its massive community on GitHub with more than a 1000 people contributing to the project. It's a great all-rounder having the ability to display a preview of a print, and it can even do so with multi extruder 3D printers that allow different support materials to be used to make removing supports easy. It's also a lightweight piece of software requiring no dependencies. Other features include brim, micro-layering, bridge detection, command line slicing, variable layer heights, sequential printing, honeycomb infill, mesh cutting, object splitting into parts, AMF support, perimeter avoidance, and different extrusion lengths. Slic3r may beat Cura in terms of its functionality, but it doesn't have a user interface.

KISSlicer has a more detail-oriented approach than the other software packages. Some of the options in KISSlicer are settings no casual 3D printer would alter. This software is tailored to more experienced individuals. KISSlicer is only able to work with single extruder 3D printing, while the dual extruder version will cost an extra 42 dollars. There is a premium option that costs 82 dollars that allows one to combine multiple STL files and print them all at once. Compared to other slicers

KISSlicer seems a bit lackluster for its cost. If the free version was used, it still has access to every content update that adds new mesh topologies, filaments, types of 3D printer, and print styles. Finally, the user interface is not that great compared to other slicers that are free. Based on this information, we didn't choose KISSlicer as our slicer software.

Tinkerine Suite is very similar to TinkerCAD. It's a great introductory piece of slicer software for people new to 3D printing. Originally it was designed for teaching children about 3D printing, but it has enough features like descriptions on how each setting change will affect the print, or how long a print will take to finish, as well as indicators for unsupported bridges and weak points in the print. Although it's a great learning tool, most slicing technology isn't that hard to learn sufficiently. As a result, it's a bit too simplistic for what we need compared to the others to ensure the design of our parts are optimal.

Prusa Slicer takes advantage of Slic3r's open-source software and reimagines it into something more specialized. It's main selling point is the attention to detail and the ability to tune almost every aspect of a print. Part of the reason why this slicer is so popular is due to its great software features that improve upon every aspect of Slic3r, as well as the quality of the Prusa printers themselves. Furthermore, it's specifically tuned to enhance the printing quality of Prusa printers, making their printers even more appealing. On the other hand, Prusa slicer can work with several other non-Prusa printers and over 50 different 3D printer filaments, it even works with resin printers. Just like Slic3r, Prusa Slicer is open source, so it's easily modifiable for anything a user might need to accomplish.

The next two slicers fit within the same category since they both allow for remote printing, but with their own pros and cons. OctoPrint and Repetier Host both set out to accomplish the same goal of allowing users to print remotely. Octo print isn't so much of a slicer as it is a platform for monitoring 3D prints remotely. Even though it doesn't offer much in terms of slicing it's still able to do so, but it's remote monitoring abilities are the best out there as of today. The user can add a temperature sensor for monitoring and adjustment and can stop printing from any location if you're connected to the internet. Repetier Host has a better slicing system than Octoprint, and it allows for larger scale remote printing services. For what we need Repetier Host is too much, Octoprint may turn out to be useful for cutting down on prototyping time.

SelfCAD is a very well-rounded slicer compared to the rest of the selections for slicer software. It's an all-in-one tool that allows you to create 3D models from scratch like CAD software but can import that model within the same system and use slicing software to prepare it for printing. The streamlined and simplistic UI makes it easy to learn SelfCAD. No download is needed since it's entirely a cloud-based service but needs an internet connection to work with it. For modeling, it uses a system like FreeCAD that is integrated within the software

itself. After modeling, SelfCAD provides a few options for slicing, but it's not as in depth as some of the slicers previously mentioned. Furthermore, it is compatible with almost all fused deposition modeling printers. There are two downsides to SelfCAD: it's a monthly subscription of \$14.99 a month, and its lack of slicing features.

As a final verdict, taking into consideration that the team has access to a personal Prusa i3 MK3S+ we used Prusa Slicer. Although, it should be noted that slicers like Cura and Slic3r were also good picks due to potentially having better support options if any overhangs need support while printing. Octoprint integrated into the Prusa printer would allow for less time wasted on failed prints since monitoring it live gives us the option to cancel the print before any time is wasted.

6.2 Hardware Design

6.2.1 Robot Design

To have Trash-E pick up cups, it must have a chassis, an arm to pick up and grip the cups, a place to store them, and a way to move around. This section will discuss the possible designs that Trash-E can utilize to complete the task.

6.2.2 Design Overview

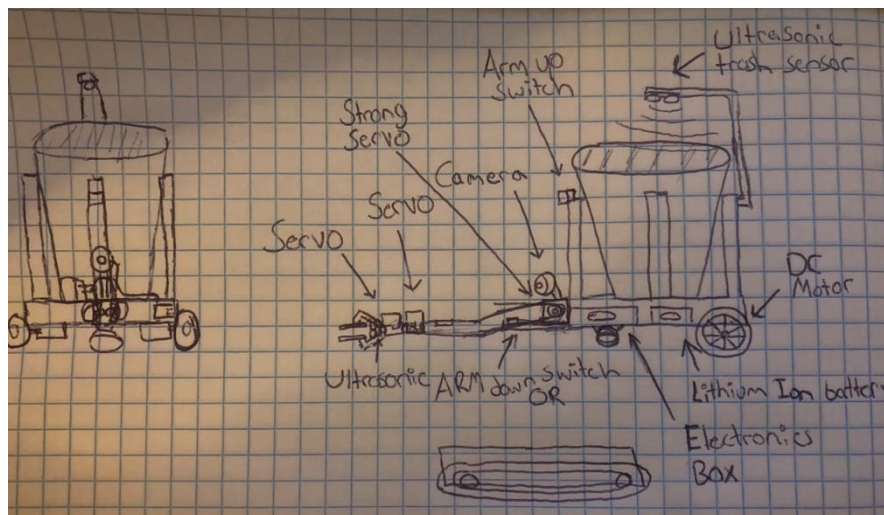


Figure 29: Project Illustration by Alex Rizk

6.2.2.1 Arm and Gripper Design 1

As shown in Figure 29, Design 1 would utilize a straight arm powered by a stepper motor that would drive the arm vertically until it hovers above the garbage bin. At the end of the arm would be a gripping device utilizing a ball and socket joint that can pick up a plastic cup in any position after Trash-E drives up to it. After driving up to the cup the open gripper, using the gear system, will close around the cup

with the action of a miniature servo. Also, at the end of the arm would be an ultrasonic sensor to aid in figuring out the distance of the trash in front of the gripper, this way knowing when to close the gripper would be easy to determine.

6.2.2.2 Arm and Gripper Design 2

Design 2 differs with the arm implementation. Instead of one long arm, it would be broken up into two segments with a motor controlling each segment. At the end of the arm would be the gripper. Two motors would control the gripper: one to open and close and one to rotate the pincers. This would allow the arm to center itself to the middle of the cup and grab it based on the orientation of the cup. This design is a stretch goal as it requires more hardware, using more motors, and software development to figure out how it should orient itself.

6.2.2.3 Arm and Gripper Design 3

Despite being a more experimental, design 3 would have a soft robotics themed gripper. Using soft robotics, the main advantage over traditional servo-gear based robotic grippers is its flexibility. This gripper would consist of two appendages with hollow cores that allow for air to be pumped into and out, this causes a closure and release motion. Direction of movement is determined by ridges on the back of it that push it in an inward motion when inflated. As mentioned before, when the core is filled with air, the appendage would move in an inward motion and conform to the object it is grabbing shown in Figure 30. With a textured adhesive grip, it could potentially grab any object we set out for it. it would allow for greater operability later down the line.

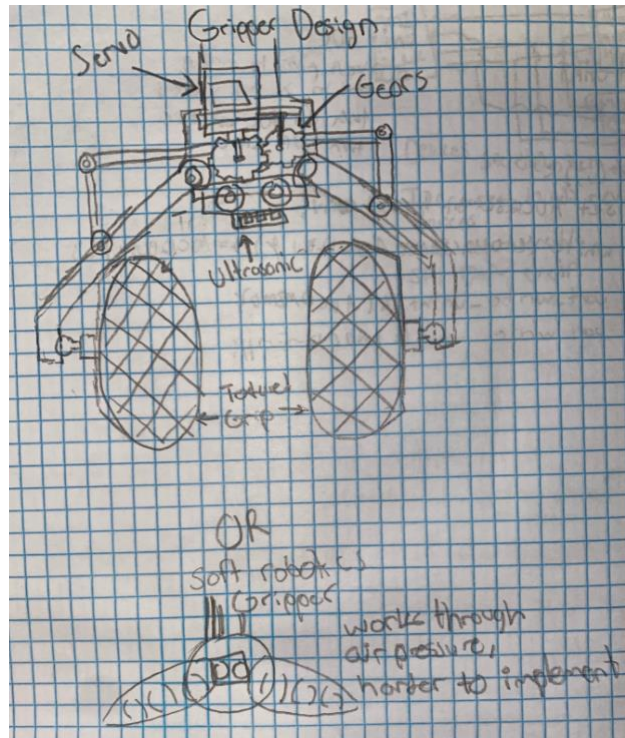


Figure 30: Prototype designs for textured gripper and soft robotics gripper for the purpose of picking up litter

Components and tools needed to create the mold: cardboard, box cutter, hot glue, scotch tape, and gloves. Though using cardboard would be the cheapest option, 3D printing would be available for a reusable mold. The downside to this is that it makes prototyping very time consuming since we would need to print a new model every time. Due to this, we wanted to reinvent the approach. Materials needed to make the gripper: Smooth on Ecoflex 00-30 (rubber), scissors, printer paper, a nail, water bottle with lid that is at least one liter, 1/8 inch outside diameter pneumatic tubing, and a curling ribbon.

First, to set up the cardboard mold we would fashion two small rectangles out of 3 x .5-inch pieces of cardboard by stacking two of them and scotch taping them together then covering with a thin layer of hot glue, these are put aside for later use. To make the walls of the mold four pieces of rectangular cardboard 4 x .75 inches are cut and the edges are taped and the entirety of the cardboard is covered in a thin layer of hot glue to smooth it out. Taking a 9 x 9-inch plane of cardboard and drawing a dot in the center then hot gluing and placing the previously made cardboard blocks $\frac{3}{4}$ inches away from it at 180 degrees and -180 degrees mirroring each other. Next, place the walls and draw some lines that are $\frac{1}{4}$ inches away from the blocks previously glued down, and then place the walls on those lines and glue it. Between the two walls at each end will be a $\frac{1}{4}$ inch gap which should be filled by an appropriately sized piece of cardboard matching the height of the previously placed walls. Finishing the air chambers, connect the two rectangular blocks with a generous channel of glue, avoiding the walls, and keep applying layers of glue until half the height of the block. Finally, seal any remaining cracks/gaps in the

walls of the mold to make sure the rubber mixture doesn't leak out later down the line, and the mold for the top part of the gripper is done.

Following this, the rubber pouring mix needs uncured Ecoflex 00-30, which is a mild skin irritant, so as a safety precaution, gloves and safety goggles are needed. Pour the mold very slowly into the mold until the air chamber blocks are submerged by at least 1/8 inches of the rubber mix while keeping an eye out for potential leaks. In the case of a leak a paper towel should be placed over the area to seal the gap, and if the mixture seems uneven some small object should be placed underneath the cardboard to level the mixture. It should take around 4 hours for the mixture to completely cure. Once the mix has finished curing it should be pulled out by its outside edges towards the center until the mold is removed entirely.

For the second mold, A trace of the first mold made of hot glue should be made with a space of ¼ inches from the newly made rubber gripper; two to three layers of glue should make walls high enough for the next mold. Similarly, take the recently made appendages and trace them on a piece of paper. Making a smaller new batch of Ecoflex, use it to cover just the bottom of the new mold, then place the paper in the center of the mold and fill the rest of it with the remaining Ecoflex. After both bottom and top pieces are made, check for imperfections, and if there are any such as gaps or holes, fill them with some more rubber and spread it evenly. To put the two pieces together, place the thinner piece back into its mold and put some more Ecoflex evenly across it, then, place the larger piece to seal them together ensuring that there is still an air chamber. Finally, after waiting another four hours for the rubber to cure, the two pieces will be melded together to form one cohesive soft robotics gripper.

With the completion of the gripper the air pump needs to be made, for this to be done hands free it needs a small electric air pump or a servo that controls a liter sized syringe. To hook up the pump to the gripper we will need the components mentioned earlier. Connecting the tubing by using hot glue along with some shrink wrap, or with other adhesive along with a pneumatic air seal. Going back to the gripper, pierce the center of the gripper ensuring it lines up with the central air cavity and insert the tubing, blow air into the gripper to test inflation. After ensuring air can cleanly enter and leave the gripper, take it out. Taking two strips of about 20 inches of curling ribbon wrap it around up one of the appendages counterclockwise with the first strip, and with the second strip wrap it clockwise up the same appendage ensuring they overlap on the top and cross on the bottom. When done wrapping that appendage, tie up the ends and remove the excess ribbon; repeat the process for the other side. When the wrapping is done, take the end of the tubing that connects to the gripper, cover it with a layer of Ecoflex keeping it away from the hole, and reinsert it back into the hole previously made to let it cure. Once finished curing, we inflate the gripper to test how the curling ribbon interacts and ensure that a pinching motion occurs, if not, then testing with different ribbon configurations is needed until the desired result is reached. Now, curling ribbons tend to slip around when force is applied, so a friction layer is

needed on the end picking up objects. Taking out the mold for the small layer, fill it with a thin layer of Ecoflex and place the end picking up object in it to cover the curling ribbons. After waiting for it to cure for another 4 hours, the robotic gripper is done and finally able to be used.

Overall, this is a low cost and effective way to make an entry level soft robotics gripper. Despite the several hours this takes to develop, it saves a considerable amount of time compared to designing a new gripper and waiting hours for it to print. The only downside is that it requires the use of a portable air compressor which would greatly effect Trash-E's weight and power consumption.

6.2.2.4 Unique Materials and Gripper Design 4

The fourth gripper design, an open-source project by Marc Schömann [22], is a hybrid between designs three and one. It takes the advantages of both designs and combines them into one gripper. From design one it pulls the ease of manufacturing because it can be 3D printed, and from two it somewhat takes the flexibility of an entirely soft robotics gripper. It is composed of two parts: the solid layer which controls the contractions of the limbs and the flexible layer that allows the limbs to return to their original position. As a result, two different plastic filaments need to be used to realize the design; the flexible one being either TPE or TPU and the solid one being PETG or nylon (Talked about under the materials sections).

As can be seen in Figure 31, the three-prong design is flat when the servo is not actuated. However, in Figure 32, the gripper bends and can easily grip the object.



Figure 31: Flexible gripper at rest (Courtesy Layershift)



Figure 32: Flexible Gripper Actuating (Courtesy Layershift)

To continue, TPE (thermoplastic elastomer) filament, is hard plastic blended with rubber. Its elasticity can vary greatly depending on its specific composition, different versions of TPE serve different purposes. For example, one variation of TPE may have a hardness suitable for high impact and friction while the other may have more of a stretchy characteristic like rubber products. Furthermore, its rubbery like nature also makes it great for a gripper limb, allowing for a better grip on objects. Rigidity (ability to return to its original shape) of TPE isn't the best, TPU is better in this category, but depending on the type used it still has some presence. Due to its very flexible nature TPE happens to be difficult to print. It needs temperatures of around 230°C for the extruder head, a print bed temperature of 110°C (very hot for print bed temps) and must be printed very slowly making hard to prototype with. Luckily, the Prusa i3 MK3S+ 3D printer we used has a direct drive extruder which can handle TPE well enough to print without jams.

TPU (thermoplastic polyurethane) filament is a category of TPE that has several major differences to standard TPE. TPU tends to be harder than TPE, meaning it can resist surface deformities better than TPE. Its shore hardness classifies it as being a medium hard rubber and hard runner which is like a tire head or shopping wheel care respectively. Despite its hardness it still manages to be very elastic which gives it a wider range of applications. Mentioned before, TPU has a greater rigidity compared TPE allowing to recover from deformations better, for the soft robotics gripper this can come in handy when we need to retract the limbs. Though, TPU tends to have a smoother texture which would lead to possible troubles when it comes to gripping objects. TPU also has a higher durability compared to similar TPEs because it shrinks at a slower rate. Due to its stiffness, TPU is much easier to print, but not as easy as non-flexible plastic filaments. Printing properties are like TPE, but slightly more manageable with lower temperatures. It still needs to be printed slowly to ensure print quality is up to snuff.

The hybrid gripper would work with either thermoplastic, so other factors must be taken into consideration to figure out which plastic would fit better into the scope of this project. A comparison of TPE and TPU prices shows that, in general, a roll of TPE is 10 to 20 dollars cheaper than most rolls TPU. The effort of printing is very similar so in this way it does not matter. Durability of the soft part of the gripper is not a concern, so the main advantages of TPU don't seem that appealing, while the rubber like grip of TPE would benefit us more. Overall, TPE would seem to be ideal filament to use in the soft part of the gripper shown below in Figure 33 along with its dimensions in Figure 34.

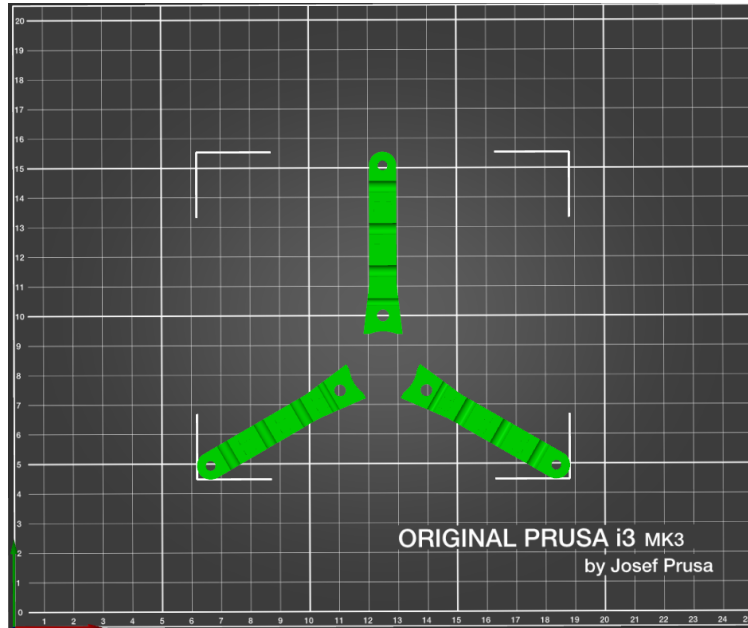


Figure 33: 3D Model of the Soft Portion of the Gripper

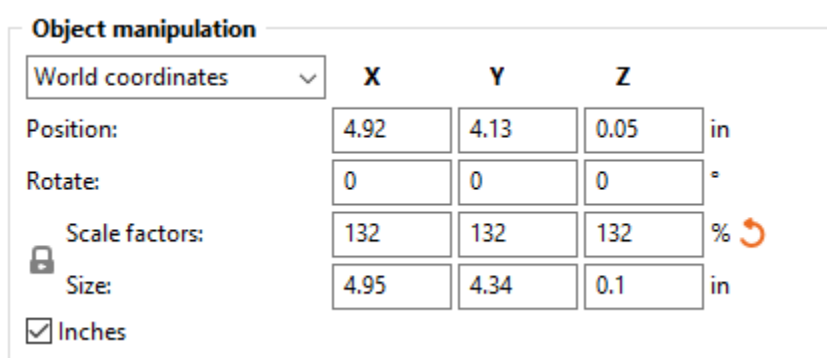


Figure 34: 3D Model Dimensions of the Soft Portion of the Gripper

6.2.2.5 Chassis Considerations for all Designs

Two different designs were taken into consideration for the chassis of Trash-E. The first design consists of two wheels towards the back of the chassis while one swivel wheel is positioned in the front forming a triangular shape. In this case, there

would be a servo on each rear wheel for movement left, right, and forward. The second design has more of a car shape with four wheels and a compartment below the operating plate to store the electronics. Placement of the driving servos can be placed in two ways with this design: one servo driving the left rear wheel and the other driving the right front wheel, or vice versa (this allows for a swivel movement like the triangular design).

Weight is a big consideration for the design of the chassis, as it would cut down on power consumption by reducing the watts needed for the motors providing movement. Without having to create a separate compartment for the electronics we can place everything on the front plate; this gives us easy access to the components and makes sufficient airflow to the Nvidia Jetson easy to achieve. Furthermore, it saves on plastic, money, and time since we would need to take the time to figure out a compartment system and use up more plastic trying to print it. On the other hand, its greatest disadvantage is its stability. Leaving the front part of the chassis reliant on a swivel wheel may introduce some wiggle to the movement of the robot. Having this unnecessary movement could lead to further power consumption due to making corrections during navigation. Calibration of the servos may be difficult since the triangular wheel orientation may introduce unpredictable movements.

On the other hand, the four-wheel design, though using a bit more power, comes with its own advantages. The first advantage of this design would be its clean design which would protect the components in its specially made compartment. As mentioned before, there would only be a design issue having to figure out how to achieve enough airflow for the jetson to not overheat during operation. This could be remedied by adding some holes near the jetson for intake as well as some for the expulsion of hot air. Unlike the triangular design that is unstable, the four-wheel design, can maintain a relatively stable movement during its operation. Due to having a wheel on each side, each opposing wheel acts as a sort of counterbalance when rotating. Whereas, with the triangular design it may wiggle. In turn, the more stable design would make it easier to calibrate the motors, as we don't have to worry about unpredictable movements during operation.

The bucket design, although simple, needs some attention to ensure it's able to contain what we are picking up. To formulate a proper design, we needed to take the dimensions of a red solo cup into account. A red solo cups dimension is irregular since the rim of the cup is larger in diameter than the base. Height was also a big concern. The dimension of the rim is 3 5/8", height is 4 5/8", and the base is 2 1/4". To determine the length and width of the bucket we need to specify how many cups need to be picked up. For the scope of this project, we accounted for around 12 cups taking the height and rim to simplify measurements. The cups would not be stacked since they'll be dropped into the bucket, so we could get a rough estimate of the dimension of the bucket. Schematics for the bucket have been drawn to give guidance for 3D modeling it shown in Figures 35 and 36.

1/2 Scale Cup: 58.8 x 46 mm Size: 2.314 x 1.81 inches Bucket Top-down View Total: 109 x 134.6 mm or 4.291 x 5.299 inches

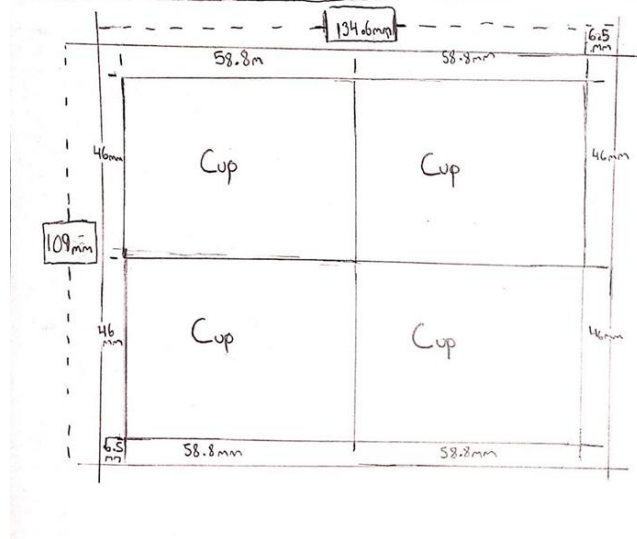


Figure 35: Top-down view schematic for trash bucket

1/2 Scale Total width: 134.6 mm Bucket Side View Total: 134.6 x 177 mm

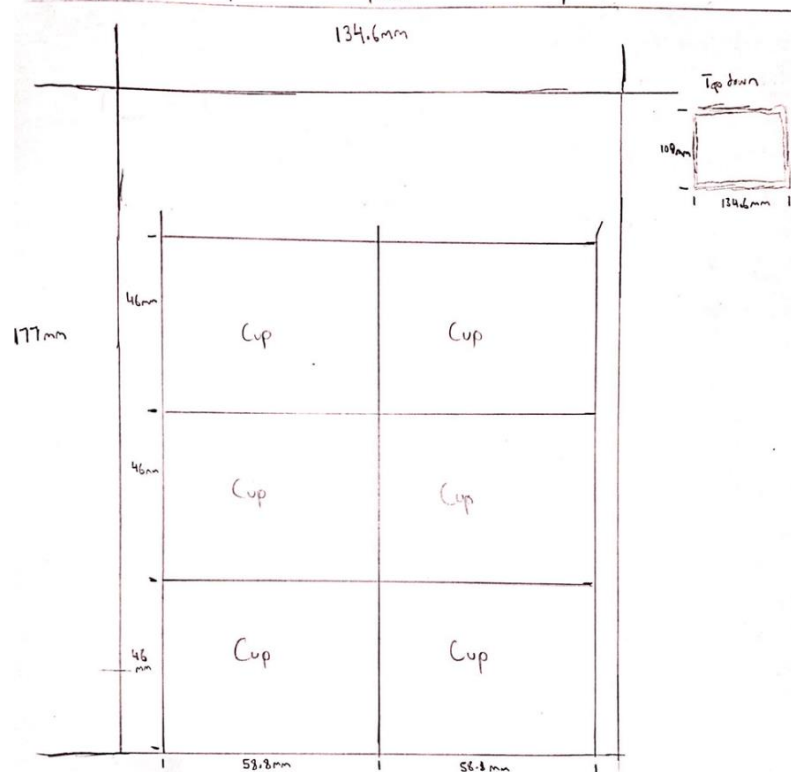


Figure 36: Side view schematic for bucket

6.2.2.6 Final Arm Design

The biggest flaw in previous arm designs mentioned before is that they don't allow for tweaking of the shape and length of the arm. Rather than having to print several versions of the arm we thought of a modular design. Like Legos, the pieces of the

arm shown in Figure 37 will interlock. As a result, it forms a tight seal that resists vertical forces while somewhat resisting horizontal ones. More specifically, the three forces we need to account for when designing these joints: Friction, Tension, and Shear. Furthermore, using PETG for the choice of plastic for each of these blocks will ensure a durable design.

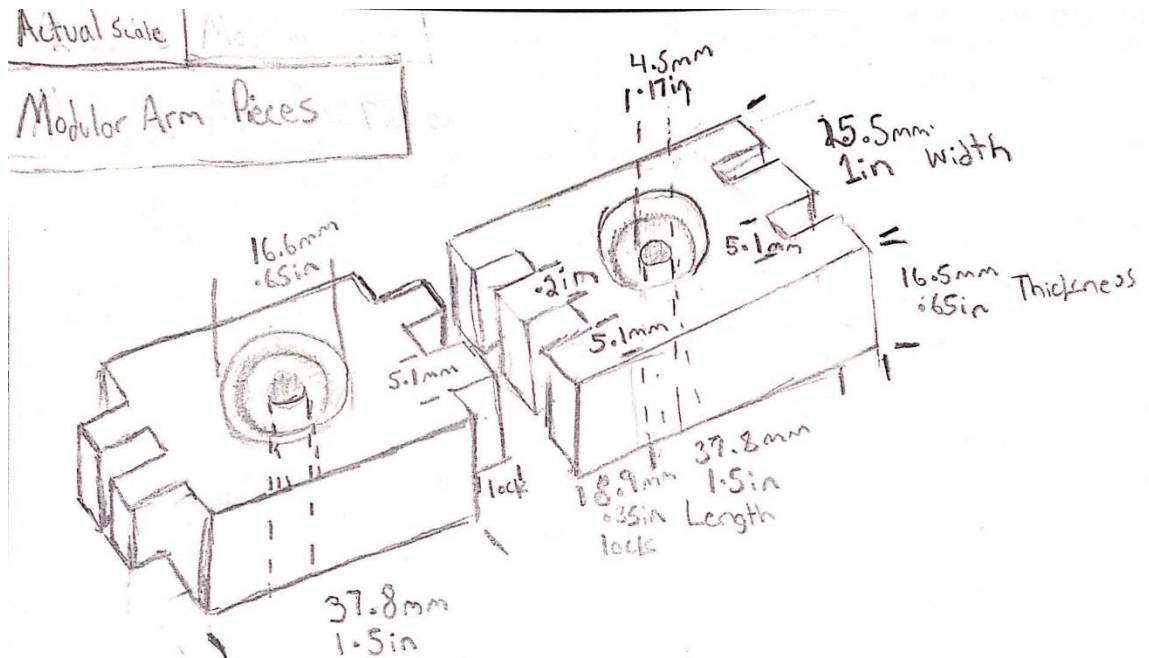


Figure 37: Illustration of modular arm pieces

Length of the arm can be increased or decreased based on the number of pieces interlocked shown in Figure 38. Attaching each piece can easily be done with one bolt, two washers to distribute force evenly, and one hex nut to tighten it. In the center, the screw holes open several configuration and reinforcement possibilities that can also be seen in the figure. For the arm to operate successfully, the gripper must be hovering over the bucket to ensure that the litter can be dropped without issue. The arm would need various tweaks to be in the proper position. Testing that follows would be to start with a base arm size, check if it's above the trash bucket, and if it is no adjustment is needed. Otherwise, adjust arm length and orientation.

Price wise, this arm is inexpensive since PETG is relatively cheap and easy to work with. The greatest issue would be the time it would take to print each individual piece one by one, or in batches. Dimensions being 1.5 x 1 x .65 inches, the estimated print time on each piece should be around 15 minutes. If each piece takes 15 minutes and we need to print 20 pieces, it would take around 5 hours (granted there are no issues printing within those 5 hours). Although it would take 5 hours it would be worth it since the design is modular and it would save time compared to other designs.

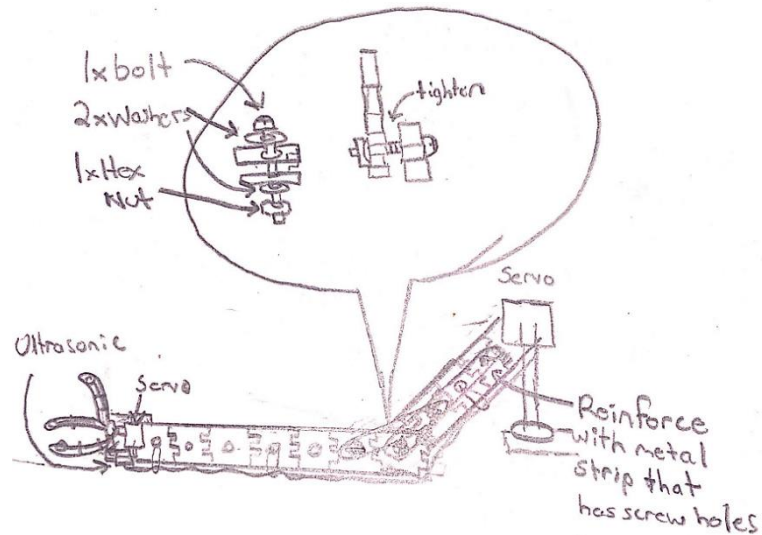


Figure 38: Illustration of modular arm as one piece

When the final length and configuration is determined adhesives can be used to further solidify the connections between the pieces. Ideally, plastic bonding glue would be used to ensure that the plastic is permanently bonded at a molecular level. Other glues will work such as super glue, polyurethane, hot glue, and epoxies. Overall, this design saves time and money in the case changes occur in other components. For this to be achievable, attention to print settings is paramount since each piece must ensure a snug fit, otherwise the design may fall apart. Several configurations for the Prusa i3 MK3S+ are available online to prevent the hassle of tweaking the settings related to this.

The final design consideration of the arm is to create an arm that has two separate pieces that can be connected using screws which can be seen in Figure 39. After determining the correct length we needed, we cut the arm down to size and updated the corresponding code to allow the robot to detect a cup in the right distance.

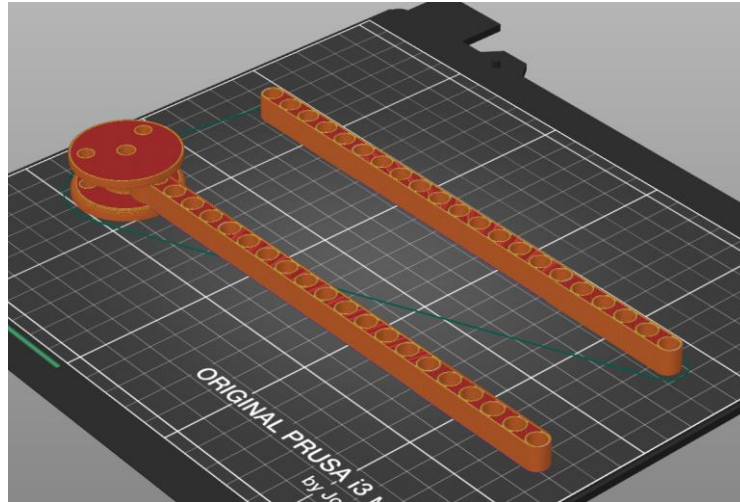


Figure 39: Final Arm Design

6.2.2.7 Material Considerations

There is a plethora of plastics available on the market and can be easily found on several websites for a reasonable price. The most used plastics available are: ABS, PLA, ASA, PET, PETG, Polycarbonate (PC), high performance materials (PEEK, PEKK, ULTEM, etc.), PP, Nylon, composite material, hybrid material, Alumide, and resins. Research was done for each of these plastics to see which one fit the best into the scope of the project. As a side note, no person on the team has experience with materials or mechanical engineering so careful consideration into each of these materials had to be made.

To start, acrylonitrile butadiene styrene, or ABS for short, is one of the most used plastics used in 3D printing. Its usages can be seen in appliances, mobile phone cases, and car bodywork. Being a thermoplastic, it is resistant to drops/shocks and is not brittle making it bendable, in turn, reducing the chances of cracking and breakage. In terms of recyclability ABS is easily reusable when run through a shredder and filament extruder, and it can also be welded together with other pieces of ABS with chemicals like alcohol and acetone. This plastic can even withstand temperatures between -20°C (-4°F) to 80°C (176°F). Though, this plastic is not biodegradable and shrinks while in contact with open air causing warping during printing. This can be prevented by taking the following precautions: having a chamber with hot ambient air temperatures, complete restriction of airflow to the outside, a heated build platform, and a nozzle temperature of 230°C (446°F) and 260°C (500°F) for printing. It's also important to keep in mind that ABS while heated/printing has one of the heavier particle emissions of microplastics out of all plastics, so an enclosed printing space is essential. Even though ABS is durable and checks all the requirements for the components of Trash-E it just requires too much to manufacture. To add, equipment for safe and easy ABS printing is not available to us, we would have to risk printing on a standard open air 3D printer.

Polylactic acid or PLA's greatest strength is its environmentally friendly properties and printability, but it stops there. Unlike ABS, PLA is biodegradable as well as easily recyclable, it is more ethical to print with. Moreover, it is one of the easiest plastics to print with as it doesn't require a heated build plate or as many restrictions as ABS. It prints at much lower temperatures between 190°C (374°F) to 230°C (446°F), which is virtually possible on all 3D printers. In terms of manipulation, PLA is very difficult to meld due to cooling and solidifying very quickly, but in our case, this is not an issue. When in contact with water it deteriorates very slowly over time, and in the case of Florida weathers year-round humidity would not see long term use. More of a decorative plastic, PLA comes in many colors, but is on the lower end of durability when it comes to 3D printable plastics.

Acrylonitrile styrene acrylate (ASA) is extremely like ABS in its properties with the benefit of UV resistance. This plastic would be great for use outdoors with the durability and heat resistance of ABS. Despite having all these benefits, its production cost is even higher than ABS due to a component added to grant its UV resistance. Styrene and microplastics are emitted during printing which can cause symptoms of styrene poisoning if one neglects to print it in an enclosed space. This plastic would be a great choice if we had planned Trash-E to operate outdoors, but the proper equipment to work with ASA is not available to us.

Polyethylene terephthalate (PET) like PLA is easy to work with but lacks the biodegradability. PET is somewhat rigid making it good for pieces that are not exposed to constant movement or collisions. It has great resistance to various chemicals making it good for contact with food, it can be seen in use with water bottles, synthetic fibers, and similar products. Unlike ABS it releases a minuscule amount of plastic during printing, as well as being odorless it is very safe to print in any 3D printer. Its biggest downside is its overall brittleness and fragility, it cannot handle high temperatures or impacts. Based off the previous statement, Trash-E being an autonomous robot, may encounter collisions during testing, so PET would not be the best choice since we want something durable and reusable. Luckily, its cousin glycolized polyester (PETG) can make up for its weakness while still retaining the same safety. It combines the durability of ABS and the simplicity of PLA. The glycol added improves the ductility, chemical resistance, transparency, hardness, and impact resistance. Though it does require a heated build plate along with extrusion temperatures of 220°C (428°F) to 260°C (500°F), it's nothing most standard 3D printers can't accomplish. PETG is one of the easiest and durable 3D printable plastics and it will most likely be used for the entirety of the chassis as it is durable, easily printable, and affordable.

Polycarbonate (PC) would be one of the greatest plastics to use for our application, but it has many pitfalls if not managed properly. This plastic is extremely durable, lightweight, as well as being able to withstand temperatures of 150°C (302°F). It's so durable that it's used in the production of bulletproof glass and other glass products. Although the durability is great, production and safety are just too taxing.

It releases bisphenol A (BPA) particles which can have several negative side effects. It is sensitive to humidity and UV rays making it practically unusable outdoors. Troubles during printing include trouble sticking to the build plate resulting in print failures, warping (peeling) from the build plate, and pretty much all the difficulties of printing ABS. High temperatures are needed not only for the nozzle, 260°C (500°F) to 310°C (590°), but also for the build plate that would have to reach temps of 80°C (176°F) to 120°C (248°F). On top of this, a sealed chamber is required if you want the optimal and safe printing results. Overall, this plastic is very far out of reach to work with and would be extremely overdeveloped if we were to implement this plastic for Trash-E.

Several high-performance polymers like polyaryletherketones (PAEK) and polyetherimides (PEI) are great choices as it is one of most durable and multipurpose plastics out there. In terms of physical and temperature resistance it can be higher than PC, and in most cases it is. Mostly found in the medical, aerospace, automotive, and military sectors it's almost the end all be all industrial plastics. Despite the plastic being an amazing all-rounder, it is just not meant to be used by the average person as expensive equipment is needed to work with it. To print PEI/PAEK nozzle temperatures need to reach temperatures over 350°C (660°F) along with build plate temperatures needing to reach 230°C (450°F). Even more, an enclosed chamber is needed to properly balance the temperature within, needing very capable cooling systems. Being one of the best plastics out there the price is also high (for good reason) with prices of PEEK reaching 195 dollars per 250 grams, which is possibly one of the most expensive options. Over time the barrier of entry for these kinds of plastics have lowered, but it is not at the point where a group of students can afford the cost or facilities to work with this plastic unless sponsored and given access to the proper equipment. Hence, high-performance polymers are completely infeasible for use with Trash-E.

An alternative to PETG, Polypropylene (PP) has similar properties, but it has great interlayer adhesion allowing for it to stretch before breaking. Its overall cohesion allows it to be a better version of PETG with the ability to resist abrasions and shocks while still maintaining good rigidity. Much of the time it's used in the automotive industry but is used in much of our everyday objects since it's non-toxic and poses no risk from using it. Printing with it is easier than with PETG with print temperatures of 220°C (428°F) to 240°C (464°F) and can even be printed without a heat bed (it helps though). It's hard to find any downsides to this plastic as it's easily accessible and printable through normal means, this is a great contender instead of PETG. Given that price is a known constraint, a spool of PP (40\$) is almost double the price of a spool of PETG (20\$). Despite the great properties, PETG is still the number one choice for the chassis.

Nylon finds a good middle ground between user friendly and durable, as it has a crystalline structure consisting of carbon. Alternatively, it is much easier to work with than PC, it removes the hassle of having to deal with the absurdly high temperatures and safety risks during printing while still maintaining a durability that

rivals PP. Because nylon is composed of carbon it has amazing temperature resistant properties being better than PC at 180°C (356°F). It's also environmentally friendly since it's bio sourced from castor oil. Overall, it's a highly stable material with one weakness, and that would be its high propensity to absorb humidity from the surrounding air and a heated chamber of at least 40°C is needed for a successful print. This one downside removes it from outdoor applications, but for a lot of companies, nylon is the go-to plastic for high end prototyping. The extra hassle from having to store nylon in a dry box and having a heated chamber isn't worth it, PETG seems to still be the best option so far. Although, it may be considered for the internal structure of the hybrid soft robotics gripper in Figure 40 since it's very flexible when printed thinly.

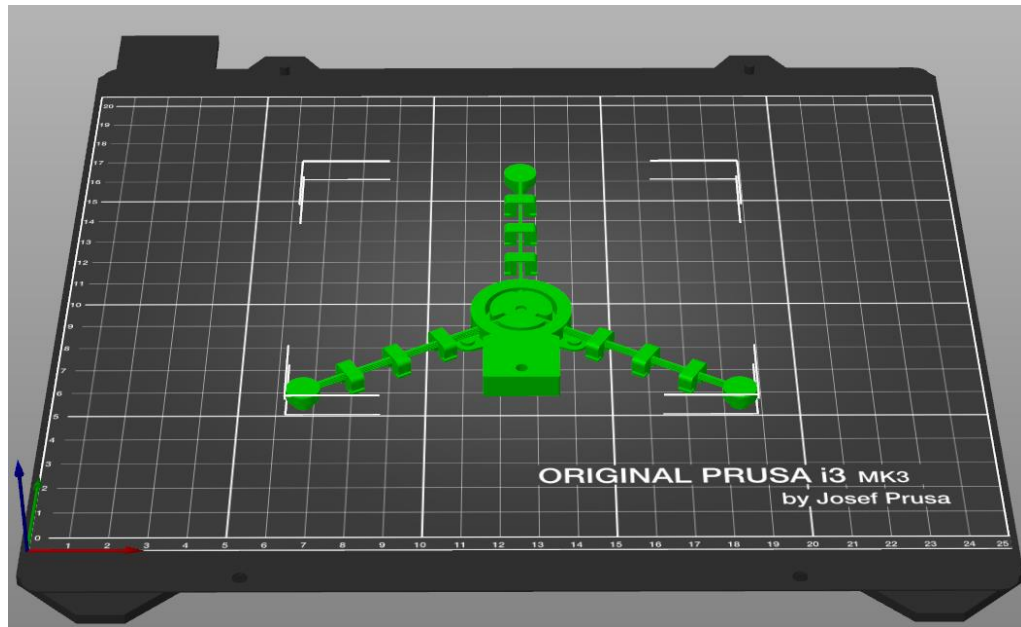


Figure 40: 3D model of the solid component of hybrid gripper

Object manipulation			
World coordinates	X	Y	Z
Position:	4.95	4.35	0.13 in
Rotate:	0	0	0 °
Scale factors:	132	132	132 % ↻
Size:	4.95	4.34	0.26 in
<input checked="" type="checkbox"/> Inches			

Figure 41: Dimensions of hybrid gripper

Composite material consists of various plastics like PLA, ABS, or nylon that are mixed with short fibers of (in most cases) carbon fiber or various other materials to achieve certain unique properties. This category is just too expansive to delve into, there are just too many combinations that could be made with composite materials. There may be one material out there that has the properties we need, but most

composite materials are priced a bit heavier than their non composite counterparts. Besides, carbon fibers mixed into any of the aforementioned plastics can deteriorate the nozzle of a 3D printer, starching it as its extruded. Taking into consideration money and time it's not worth considering as we just need a material that is relatively durable while still being able to easily work with and print.

Delving into hybrid materials, they suffer from the same issue as composite materials. Many hybrid filaments, or might I say the most popular, are typically composed of PLA mixed with various materials to achieve a certain look or mechanical property. For example, PLA can be mix with several wood products like wood dust, oak, mahogany, etc. to achieve a look that is like wood but retains the properties of PLA. Metal like copper or brass can also be mixed in to make a print have conductive properties or make it look like either metal. Hybrid filaments can also cause damage to nozzles, so special reinforced nozzles are needed to print this material. Like before, adding extra material to filament incurs extra costs and requirements to print, so to remove unnecessary complexity from the project we will avoid hybrid materials.

Alumide would be a decent alternative to almost everything on this list as it shares the properties of aluminum. Its temperature resistant and is great for use on small models that require a lot of detail, great for use in creating replacement parts. Though, unlike every other material in this section it is not printable through normal means. A selective laser sintering (SLS) machine is needed to even work with it. On top of this, it's not purchasable through Amazon and seems to be only available to the manufacturing industry. Basically, this material is inaccessible to average students and not to even mention the expensive equipment needed. Alumide is unusable for Trash-E.

The last considered material is resins, the best way to describe this would be to say it has the same properties as ABS with half the benefits. It can only be used in special printers that utilize UV light to harden the resin into whatever shape needed. The greatest advantage is the amount of detail achievable by resin, the results are comparable to injection molding. Resin printers are cheap, cheaper than some extruder printers, but working with them is a hassle. There are several kinds of resins that can be used for many kinds of applications such as dentistry and various hobbies, but there really isn't a use case for this project.

In summation, after running through the available 3D printable plastics such as ABS, PLA, ASA, PET, PETG, PC, high performance materials, PP, Nylon, composite material, hybrid material, Alumide, and resins we now have a greater understanding of 3D printing. Considering the various pros and cons of all the mentioned materials PETG offers the best middle ground in terms of price, strength, accessibility, and printability. PETG was the plastic used in the prototyping, build, and testing sections. It came close between PP and Nylon because they were the best affordable options when it came to what we were looking for, but the main deciding factors among these options was price. In this

case, PETG is 20 dollars, PP is 40 dollars, and nylon is 30 dollars. With a budget of \$400, minimizing our expenses is paramount.

6.2.2.8 Overall Design

In the end, the three-wheel design shown in Figure 42 is the design we chose for Trash-E. We chose a two-pronged pincer gripper for the design to allow for a wide opening as shown in Figure 42 below. Rubber bands were also added instead of utilizing a rubber spray to allow for a stronger force of friction between the pincers and the cup. The arm was created to be cut shorter after we determined the best length with testing. This length ended up being six inches. Considerations for weight distribution of components were also considered to have adequate maneuverability. Furthermore, several other significant additions to the chassis were made. An 8.5-inch elevated stand for the LiDAR sensor was made for the purpose of giving it a wide enough clearance from the rest of the components to get a clear scan of the room. Trash-E's sharp corners tend to get caught in tight spaces, so button bumpers are placed on the front two corners to run a protocol to get out of any tight spaces when pressed. Casings for various components such as the PCB's, ultrasonic sensor, and Nvidia Jetson Nano were needed for mounting to the acrylic base. Most components are mounted to Trash-E using adhesives like super glue and hot glue. Finally, since every chassis component except for the acrylic is composed of PETG the overall design is lightweight and durable, weighing only 7 pounds more than meeting our requirements for its minimum weight.

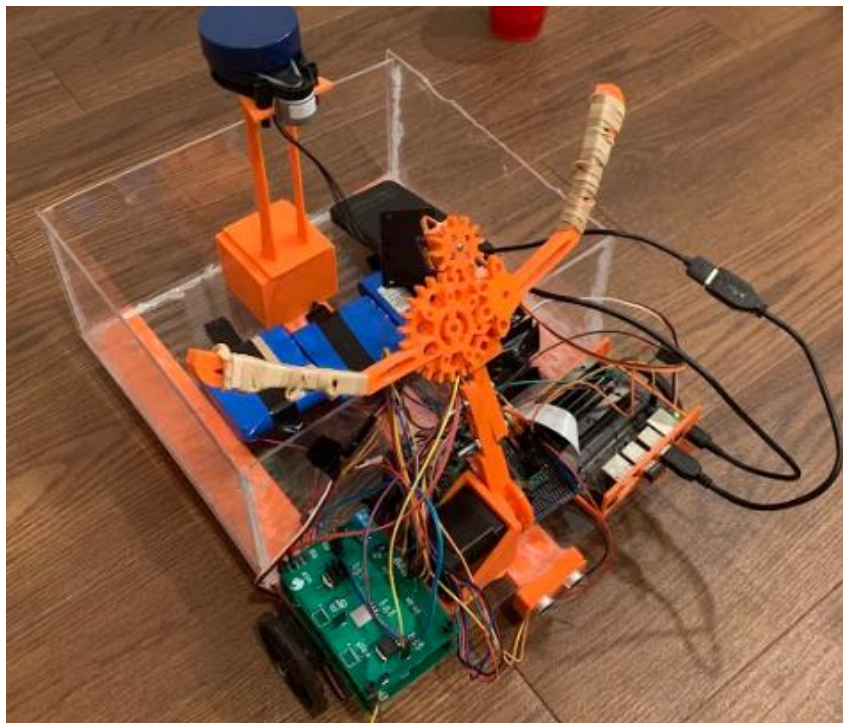


Figure 42: Final Design for Trash-E

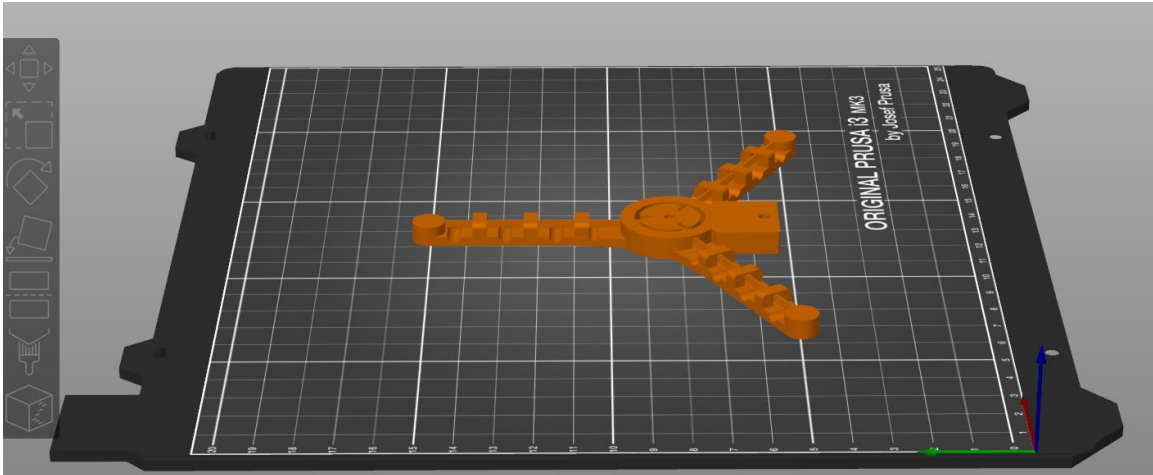


Figure 43: Full design of the hybrid gripper

6.2.3 Motors

6.2.3.1 Stepper Motors

Stepper motors are very useful for moving in precise increments while also having high torque at low power. It can achieve these precise movements because each stepper motor has a specific step angle which it turns every time the motor moves one step. The Twotrees Nema 17 motor has a step angle of 1.8° . Using the formula

$$\text{Steps per revolution} = \frac{360}{\text{Step angle}}$$

we can rotate the motor 200 times until it has completed one full revolution. For the tradeoff of torque, we can increase the number of total steps by dividing the step angle even further. The movement will be more precise, but the max weight we can lift will be reduced. To calculate the microstep we use the following formula:

$$\text{Microstep} = \frac{\text{Step Angle}}{\text{Step Divisor}}$$

If we wanted to divide the current step angle by a step divisor of 16, we would have a new step angle of 0.1125° and achieve 3,200 total steps instead of the original 200. These motors best fit the application of moving the arm vertically when it has picked up a cup and then when it needs to return to rest. We knew how many steps it has moved since being at rest so it could be returned exactly to the same place. We chose to divide the step angle by a divisor of 4 to achieve smooth movement without the loss of too much torque.

6.2.3.2 Servo Motors

Two types of servos were utilized in Trash-E: positional and continuous. Positional servos allow the user to control the position of the servo using a potentiometer driven by a pulse-width modulation (PWM). Based on how long PWM is high, the

motor can be all the way to the left, right, or somewhere in-between. This type of servo was used for the gripper at the end of the arm to grab the cup and release it.

Continuous servos allow the control of speed and direction but lose the positioning information. Altering the PWM signal will change the speed and direction of the servo rotation. This was utilized in the movement of Trash-E to move it forward, backward, and/or turn.

6.2.4 Motor Driver

The output power of the microcontroller is too small to activate the stepper motors. This requires the use of a motor driver. Motor drivers allow a signal to still be sent from a microcontroller on how it should move, but the motor will be powered by an external source. Figure 44 depicts the typical wiring for the A4988 stepper motor driver we used. The motor power supply is powered by the onboard battery.

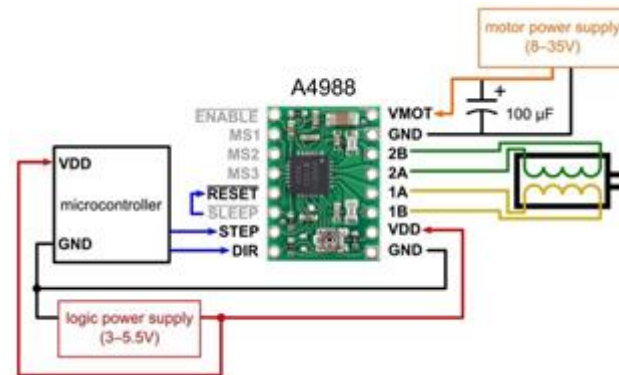


Figure 44: Wiring Diagram for A4988 Motor Driver (Courtesy Polulu)

6.2.5 Ultrasonic Sensor

Determining how far away the cup is from the grabber is necessary for picking it up. If the cup isn't close enough or is too far away, the gripper won't be in the correct position leading to the cup not being picked up. The computer vision guides the robot left and right to go straight into the cup. Once the ultrasonic sensor determines the cup is six inches away, Trash-E stops and proceeds to pick up the cup.

6.2.6 3D Printing

After looking online for a pre-made chassis we could utilize for the base of this project, we could only find options out of our budget that met the size requirements. We utilized 3D printing for the chassis, bucket, arm, gripper, LiDAR stand, component mounts, and chassis support beams.

6.2.7 Power Supply

6.2.7.1 Battery Options

Table 12: Battery Options

Part #	Manufacturer	Availability	Price/Unit (\$)	Capacity (mAh)	Discharge Max Current (A)
35E 18650	Samsung	<input checked="" type="checkbox"/>	\$7.99	3500	8
ICR18650-2600-F	PKCELL	<input checked="" type="checkbox"/>	\$6.00000	2600	3.9
LION-1865-26	Dantona Industries	<input checked="" type="checkbox"/>	\$4.99000	2600	?
ICR18650-2200-F	PKCELL	<input checked="" type="checkbox"/>	\$5.00000	2200	3.3
ICR14430-650-F	PKCELL	<input checked="" type="checkbox"/>	\$3.00000	650	?
PRT-12895	SparkFun Electronics	<input checked="" type="checkbox"/>	\$5.95000	2600	3.9
ASR00050	TinyCircuits	<input checked="" type="checkbox"/>	\$5.95000	2500	?
LI18650JL PROTECTED	Jauch Quartz	<input checked="" type="checkbox"/>	\$13.05000	3250	4.875
MJ1 18650	LG	<input checked="" type="checkbox"/>	\$6.99	3500	10
NCR 18650B Protected	Panasonic	<input checked="" type="checkbox"/>	\$9.99	3400	4.9?
Epoch 18650 Protected	Epoch	<input checked="" type="checkbox"/>	\$9.99	3500	8
35E 18650 - Protected Button Top Battery	Samsung	<input type="checkbox"/>	\$6.99	3500mAh	8

Table 12 showcases several batteries that we considered using. The highlighted batteries are batteries that have battery management systems built into them. For our use case, we believe that 18650 batteries are the best bang for our buck. They

can have high maximum discharge current, and can be used to create a battery pack with our desired voltage output. The batteries with capacities over 3000 mAh are the most expensive from the ones put in our table. The batteries that have a question mark under the Discharge Max Current column are ones that we had trouble finding the datasheet for. The specs they have are shown on the product page, but for some reason or another the datasheets were unavailable and the specs page did not show the discharge current.

We will most likely use batteries that are above 3000 mAh. The batteries in the 2000 mAh or less range were considered because they are cheaper, but to get the same output, we would end up using more batteries, which would cost more. For example, to get 7000 mAh using the ICR18650-2200-F, with the same max discharge that we calculated in Section 3.4, we would need 15 batteries. The total cost would be 75\$, while with the Samsung 35E unprotected, the price is 42\$.

Another decision we had to make was whether or not to use batteries with built in Battery Management System (BMS), buy separate BMS circuits that we can implement, or create our own that we can use. The benefits of having built in BMS in our batteries is that we will not have to worry as much about incorrectly recharging the batteries. We also would not need to worry about designing our own BMS or finding a third party BMS. However, the price was much steeper when the batteries have built-in BMS. Comparing the Samsung 35E and the Epoch Protected battery, the price is 2\$ more. If we bought 6 batteries, the price would end up being 12\$ more. We may have been able to find a third party BMS that costs less than 12\$, or even design our own for less.

Ideally, we would want to use the Samsung 35E 18650 3500mAh 8A - Protected Button Top Battery because it was cheap, with a large capacity, and was protected. However, due to supply chain issues, this battery will not be in stock for the foreseeable future. Therefore, we needed to assume we would not be able to get these batteries at all. If we decided to use batteries that have BMS built in, we would likely have used the Epoch 18650 Protected batteries. The NCR 18650B Protected batteries had good specs, but no datasheet was able to be found for them, so it might have been hard to work with the batteries in the future. The LI18650JL PROTECTED batteries are usable, but they are very expensive. For batteries that do not have BMS built in, the Samsung 35E 18650 3500mAh was the best battery that we could find. As stated above, the other non-protective circuits do not meet the requirements that we need for our use case.

As stated earlier, we were lucky to have been donated batteries by Smart Charging Technologies, a company that Chrizzell Jay interned at the time. These batteries were perfect for our use, being 2600mAh and 14.8V nominal.

6.2.7.2 Battery Management System (BMS)

Table 13: BMS Board Options

Part Name	Price
2Pcs 3S 11.1V 12.6V 25A W/Balance 18650 Li ion Lithium Battery PCB Protection Board	\$9.99
5S 20A 18V 21V Li-Ion Lithium Battery Pack Battery Charger Protection Board Circuit	\$8.88
Anmbest Balancer 4S 16.8V 30A 18650 Charger PCB BMS Protection Board	\$9.49

For the 3 possible BMS boards found in Table 13, we were not confident in using any of them. This is because many of the reviews found for these boards said that the boards were not able to handle the rated currents, and some got obscenely hot even at low current. Furthermore, there was little to no information on the technical specifications of the boards. No datasheets could be found for these boards.

6.2.7.3 Battery Test Plan

At the time, we believed that it would be best to buy batteries with BMS protection. Buying a third-party board to connect will add unnecessary bulk to the interior of our robot. Furthermore, out of the ones that we found, the reviews seemed to indicate that the boards had heat problems, which would cause a large problem within our robot. Creating our own circuit may add more complexon to our robot, because we will need to have an esp32 and create more code to help keep track of the voltage and current.

6.2.7.3.1 Procedure

We could run tests by fully discharging and fully charging the batteries to make sure the capacities and max discharge currents are correct. We can further look at the discharge curve to make sure that the batteries are discharging at a steady rate.

1. Connect battery pack to voltage recording device such as an 34970A Data Acquisition / Data Logger Switch Unit
2. Connect battery to power supply and set power supply to 4.2V
3. Allow battery to reach Nominal Voltage at 4.2V
 1. If we decide to get batteries without BMS, we need to keep careful watch of the batteries when they are charging and discharging because they can very easily catch fire. The batteries with BMS will still need to be monitored regardless.
4. Record time taken to reach maximum charge

5. Connect battery to load (Such as DC Electronic Load) and set discharge rate to expected current draw
6. Record periodic voltages and current with Agilent until battery is depleted (3.0 V)
7. Graph output voltage of the battery over time to make sure battery discharge is sufficient
8. Current should be constant and voltage drop should be constant

We did not need to run these tests, as the batteries donated by Smart Charging Technologies were batteries that had already been tested and were in production at the time.

6.2.8 Voltage Regulator

For our use, we planned to use 12V input from the power supply that we made. We then used voltage regulators to step down the 12V input down to 5V, and 3.3V. The different output voltages will be used to power the different components on the robot such as the webcam, servos, sensors, and microcontrollers. Below are possible configurations using the TPS52903RPJ regulator from Texas Instruments

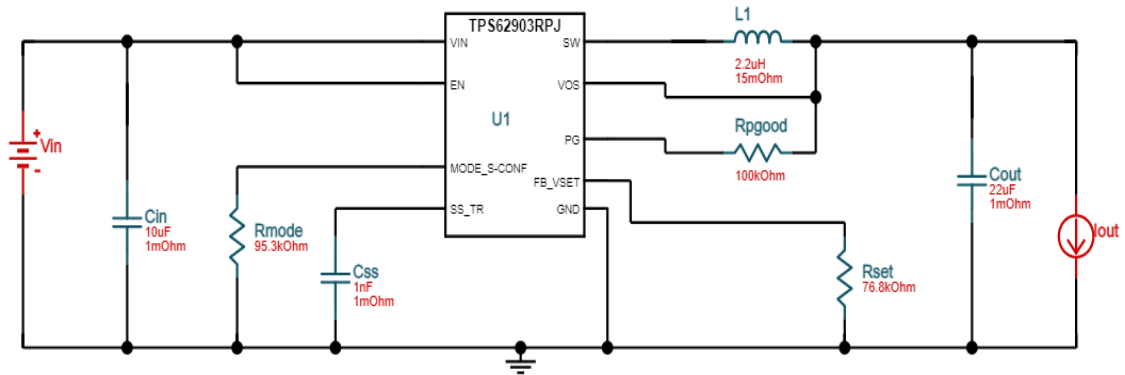


Figure 45: 12V to 5V Buck TPS52903RPJ Voltage Regulator

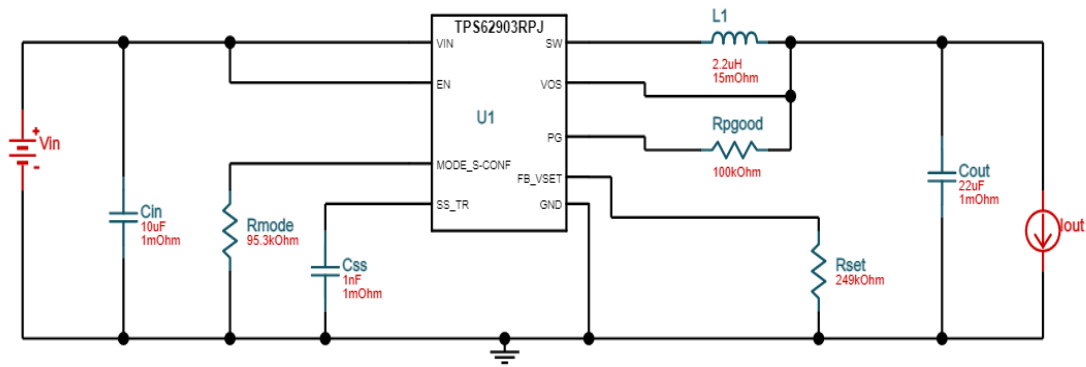


Figure 46: 12V to 3.3V TPS52903RPJ Voltage Regulator

One of the parts that we considered to use for the Voltage Regulator is the TPS62903RPJR from Texas Instruments as shown in Figure 45. We believe that this is a good part because it is low cost and has high efficiency at our load of 5V. Furthermore, as seen in Figure 45 and 46, the circuits between the 12V to 5V and the 12V to 3V were almost identical. By changing the resistance of Rset, we could easily change the output voltage. This was beneficial because we could reduce the variety of different parts that we order. We could also save money by buying the same components aside from Rset.

While we wanted to use the part TPS62903RPJR for a voltage regulator, due to the supply chain limitations from COVID, this part was currently out of stock on all known websites, including TI's, the original manufacturer. We proceeded as if this part will always be out of stock. However, if this part came back into stock, we would consider ordering it along with different regulators to test the efficiency, in case this part ends up working better than the new parts.

After further investigation, most, if not all of the voltage regulators from Texas Instruments that we planned to use are out of stock for the foreseeable future. Therefore, we had to look at other manufacturers. Table 1 shows a compilation of voltage regulators that we plan to use if they come into stock. This stock was based off the stock available on Digikey. We aimed to have at least 5 different voltage regulators in case the stock suddenly changes, so we could have multiple layers of backup. We wanted to minimize the cost of the unit, while being within the specs of the robot. We knew that the max current draw of the whole unit would be 10 A, but the robot should never be running anywhere close to that maximum. Realistically, it will be more likely that the max current will be at 5A, while the typical current draw should be less than 3A. The regulator AP62150Z6-7 is the cheapest price/unit regulator that we have on the table, but it had a current output of 1.5A. It may be sufficient most of the time, but it is cutting close to the maximum expected current.

Table 14: Possible Voltage Regulators

Part #	Manufacturer	Availability	Price/Unit	Current (A)
TPS62903RPJR	Texas Instruments (TI)	<input type="checkbox"/>	\$2.04000	3
TPS566238RQFR	TI	<input type="checkbox"/>	\$2.330	6
TPS564208DDC R	TI	<input type="checkbox"/>	\$0.788	4
BD86120EFJ-E2	Rohm Semiconductor (RS)	<input checked="" type="checkbox"/>	\$2.19000	5

NR111E	Sanken	☑	\$1.69000	4
LM22673MR	TI	☑	\$6.04000	3
AP62150Z6-7	Diodes Incorporated (DI)	☑	\$0.54000	1.5
LMR14030SSQD DAQ1	TI	☑	\$3.94000	3.5
SC4524FSETRT	Semtech Corporation	☑	\$1.56000	2

From Table 14, we decided that the voltage regulators NR111E or SC4524FSETRT are the best cost performing price/unit for our case. NR111E has much more room for error with a higher maximum current output. The voltage regulator LM22673MR is very expensive compared to the other units, and expensive in general for our budget, because if we order 10 for testing, it will be 60\$, which is a large portion of our budget. The voltage regulator LMR14030SSQDDAQ1 is a possible regulator, but is more on the expensive side. We will keep this one in mind, but prefer not to use this or the previously discussed one. TPS566238RQFR was a good regulator because it had a high current ceiling of 6A. It was in the middle in terms of pricing for all of the prices in our table. However, it was out of stock for the foreseeable future. TPS564208DDCR was cheap, and had good room for error in regards to current, but it was also out of stock. The regulator BD86120EFJ-E2 was a possible candidate, as it had high current output, but begins to encroach on the expensive territory. The top 3 candidates are: NR111E, SC4524FSETRT, BD86120EFJ-E2. This was before deciding to use the LM1084-ADJ regulators in the final design.

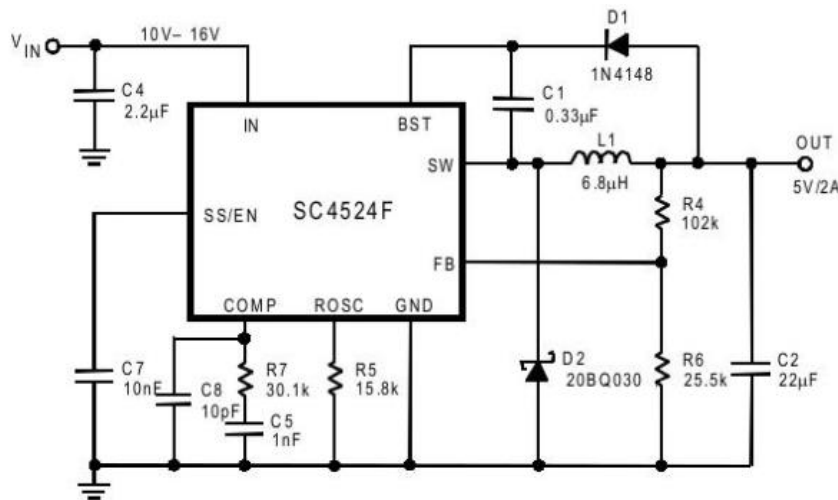


Figure 47: 12V to 5V Buck SC4524F Voltage Regulator Circuit

Using the voltage regulator SC4524F, Figure 47 shows a possible circuit that we could use to drop the 12V input from the battery down to the 5V output to power the different devices such as the Jetson Nano. Furthermore, after dropping the voltage down to 5V, we could add another SC4524F voltage regulator circuit after, to drop the 5V even further down to 3.3V for the other components that require it. This circuit can be found in Figure 48.

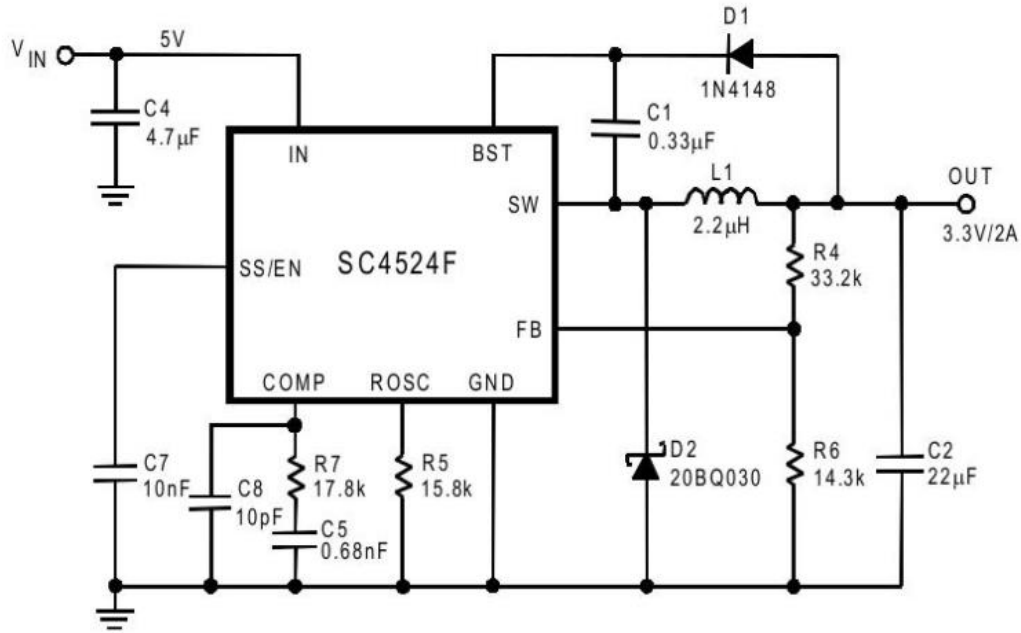
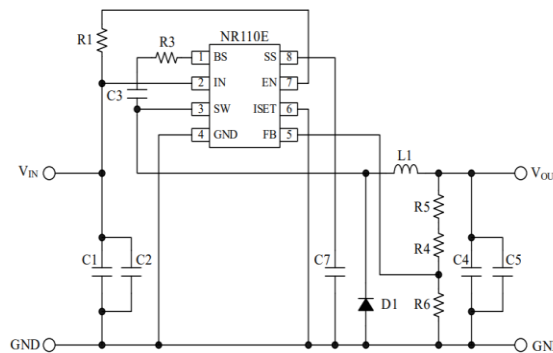


Figure 48: 5V to 3.3V Buck SC4524F Voltage Regulator Circuit

For the NR110E voltage regulator circuit, Figure 49 showcases the configuration for 5V output. According to the table, V_o is controlled by changing the R5 resistor.



Symbol	Ratings	Symbol	Ratings
C1	10 µF / 35 V	R1	510 kΩ
C2	10 µF / 35 V	R3	22 Ω
C3	0.1 µF	R4	18 kΩ
C4	22 µF / 16 V	R5	2.7 kΩ ($V_o = 5.0$ V)
C5	22 µF / 16 V	R6	3.9 kΩ
C7	0.1 µF	L1	10 µH
		D1	40 V, 5 A (Schottky diode)

Figure 49: 12V to 5V Buck NR110E Voltage Regulator Circuit

From the figures above, it was likely that we will be using the SC4524F voltage regulator made by the Semtech Corporation. While we may be using the SC4524F regulator, it would be a good idea to still get the top 3 candidates, and run testing on them and compare the results.

Once we had gathered the required components for the Buck Circuit in Figure 49, we could test the circuit using a breadboard, before physically making the circuit on a PCB. During testing, we wanted to test to make sure the output voltage is 5V, and that the maximum current was 2A. This can be done by measuring with a DMM, and using a DC Electronic Load to vary the current. We also wanted to make sure the regulator did not have bad efficiency and thus did not heat up too much.

Table 15: Components Needed for Regulator Circuits

Component	Value	Quantity (for 1 board)
Capacitor	2.2 μ F	1
	0.33 μ F	2
	10nF	2
	10pF	2
	1nF	1
	22 μ F	2
	4.7 μ F	1
	0.68nF	1
Resistor	102k	1
	25.5k	1
	15.8k	2
	30.1k	1
	17.8k	1
	14.3k	1

	33.2k	1
Inductor	6.8 μ H	1
	2.2 μ H	1
Diode	1N4148	2
Zener Diode	20BQ030	2
SC4524F	N/A	2

The above table shows the materials needed to implement the circuits from Figures 48 and 49 onto a PCB. The table shows the minimum number of components to make just one board, so if we wanted to create several boards to test, more components would be required.

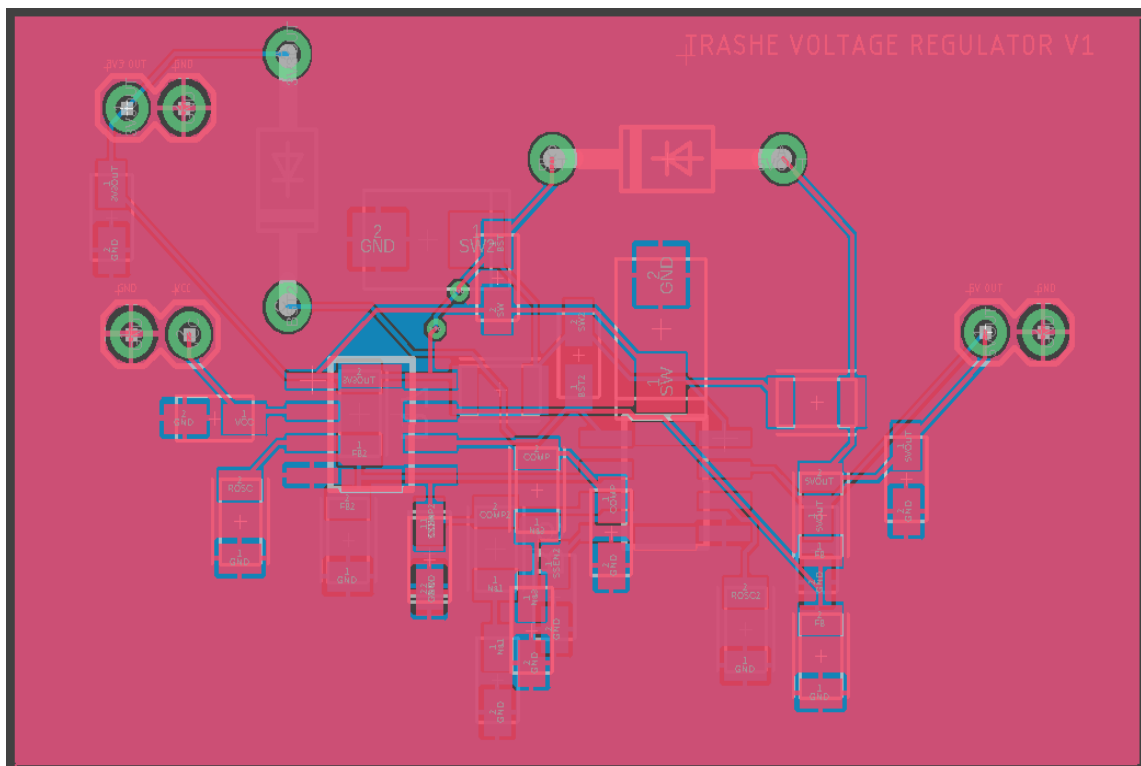


Figure 50: Voltage Regulator PCB

The PCB above was the voltage regulator design for Trash-E. The PCB includes both the buck converters for the 12V-5V step down and the 5V-3.3V step down converters. The 12V input from the battery is VCC+ on the left. The 5V output is on the right and the 3.3V output is top left. This PCB should be able to handle all

the voltage regulation that we need done on the robot. If we need more output ports, we can add more headers later after testing.

In our final design, we used the lm1084-ADJ linear regulators made by TI. This was because we were unable to get switching regulators with high current capacity.

6.2.9 Solar Panels

Table 16: Possible Solar Panels

Part #	Manufacturer	Voltage @ mpp (V)	PV Cell Type	Price/Unit (\$)
SM500K12TF	ANYSOLAR Ltd	6.7	Monocrystalline	\$6.26000
SP3-37	PowerFilm Inc.	3	?	\$2.99000
10Pcs 5V 60mA Epoxy Solar Panel	SUNYIMA	5	Polycrystalline	\$15.99/10
2 Pieces 2.5W 5V/500mAh Solar Panel	ALLPOWERS	5	Polycrystalline	\$12.99/2

Table 16 shows the solar panels that we found that were not too expensive. Since the panels were not 12V, we could have 2 or 3 panels, depending on the Voltage at mpp, in series to create a 12V output. The SM500K12TF panels were monocrystalline and claimed to be highly efficient. We could test this following the Solar Panel Test Plan outlined in the next section. The SP3-37 panels were cheap per unit. We considered these cells because the spec sheet on the listing showed promising specs. However, the datasheet of these cells did not contain much useful information, and we believe that it would be best to stay away from the cells. The solar cells made by SUNYIMA were a possible candidate, because they came in large quantity for a fair price. The only problem was that they could only supply up to 60mA at maximum output. This could be solved by adding more cells in parallel, but that conflicted with how much room we had on Trash-E to add cells. The solar cells made by ALLPOWERS were a good candidate because they could supply up to half an amp at maximum output.

Out of the above options, we believed that it is a good idea to get the SM500K12TF and the panels made by ALLPOWERS. These panels seemed to have the best output for our case, and we could compare the two panels once we physically have them.

We did not use solar panels in our final design, as it was not feasible because Trash-E only ran indoors.

6.2.9.1 Solar Panel Test Plan

The test plan for the solar panels was to test the solar panels when they are receiving the maximum amount of sun they could receive, and when the panels are getting partial coverage.

1. Create a text fixture so the solar panel can get maximum coverage from the sun on a sunny day.
2. Connect solar panels to a DC electronic load, and have a voltage recording device such as a 34970A Data Acquisition / Data Logger Switch Unit.
3. When the sun is at its peak, and when there are no clouds, record the voltage and current from the solar panels over an hour, as that is how long we aim to have Trash-E running.
4. When the sun is not at its peak, or when there are partial clouds or obstructions between the sun and the panels, repeat step 3.
5. Take the readings from the Data Acquisition Unit and plot a current and voltage curve. With this, we can determine which panels are the best fit for us, and if they follow the datasheet that are provided.

6.2.10 PCB Design

Due to the package of our microcontroller (LQFP), we made a printed circuit board (PCB) to use it. We made a breakout-style PCB that has headers connected to the desired pins we wanted to use. An issue we ran into while selecting the pins was that a pin can have multiple functionalities, and the specified use is configured in software. This means that while the microcontroller can have a specified maximum number of capabilities, that's not always the case depending on the amount of a specific feature the design requires. While optimizing the PCB for space and shortest traces, we tried to use pins 13 and 14 which are not only PWM signal generators but are also UART pins as shown below in Table 17 and were the ones we are already using to communicate with the Jetson Nano.

Table 17: Conflicting Pin Functionalities

Pin Number	Pin Name	Description
13	PC7 U3Tx WT1CCP1	GPIO port C bit 7. UART module 3 transmit. 32/64-Bit Wide Timer 1 Capture/Compare/PWM1.
14	PC6 U3Rx WT1CCP0	GPIO port C bit 6. UART module 3 receive. 32/64-Bit Wide Timer 1 Capture/Compare/PW0.

Before designing the PCB, we made a schematic with all the necessary parts and connections using Autodesk Fusion 360 as seen below in Figure 51. The selected components are shown in Table 18. We decided on the 1206 package for the slightly larger size and availability of components. Having the 1206 made it a bit easier for us to solder the components by hand due to the larger pad sizes when compared to other packages like the 0805. Many basic components such as the capacitors and resistors are relatively cheap across different packages so price did not play a large role in this selection process.

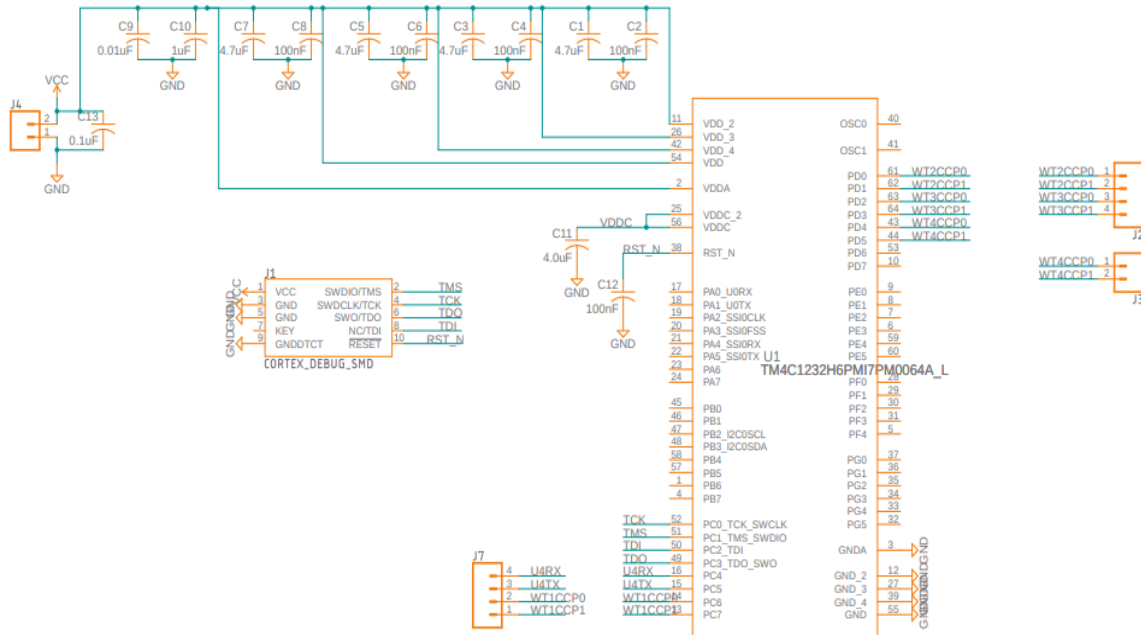


Figure 51: Microcontroller PCB Schematic

Table 18: Microcontroller Schematic Components

Schematic Part	Value	Part Number	Type
C1,C3,C5,C7	4.7uF	UMJ316BC7475KLHTE	Multilayer Ceramic
C2,C4,C6,C8,C12,C13	100nF	C1206C104K5RAC7867	Multilayer Ceramic
C9	0.01uF	C1206C103J3GECTU	Multilayer Ceramic
C10	1uF	C1206F105M5RACAUTO7210	Multilayer Ceramic
C11	4.0uF	C1206C395K3PACTU	Multilayer Ceramic
U1	-	TM4C1232H6PMI7	Microcontroller
J1	-	-	JTAG connector
J2	-	-	Header 4x2-pin Female
J3	-	-	Header 4-pin Female
J4	-	-	Header 2-pin Female
J5	-	-	Header 2-pin Female
J6	-	-	Header 4-pin Female

The datasheet indicated that decoupling capacitors are required to filter out high frequencies as well as stabilizing the supply voltage to the microcontroller when voltage dips occur due to changing load requirements. We opted for a design that minimizes space and maximizes routability. To do this we chose the pins we knew we would need to implement the basic peripherals as well as a few extra so we can further expand upon our design in the future with stretch goals. We have two pins dedicated to UART transmission between the microcontroller and Jetson Nano, as well as eight pins that can be used for GPIO or PWM purposes to allow

for motor control or ultrasonic sensor trigger and feedback signals. Ground pins are available for each PWM pin for the connections to motors.

There are two PCB designs that could have been sent to a manufacturing house. Design 1 as shown in Figure 52 is the initial design that is focused on minimizing the amount of layers as well as cost. Design 2 as shown in Figure 53 is focused on minimizing space and traces to keep the board compact. The major difference between these designs is that Design 2 has two copper ground pours, and the capacitors placed on the bottom. The copper pour increases the cost of the overall board since there are now two layers of copper, but it also reduces the amount of traces needed and the routing is much simpler. As seen in Figure 52, the amount and length of traces is drastically longer, as well as the number of vias drilled into the board. Design 2, however, is much simpler and keeps the traces, especially between the capacitors and microcontroller, nice and concise. It is also worthy to note that Design 2 has the capacitors placed on the back to reduce the amount of vias needed for routing, as well as ground vias placed throughout the board to ensure every component is properly grounded to the copper pours. The microcontroller is placed in the middle of both designs since it has the most connections. This reduces the complexity of the board by keeping traces direct and not having to go roundabout ways to connect to their destination. Finally, four mounting holes are added to the corners to allow for easy attachment to the Trash-E's chassis. With there only being one voltage supplied to the board, this makes it easier to design as there's no worries about noise between nets. We considered adding the voltage regulators to this board as well but decided against it. It would be more beneficial to have the voltage regulators on their own boards so we can test their functionality easier by probing the inputs and outputs of the regulators themselves. This also provides easier placement of the voltage regulators on the chassis since different components need to be powered by the different voltages and they won't be near the microcontroller PCB. We sent Design 2 to be manufactured and is our final PCB design for the MCU board.

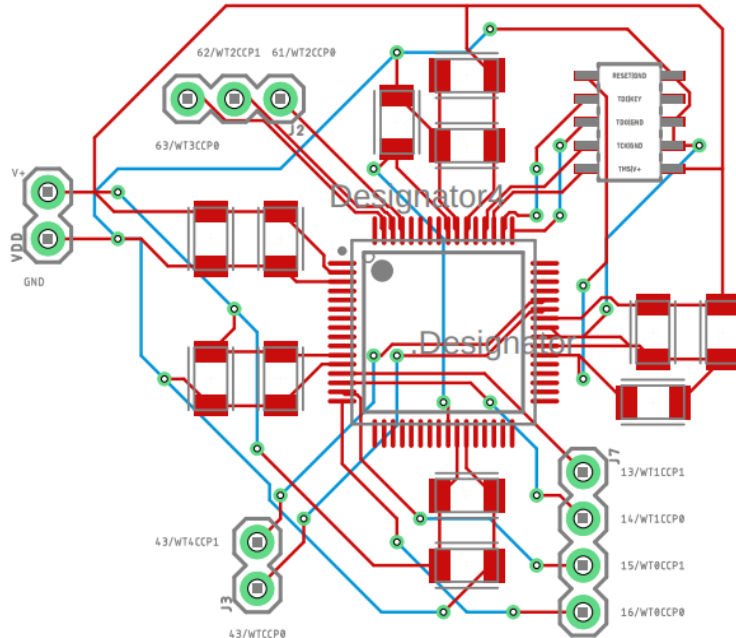


Figure 52: Design 1 PCB Layout

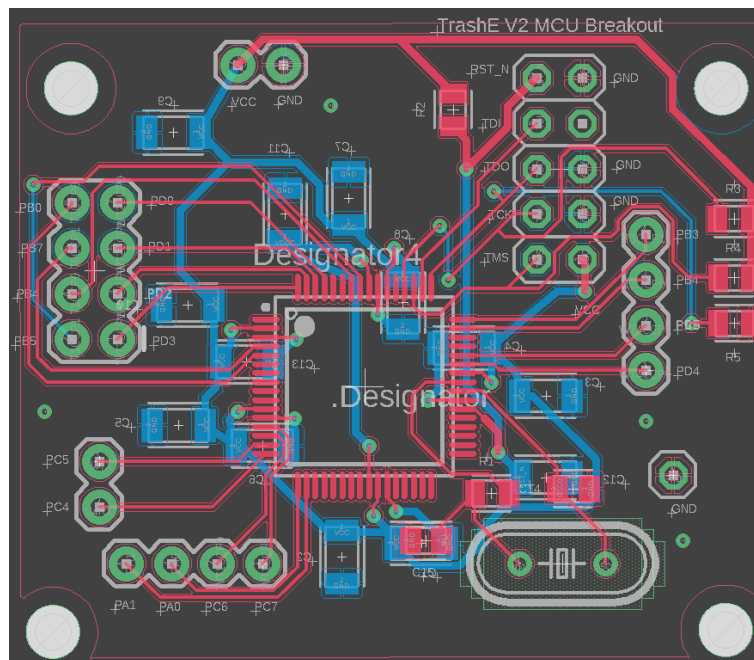


Figure 53: Design 2 PCB Layout

Table 19: PCB Manufacturing Costs

Manufacturing House	Design	Number of Boards	Price
JLPCB	Breakout 2	5	\$9.37

OSH Park	Breakout 2	3	\$11.50
PCB Way	Breakout 2	5	\$14.93

We uploaded the Gerber files for our PCB design to different PCB manufacturers' websites to get quotes on how much it would cost for our PCB to be made. After an online search to find a few reputable manufacturing houses, we decided on these three as candidates: JLCPCB, OSHPark, and PCBWay. JLCPCB and PCBWay are based in China, whereas OSHPark is based in the United States. Being based in the US is nice for us because it can help save on shipping since the boards don't need to be shipped across the sea. For the base material of our board, we chose FR-4. Aluminum has better heat dissipation and thermal transfer than FR-4, but it is also more expensive and won't be necessary for our PCB with the few components we put on the board. Choosing aluminum also restricts other decisions for our board, like the minimum thickness of the board or the number of layers to be one layer maximum. We knew our board needs two layers, so aluminum was not an option based on that requirement. The delivery format is a single PCB where they only manufacture the design how it is and don't add components onto it. We wanted this option since we wanted to solder the components on ourselves. Doing the soldering ourselves decreased the cost and increased our skills with dealing with electrical components and PCBs. The outer copper weight is 1oz since it is sufficient for our ground planes, as well as the 2oz option being drastically more expensive to have manufactured. The default, also the cheapest, options was chosen for color, silkscreen, thickness, gold fingers, probe testing, and castellated holes.

Manufacturing houses have different requirements for the minimum number of boards that can be printed in a single order. Since we were not mass producing these and don't want to spend all our budget on just the boards, being able to order small batches was necessary. We wanted multiple boards to prototype, and to ensure we had extras in case one gets damaged while we were soldering. JLCPCB allows a minimum of five boards to be printed. PCBWay also has the same minimum requirement. OSHPark, however, has a minimum requirement of three boards. Being able to have three boards was enticing to help keep costs down since more materials wouldn't be used to manufacture more boards, specifically the copper.

Speed of the manufacturing and delivery of our PCBs was also crucial for Trash-E. Houses from China would obviously introduce longer shipping times and cost more for the same shipping than a US based house. JLCPCB offers 12-20 business day shipping as their cheapest and slowest option and PCBWay offers 6-16 business day shipping for theirs. OSHPark has free, five business day shipping since the company is in the US. We utilized JLCPCB for our PCB manufacturing needs. The number of boards we received for the price as well as the timely delivery, given the distance they traveled, couldn't be beat.

6.2.11 Manufactured PCBs

Figures 54 and 55 show the first prototype PCBs of the Trash-E Breakout Board. They were ordered from OSHPark and took a turnaround time of two weeks from the initial placement of the order to the boards being shipped to the final address. Overall, the boards are good quality and do not bend. There is complete separation between pads and the contacts are properly grounded. The final PCBs can be seen in Figures 56 and 57. They were printed at JLCPCB.



Figure 54: Top Side of Trash-E MCU Breakout

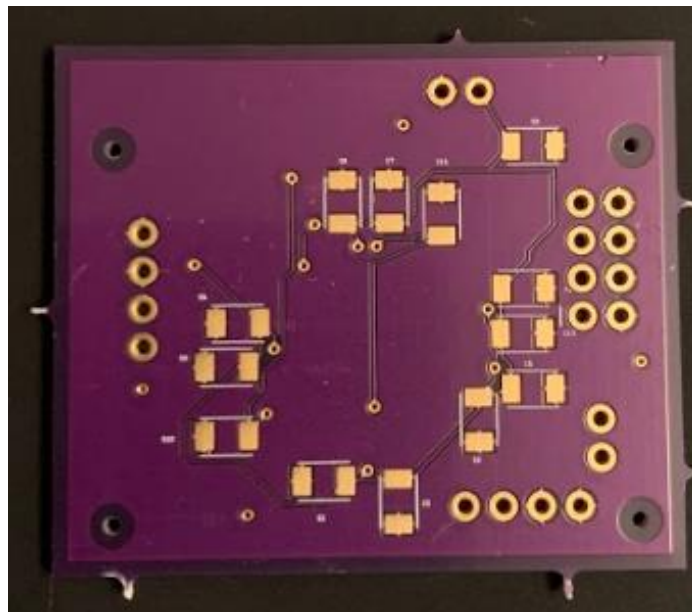


Figure 55: Bottom Side of Trash-E MCU Breakout



Figure 56: Front of Final MCU PCB

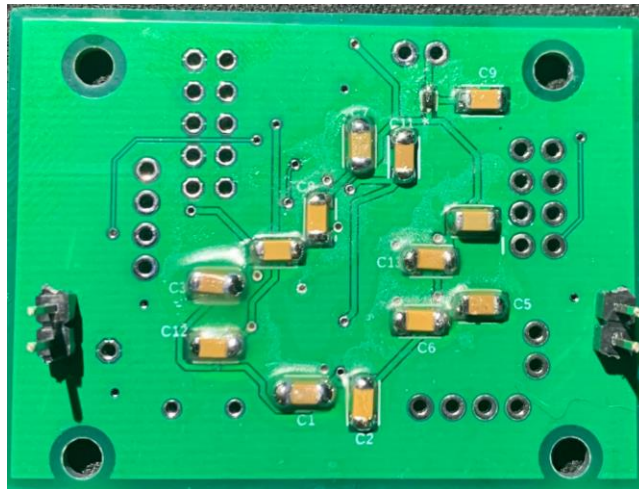


Figure 57: Back of Final MCU PCB

6.3 Bill of Materials (BOM)

Table 20: Bill of Materials

Type	Part Name	Description	QTY	Unit Cost	Total Cost
Wheels	Pololu Wheel for Standard Servo	Wheel for robot to move.	2.00	\$4.75	\$9.50
Motor Driver	A4988 Stepper Motor Driver Carrier	Drive stepper motors.	5.00	\$5.47	\$27.35
Stepper Motor	Twotrees Nema 17	Move the arm (Design 1)	1.00	\$9.99	\$9.99
Servos	154	Move the wheels	2.00	\$11.95	\$23.90
Micro Servo	SER0006	Servo to open and close gripper on arm.	1.00	\$3.62	\$3.62
Ultrasonic Sensor	SainSmart HC-SR04	Sensor to detect cup distance from gripper.	1.00	\$4.45	\$4.45
MCU	TM4C1232H6PMI7	Microcontroller for peripherals	1.00	\$7.14	\$7.14
Jetson Nano	4GB Mini-Computer	For Computer Vision	1.00	\$99.00	\$99.00
Camera	SainSmart IMX219	For CV	1.00	\$21.99	\$21.99
Lidar Rangefinder	MakerFocus YDLIDAR X2L	For SLAM	1.00	\$69.99	69.99
Chassis/ Bucket	3D Printed	Robot Chassis	1.00	3D Printed	3D Printed
Arm	3D Printed	Arm to pick up cups	1.00	3D Printed	3D Printed
Gripper	3D Printed	To grip cups	1.00	3D Printed	3D Printed

7.0 Prototyping, Build, Test, Evaluation Plans

7.1 Prototyping

7.1.1 Block Diagram Explanation

The hardware composition of Trash-E is visualized on the block diagram below in Figure 58. The block diagram contains the overall grasp of the project. This gives a development path as well as a visual representation of how each part connects to each other.

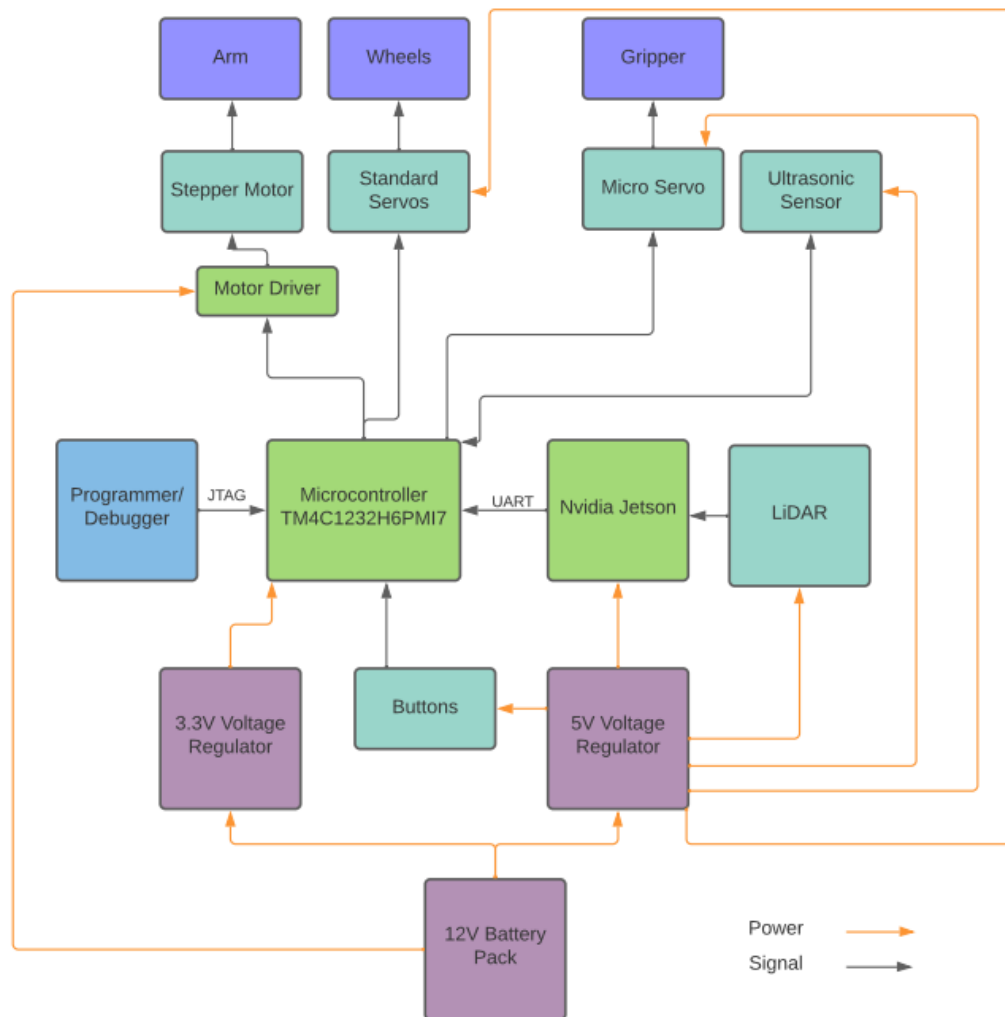


Figure 58: Trash-E Hardware Block Diagram

7.1.1.1 Electrical Components

Starting from the bottom of Figure 58, the power supply designed with constraints and standards in mind was grounded and had no exposed circuitry. It was also able to satisfy all power needs of one stepper, two continuous servos, one regular servo, a microcontroller, an Nvidia Jetson 4GB, as well as all the corresponding peripheral components. To ensure everything receives the specified voltage and wattage to perform optimally a voltage regulator for two separate blocks of the project as they will have separate voltage requirements.

7.2 Computer Vision Testing

One of the most important aspects of Trash-E was the object detection component. Our robot would not do anything at all if our object detection software was not working or working optimally. For our computer vision object detection software we wanted to test a few aspects before even proceeding to the microcontroller software. Our object detection software should be able to accurately detect the trash objects that we train it to detect and should ignore any other object. It should be able to detect objects regardless of orientation and the surrounding environmental factors such as lighting and background noise. The software should be able to select the closest object of interest by the size of its bounding box. The software should be able to correctly track where the closest object is on the image and use the coordinates from its bounding box to calculate the correct result to send over to the microcontroller. The performance of the object detection should be fast enough that we can accurately detect objects while moving.

7.2.1 Testing Object Detection

The first step to successful object detection was being able to detect the object. For each class item of trash that we train our model to detect, we wanted to test for accurate detection meaning no false positives or negatives as well as accurate detection in different orientations, lighting, and backgrounds. The object should be detected from orientations horizontally and vertically. For example, we detected a cup that was standing upright, sideways, flipped, or even rotated. We expected that we cannot detect objects in darkness but should function normally in natural and artificial light. Different color backgrounds and noisy backgrounds should not affect detection. Successful detection of these objects meant that we could move forward to sorting these detections and moving on in our object detection process.

Procedure:

1. Run the object detection model while using the camera feed as the image input.
2. Check that the video feed is live in the window.
3. Check that the frame rate performance of the model is more than 10 frames per second.
4. Check that no objects are being currently detected.

5. Place an unknown object in camera view and check that it is not detected.
6. For each trash object we have trained to detect, repeat steps 3 to 5.
7. In artificial light, place the object in view of the camera and do steps 7 to 10.
8. In natural light, place the object in view of the camera and do steps 7 to 10.
9. Check that the object is correctly identified and has been labeled correctly.
10. Check that the object is correctly identified in multiple orientations (rotations by 45 degrees, flipped, sideways).
11. Check that the object is correctly detected with a solid background as well as a noisy background.
12. Ensure there are no ghost objects or false positives being detected.

7.2.2 Testing Object Selection

Upon successful detection of trash, our algorithm sorted the detections and their respective bounding box coordinates. These detections were sorted by decreasing area. This aspect is essential to Trash-E being able to decide which object it should head towards out of all objects in the camera view. During this testing we used a plastic cup as our object.

Procedure:

1. Run the model and use the camera as the input.
2. Check that the program window is showing a live camera feed.
3. Check that the object detection is working by placing a plastic cup in the view.
4. Grab at least four plastic cups and place them in view at different distances away from the camera view.
5. Check that the program is only tracking the closest object.
6. Repeat steps 4 - 6 as many times as needed.

7.2.3 Testing Object Tracking

Once the object detection model successfully detected and selected the object that is closest, the algorithm correctly calculated the position of this object relative to the image center and determined the direction that the robot turned in order to make it face straight at the object. In this testing, we will only be checking for correct results prior to serial communication over UART. This test only focused on the computer vision aspect of the software. During this testing we used a plastic cup as our object.

Procedure:

1. Run the model and use the camera as the input.
2. Check that the program window is showing a live camera feed.
3. Check that the object detection is working by placing a plastic cup in the view.

4. With no objects in view of the camera, check that the value to be sent via UART is 0.
5. Use valid objects for the following steps.
6. Place one cup on the image center.
7. Check that the program output says to move forward and the value to be sent via UART is 1.
8. Place one cup to the left of the image center.
9. Check that the program output says to move left and the value to be sent via UART is 2.
10. Place one cup to the right of the image center.
11. Check that the program output says to move right and the value to be sent via UART is 3.

7.2.4 SLAM Test

The objective of this test was to ensure the SLAM algorithm can generate an accurate map of an unknown environment.

Procedure:

1. Connect the lidar sensor to the Jetson Nano GPIO pins.
2. Connect the 5V power supply to the lidar sensor.
3. Place various obstacles around the room. Ensure there are obstacles on every side of the lidar sensor to test the 360° rotation of the sensor.
4. Draw a map of the current environment, labeling walls and drawing the shapes of obstacles the algorithm will detect.
5. Transfer the SLAM algorithm onto the Jetson Nano, provide a power source and turn it on.
6. Wait for the algorithm to generate a map of the area.
7. Compare the generated map to the map that was drawn by the tester and ensure they are identical.

7.3 Hardware Testing Plans

Before we built Trash-E, we needed to test the individual aspects our design would achieve to ensure it would work overall. Testing was done at the Senior Design lab in Engineering 1 on UCF campus or at a group member's residence. Location was determined by each specific test and the equipment required by said test. Each test had procedural steps so that the test could be replicated by anyone given they have the required equipment.

Equipment:

The following equipment is required to complete all tests below:

- Multimeter
- DC Power Supply

- Oscilloscope or Discovery Kit
- 12V battery configuration
- Voltage Regulator PCB
- TM4C1232H6PMI7 Design 2 PCB
- Power Rail Breakout Board

7.3.1 Voltage Regulator Testing

The objective of this test was to ensure the batteries can power the components at the correct voltages using the voltage regulator PCB we had manufactured. This also checked to make sure the soldering is done correctly.

Procedure:

1. Check all solder connections through a microscope to ensure all components are secured and connected to the board, and that there is no solder bridging between pins.
2. Connect a jumper cable to the 5V output header "5V OUT" pin.
3. Connect a jumper cable to the 5V output header "GND" pin.
4. Attach the positive multimeter probe to the "5V OUT" jumper cable.
5. Attach the negative multimeter probe to the "GND" jumper cable.
6. Connect a jumper cable to the input header "VCC" pin.
7. Connect a jumper cable to the input header "GND" pin.
8. Attach the positive DC power supply alligator clip to the "VCC" jumper cable.
9. Attach the negative DC power supply alligator clip to the "GND" jumper cable.
10. Sample the voltages for ten seconds at each input voltage: 8V, 9V, 10V, 11V, 12V.
11. Ensure the multimeter is reading 5V +/- 0.2V at 10V and higher.
12. Turn off the power supply.
13. Connect a jumper cable to the 3V3 output header "3V3 OUT" pin.
14. Connect a jumper cable to the 3V3 output header "GND" pin.
15. Attach the positive multimeter probe to the "3V3 OUT" jumper cable.
16. Attach the negative multimeter probe to the "GND" jumper cable.
17. Sample the voltages for ten seconds at each input voltage: 8V, 9V, 10V, 11V, 12V.
18. Ensure the multimeter is reading 3.3V +/- 0.2V at 10V and higher.
19. Repeat steps 2-18 except change the DC power supply for the battery configuration that will be used with the robot.

7.3.2 Powering of the Microcontroller

The objective of this test was to verify that the microcontroller was receiving power and all soldering had been done correctly.

Procedure:

1. Check all solder connections through a microscope to ensure all components are secured and connected to the board, and that there is no solder bridging between pins.
2. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
3. Flash the program "Hello.c" onto the microcontroller.
4. Connect a jumper cable to the input header "VCC" pin.
5. Connect a jumper cable to the input header "GND" pin.
6. Connect a jumper cable to the "WT4CCP1" pin.
7. Connect a jumper cable to a nearby "GND" pin.
8. Connect the probe of an oscilloscope to the "WT4CCP1" pin.
9. Connect the corresponding ground probe to the nearby "GND" pin.
10. Connect a DC power supply to the input header "VCC" and "GND" pin.
11. Turn on the DC power supply and set the voltage to 3.3V.
12. Verify the oscilloscope shows a square signal with a 50% duty cycle.

7.3.3 Powering a Servo Motor

The objective of this test was to ensure we can generate a PWM and power any servo motor that will be hooked up.

Procedure:

1. Check all solder connections through a microscope to ensure all components are secured and connected to the board, and that there is no solder bridging between pins.
2. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
3. Flash the program "Hello.c" onto the microcontroller.
4. Connect jumper cables to both the 5V and 3.3V and their respective ground pins on the voltage regulator board.
5. Connect the 3.3V output of the voltage regulator to the input header "VCC" pin on the microcontroller PCB.
6. Connect the 3.3V ground cable of the voltage regulator to the input header "GND" pin on the microcontroller PCB.
7. Connect a jumper cable to the "WT4CCP1" pin.
8. Connect a jumper cable to a nearby "GND" pin.
9. Connect the 5V output of the voltage regulator to the positive power rail on the breakout board.
10. Connect the 5V ground pin of the voltage regulator to the negative power rail on the breakout board.
11. Connect the power connection of the servo motor to the 5V power rail.
12. Connect the signal connection of the servo motor to the "WT4CCP1" pin.
13. Connect the ground connection of the servo motor to the ground power rail.
14. Connect the DC power supply to the "VCC" and "GND" pins of the voltage regulator.

15. Turn on the DC power supply to 12V.
16. Observe that the servo motor is alternating between spinning one way 180° and the other way 180°.

7.3.4 Powering a Stepper Motor

The objective of this test was to ensure we can generate a PWM for the stepper motor and control it with the motor driver and microcontroller.

Procedure:

1. Check all solder connections through a microscope to ensure all components are secured and connected to the board, and that there is no solder bridging between pins.
2. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
3. Flash the program "HelloStepper.c" onto the microcontroller.
4. Connect jumper cables to both the 5V and 3.3V output and their respective ground pins on the voltage regulator board.
5. Connect the 3.3V output of the voltage regulator to the 3.3V power rail on the power rail breakout board then to the input header "VCC" pin on the microcontroller PCB.
6. Connect the 3.3V ground cable of the voltage regulator to the ground power rail on the power rail breakout board then to the input header "GND" pin on the microcontroller PCB.
7. Connect a jumper cable to the "WT4CCP1" pin.
8. Connect a jumper cable to a nearby "GND" pin.
9. Connect a jumper cable to the "WT4CCP0" pin.
10. Connect a jumper cable to a nearby "GND" pin.
11. Connect the positive terminal of the DC power supply to one of the positive terminals of the power rail breakout board and the negative terminal to the ground rail.
12. From the positive terminal on the power rail breakout board, make one connection with a jumper cable to the "VCC" input on the voltage regulator and another to the "VMOT" pin on the motor driver as well as their corresponding ground connections to the power rail breakout board.
13. Connect the "WT4CCP0" and "WT4CCP1" pins to "STEP" and "DIR" on the motor driver, respectively.
14. Connect the 3.3V and ground pin from the power rail breakout board to "VDD" and the ground power rail to "GND" on the breakout board, respectively.
15. Turn on the power supply and set the voltage to 12V.
16. Observe that the stepper motor is alternating between spinning one way 180° and the other way 180°.

7.3.5 Ultrasonic Sensor Testing

The objective of this test was to ensure we can get a reading from the ultrasonic sensor with different distances.

Procedure:

1. Check all solder connections through a microscope to ensure all components are secured and connected to the board, and that there is no solder bridging between pins.
2. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
3. Flash the program "HelloUltrasonic.c" onto the microcontroller.
4. Connect jumper cables to both the 5V and 3.3V output and their respective ground pins on the voltage regulator board.
5. Connect the 3.3V output of the voltage regulator to the 3.3V power rail on the power rail breakout board then to the input header "VCC" pin on the microcontroller PCB.
6. Connect the 3.3V ground cable of the voltage regulator to the ground power rail on the power rail breakout board then to the input header "GND" pin on the microcontroller PCB.
7. Connect the 5V output of the voltage regulator to the 5V power rail on the power rail breakout board then to the input header "VCC" pin on the ultrasonic sensor.
8. Connect the 5V ground cable of the voltage regulator to the ground power rail on the power rail breakout board then to the input header "GND" pin on the ultrasonic sensor.
9. Connect the "WT4CCP0" and "WT4CCP1" pins to "Trigger" and "Echo" on the motor driver, respectively.
10. Add an LED in series with the "Echo" pin.
11. Connect the DC power supply to the input header "VCC" and "GND" pin on the voltage regulator.
12. Turn on the power supply and set the voltage to 12V.
13. Move things closer and further away from the sensor and view the LED staying illuminated for longer intervals when the object is further away.

7.4 Evaluation

After component testing, building, and software had been completed, the following tests were run to evaluate Trash-E and confirm it worked as intended.

7.4.1 Robot Movement Testing

The objective of this test was to ensure the robot can accomplish all types of required movement: forward, backward, turn left, turn right, and spin. Before starting, ensure the electrical components are hooked up correctly according to Figure 59.

Procedure:

1. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
2. Flash the program “TestMovement.c” onto the microcontroller.
3. Connect the positive jumper cable of the battery to the “VCC” pin on the voltage regulator, and their corresponding ground pins.
4. Observe the robot perform the movements in this order:
 1. Move forward.
 2. Move backward.
 3. Turn left.
 4. Recenter.
 5. Turn right.
 6. Recenter.
 7. Spin 360°.

7.4.2 Collecting Trash Testing

The objective of this test was to ensure the robot can pick up trash in multiple different orientations. Five different trash orientations were tested. Before starting, ensure the electrical components are hooked up correctly according to Figure 59.

Procedure:

1. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
2. Flash the program “TestGrabbing.c” onto the microcontroller.
3. Connect the positive jumper cable of the battery to the “VCC” pin on the voltage regulator, and their corresponding ground pins.
4. Place a cup one inch away from the base of the gripper with the mouth of the cup facing up.
5. Observe the robot pick up the cup, raise it to the bucket on its back, drop it in, and move the arm back to starting position.
6. Repeat steps 4 and 5 with the following cup configurations:
 1. Mouth of the cup facing down.
 2. Cup on its side, mouth facing left.
 3. Cup on its side, mouth facing the robot.
 4. Cup on its side, mouth facing 45° (in between facing left and facing the robot).

7.4.3 Idle State Testing

The objective of this test was to ensure the robot would follow the path-planning algorithm until a cup was detected. Before starting, ensure the electrical components are hooked up correctly according to Figure 59.

Procedure:

1. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
2. Flash the program “TestIdle.c” onto the microcontroller.
3. Connect the positive jumper cable of the battery to the “VCC” pin on the voltage regulator, and their corresponding ground pins.
4. Put a cup in an arbitrary place on the ground in any configuration.
5. Allow the robot to move around the environment freely using SLAM.
6. Observe the robot detect the cup once it is in frame and collect it into the bin.

7.4.4 Multiple Cup Testing

The objective of this test was to ensure the robot can handle scenarios where there was more than one cup present. Before starting, ensure the electrical components are hooked up correctly according to Figure 59.

Procedure:

1. Connect the DAOKI ST-Link V2 to the JTAG connector on the board using the datasheet and Figure 58 in this document.
2. Flash the program “main.c” onto the microcontroller.
3. Place two cups in the frame of the camera that are both equally close to the robot.
4. Connect the positive jumper cable of the battery to the “VCC” pin on the voltage regulator, and their corresponding ground pins.
5. Observe the robot pick up one of the cups, then the other.
6. Repeat steps 3 through 5 with one cup closer than the other.
7. Repeat steps 3 through 5 with both cups out of the frame and equally close to the robot.
8. Repeat steps 3 through 5 with both cups out of the frame but with one cup closer than the other.

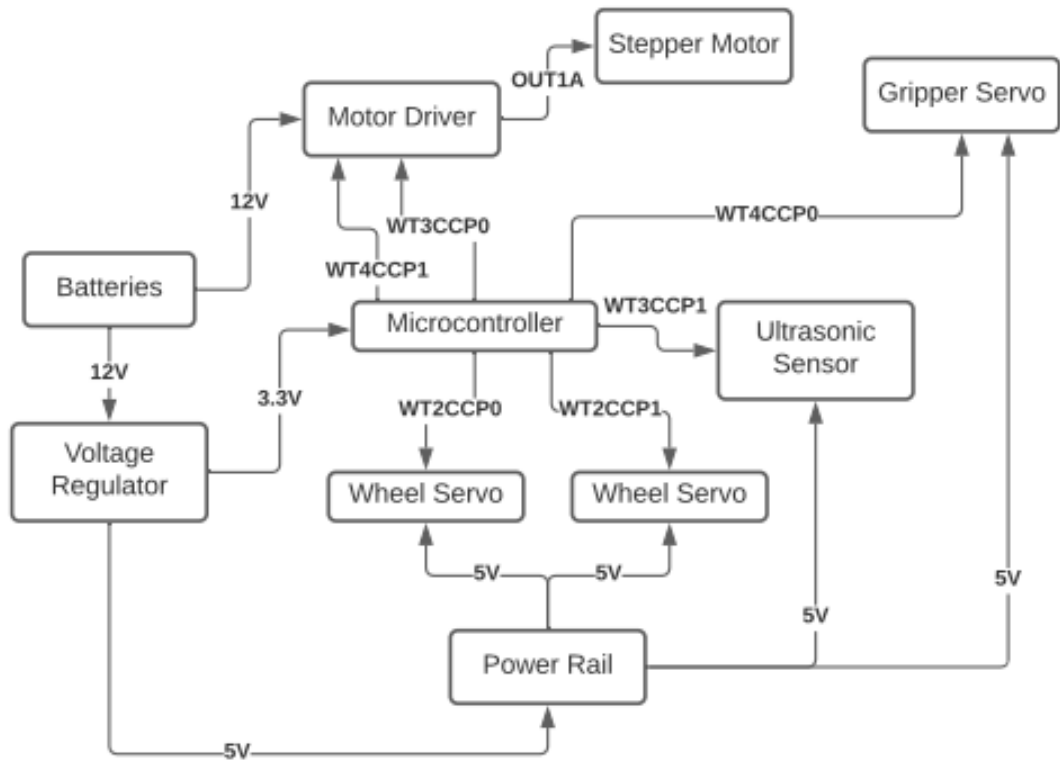


Figure 59: Electrical Connection Layout

7.5 Hardware Component Testing

7.5.1 Voltage Regulator Prototyping

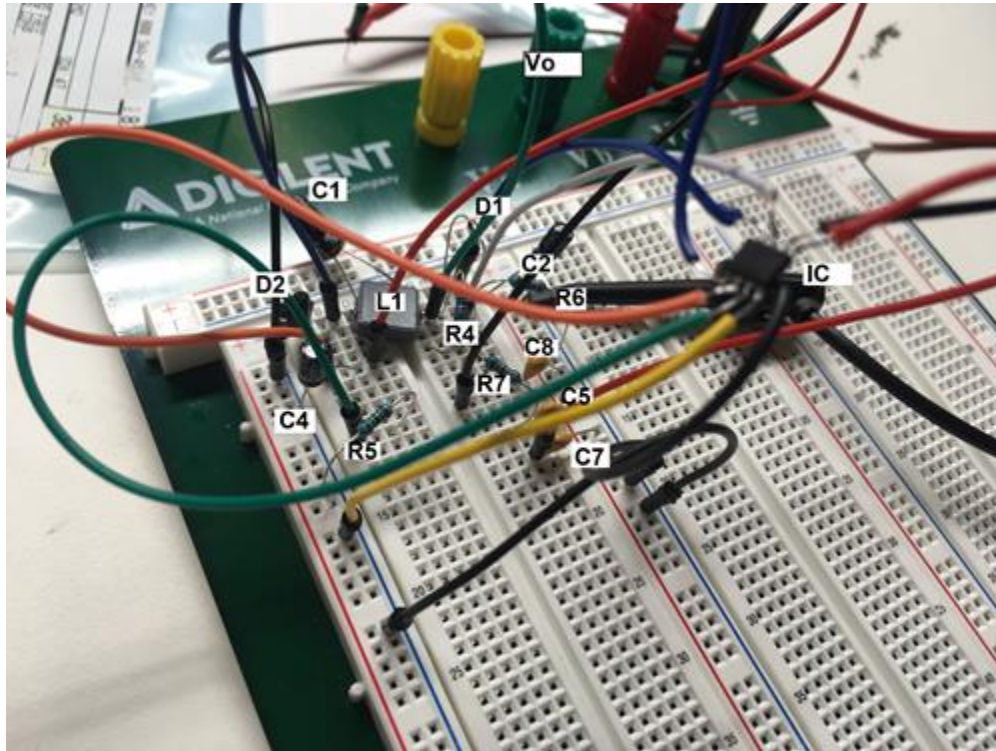


Figure 60: 12-5V Voltage Regulator Breadboard



Figure 61: 12-5V Voltage Regulator Breadboard

Figure 60 showcases the 12-5V voltage regulator that we had designed for the robot. Figure 61 shows the output of the voltage regulator at 4.699V. This is because while we were waiting for the specific resistor and capacitor values to come in, we used common resistor and capacitor values that are close to the values we want. Once we had the parts, we got 5V output. The circuit above uses the SC4524F Integrated Circuit Voltage Regulator that we chose.

Table 21 shows the components that are to be used once the actual components had come in. In the meantime, the part numbers for the capacitors and resistors are omitted, as the current values are different.

Table 21: Components used in breadboard testing

Component Reference	Component Type	Component Value	Part Number
C1	Capacitor	0.33uF	-
C2		22uF	-
C4		2.2uF	-
C5		c5uF	-
C7		10nF	-
C8		10pF	-
R4	Resistor	102kΩ	56-MRS25000C1023FCT00CT-ND
R5		15.8kΩ	PPC15.8KYCT-ND
R6		25.5kΩ	RNF14FTD25K5CT-ND
R7		30.1kΩ	PPC30.1KZCT-ND
L1	Inductor	6.8uH	
D1	Diode	-	1N4148
D2		-	20BQ030
IC1	Integrated Circuit Voltage Regulator	-	SC4524FSETRT

The circuit was able to hold a constant voltage of 5V when the input varies, from 10V to 14V, in case the battery pack had a fluctuation that may cause it to change. Voltages were varied from the input with the power supply from 10V to 20V. The results are in Table 22. From the table, we can see that the voltage difference between the input voltage at 20 V and 10 V are 0.2 V. Thus, the voltage regulator works for our use case. In Table 23 the output voltage differs by 0.3 V.

Table 22: Input and Output Voltage of 12 - 5 V Step Down Regulator

Input Voltage	Output Voltage
20 V	5.5 V
18 V	5.4 V
16 V	5.4 V
14 V	5.3 V
12 V	5.3 V
10 V	5.3 V

We are concerned that the current of the regulator may not be high enough for the autonomous robot, so we considered using regulators with higher current capabilities. Further testing was required once we gathered all the components to the robot such as all the motors, sensors, and microcontrollers.

Although these tests went well, in the actual robot we were unable to gather the necessary parts in order to make these regulators function properly. Therefore, we switched to the lm1084-ADJ linear regulators made by TI.

Table 23: Input and Output Voltage of 5 - 3.3 V Step Down Regulator

Input Voltage	Output Voltage
20 V	3.6 V
18 V	3.6 V
16 V	3.5 V
14 V	3.5 V
12 V	3.5 V
10 V	3.5 V
8 V	3.4 V
6 V	3.3 V
5 V	3.3 V

7.5.2 Ultrasonic Sensor Testing

This test utilized the TI Evaluation Kit EK-TM4C123GXL since we were unable to complete the building of the PCB due to shipping delays from the PCB manufacturer. The microcontroller on the Evaluation Kit was the same microcontroller we used on the PCB so it was an accurate representation of the environment the ultrasonic sensor will be powered by.

To determine the distance of the object from the sensor, the following equation is used

$$Distance = \frac{\left((High\ Level\ Duration) * 340 \frac{m}{s} \right)}{2}$$

Based on this equation, the shorter the high pulse was from the ultrasonic sensor, the closer the object was to the sensor. Table X below compares the different values of the ultrasonic sensor output pulse width between the set distances we chose to test at. Since we want the ultrasonic sensor to be used to stop the robot when the trash was close to the arm and gripper, we chose a maximum of two feet (60 centimeters) and a minimum distance of two inches (5 centimeters). This will give us a realistic use case for the ultrasonic sensor when it was attached to the arm.

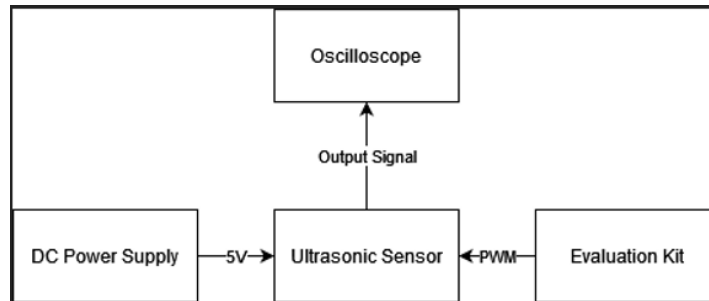


Figure 62: Test Setup for Ultrasonic Sensor

Figure 62 depicts the setup for this test. Code was written on the evaluation kit to continuously generate a PWM at 12kHz with a 50% duty cycle. This signal was sent to the trigger pin of the ultrasonic sensor which activates the ultrasonic waves to be emitted for distance detection. The ultrasonic sensor was also being powered by a DC power supply set to 5V. The output was viewed using an oscilloscope to check the length of the returning square wave.

Table 24: Ultrasonic Sensor Testing Results

Distance From Sensor (cm)	Input Duty Cycle	Frequency (kHz)	Positive Sensor Output Duration (ms)	Measured Distance (cm)
60	50%	12	3.08	52.36
30	50%	12	1.6	27.2
15	50%	12	0.880	14.96
5	50%	12	0.360	6.12

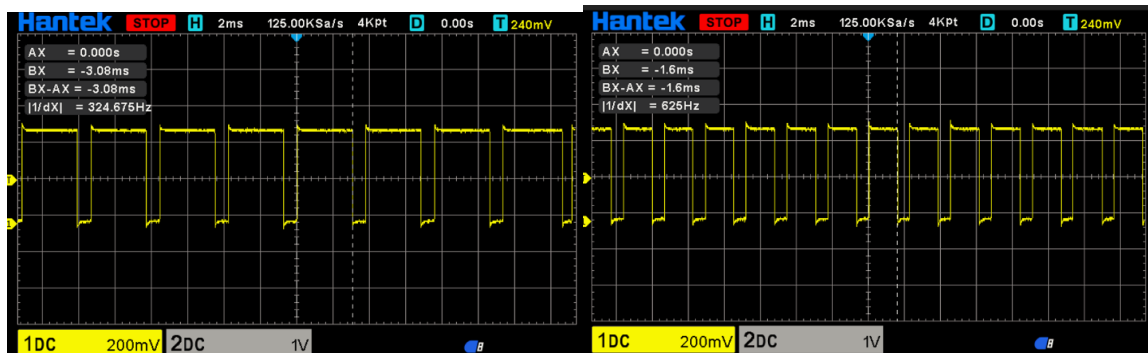


Figure 63: 60cm Output (Left) and 30cm Output (Right)

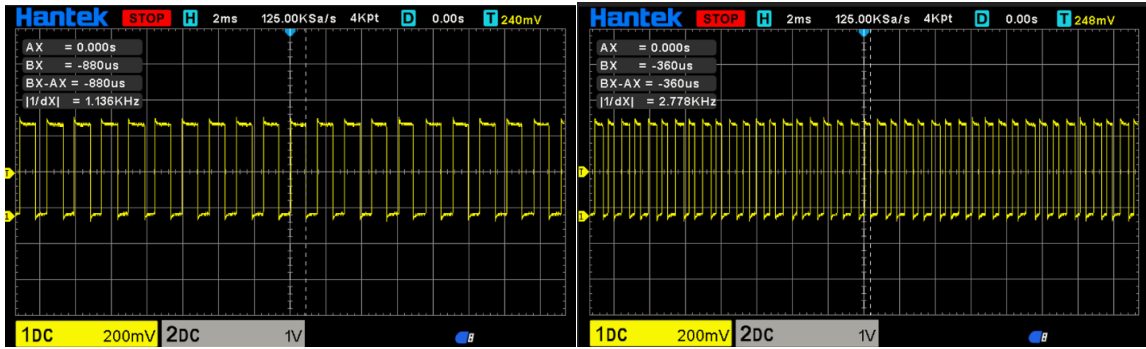


Figure 64: 15cm Output (Left) and 5cm Output (Right)

While an object was further away, it was not as accurate as can be seen with the distance of 60 centimeters. The measurements get more accurate as the object drew closer which was good for our use case. When we read that the ultrasonic sensor on the base of the arm was within two inches, we want to stop the robot to allow for picking up the trash. The inaccuracy of the further distances was negligible for our use case since the closer distance accuracies are good.

8.0 Project Operation

In this section, we will be explaining the operation of Trash-E: Autonomous Litter Picker Upper. Trash-E is a mobile robot that picks up red solo cups. The robot can be placed anywhere in any room that is smaller than 12 meters wide by 12 meters long. Ensure the batteries are charged before use for the longest runtime. Trash-E can be placed anywhere in the room to start.

8.1 Installation, Configuration, and Operation

8.1.1 Setup Jetson Nano for VNC

Plug in the Jetson Nano to the battery pack, and connect a keyboard and mouse, as well as a monitor via HDMI. Once the Jetson Nano is powered on and logs in automatically, open the LXTerminal from the desktop icon. Type in the command 'ifconfig' and press enter. Locate the line that starts with 'wlan0'. Write down the 'inet' address under the 'wlan0' tab. This will be used to remote into the Jetson Nano for startup. Remove the keyboard, mouse, and HDMI cord from the Jetson Nano and leave it powered on.

8.1.2 Connect to Jetson Nano via VNC

On a separate computer that has VNC Viewer or another VNC connecting software installed, enter the 'inet' address obtained from the process above. Click 'Continue' when prompted about an unencrypted connection. Enter the password '1234' when prompted to authenticate. From here you should be able to see the desktop of the Jetson Nano from the computer running VNC viewing software.

8.1.3 Running the Script

Connect the jumper cables on the PCB to allow power to the other components. Ensure the LiDAR is connected and spinning. The microcontroller already is waiting for the instructions from the Jetson Nano so there's no extra configuration needed to get it ready for operation. From the computer that has the VNC connection to the Jetson Nano, open a terminal on the Jetson Nano. Enter the command 'sudo service nvargus-daemon restart' to ensure there are no other processes using the camera feed. From here, type in the command 'roscd trashe_detection'. Then type 'cd scripts'. Finally type 'python trashe-detection.py'. Once enter is hit on the keyboard, the script will start running.

8.1.4 Trash-E Normal Operation

After about 10-20 seconds from running the script, Trash-E will start moving around to generate the map. During this time, it will be using basic obstacle avoidance (the buttons and ultrasonic sensor) to move about the room and generate the map. After 30 seconds of movement, Trash-E will stop moving to save the map and switch modes. This process takes about 20 seconds. Once it starts moving again, Trash-E will use the map previously generated to pick up cups and

maneuver about the room. This is the final step that needs to be taken and Trash-E will continue to run and pick up cups.

To turn Trash-E off, press 'Ctrl+C' on the VNC connection in the terminal. Wait until the ROS log states that all the nodes have been shut off and there are only numbers being printed in the terminal. Close the terminal and shut off the Jetson Nano by clicking in the bottom right of the screen and choosing 'Shutdown' from the pop-up menu. Detach the jumper cables from the PCB.

9.0 Administrative Content

9.1 Milestones

For the fall semester we had established milestones for the project. The first semester was mainly be focused on research, documentation, and design as shown in table 20.

Fall 2021

Table 25: Fall Milestones

Week	Date	Milestone
1	8/23 - 8/29	<ul style="list-style-type: none"> • Group Formation
2	8/30 - 9/5	<ul style="list-style-type: none"> • Brainstorm Projects • Choose Project • Senior Design Bootcamp
3	9/6 - 9/12	<ul style="list-style-type: none"> • Determine Project Requirements • Discuss Project Budget • Divide Project Block Diagram
4	9/13 - 9/19	<ul style="list-style-type: none"> • Divide and Conquer 1.0
5	9/20 - 9/26	<ul style="list-style-type: none"> • Decide Primary and Secondary Features • Revise Project
6	9/27 - 10/3	<ul style="list-style-type: none"> • Divide and Conquer 2.0
7-12	10/4 - 11/14	<ul style="list-style-type: none"> • 60 Page Draft • Research and order Microcontrollers and other parts • Research Software • Begin PCB Design • Decide Pre-built or Custom Chassis
13	11/15 - 11/21	<ul style="list-style-type: none"> • 100 Page Draft
14-16	11/22 - 12/12	<ul style="list-style-type: none"> • Final Submission

Spring 2022

The second semester focused on implementing and building our robot. Table 21 below shows a general outline for milestones of the semester.

Table 26: Spring Milestones

Week	Milestone
2/25	CDR
3/23	Midterm Demo
4/21	Final Presentation
4/26	Final Submission

9.2 Budget and Finance

For this project, we decided that a \$450 budget is possible for the four of us to handle. Since there was no sponsor for this project, the funding was coming from all of us split at \$112 per person. For robotic movement parts and sensors, we planned to have a maximum cost of \$100. The mini-computer was given a maximum cost of \$80. Utilization of a group member's 3D printer, or the 3D printer that is on UCF's main campus, was used to keep costs low for making the chassis. A budget of \$70 was dedicated to 3D printing. For electronic components, like the microcontroller and PCBs, a budget of \$100 was allocated. Due to this project being prone to human error when assembling, we need to have overhead in case unforeseen issues arise and need to be fixed, or new parts need to be acquired. There was \$50 of overhead to cover these costs. The final \$50 was allocated to implementing stretch goals and the possible new parts we will need to acquire.

10.0 Project Summary and Conclusion

During our research and prepared development of the trash picking up robot Trash-E, our group had learned about many topics that could be applied when making an autonomous robot. By researching topics from different projects such as autonomous drones, self-driving cars and other similar emerging technologies, we believed that we would be able to create Trash-E.

From our research, we were able to create an overall conceptual design that we believed will be able to properly do the objectives we wanted to do. With the robot being on four wheels, we were able to maneuver terrain easily. The gripper allowed to pick up more than just small particles such as dust that previous robots which have been made before such as the Roomba, a robot vacuum. We hoped to build upon these previous iterations by broadening the capabilities of a robot by picking up trash instead of just being a vacuum. Some self-driving cars and Roombas use LiDAR, which we believed was a fantastic way to help our robot navigate through its surroundings. The reason we chose our specific components was to keep costs down while also being efficient.

We chose to do this project because we were passionate about cleaning up the environment. With the popularity of social media trends such as cleaning up the beach, we hoped to help motivate people to continuously keep the environment clean. We are also all interested in robotics, so we felt that this would be right up our alley. We felt that this robot would provide an adequate challenge to our abilities and allow us to learn a lot.

11.0 Appendices

This section is used to show the material that was referenced in the document.

10.1 Bibliography

- [1] https://en.wikipedia.org/wiki/ARM_Cortex-M
- [2] <https://www.eembc.org/coremark/>
- [3] <https://en.wikichip.org/wiki/coremark-mhz>
- [4] <https://www.eembc.org/ulpmark/ulp-cp/scores.php>
- [5] <https://developer.arm.com/ip-products/processors/cortex-m/cortex-m4>
- [6] https://www.ti.com/lit/ds/symlink/tm4c1232h6pm.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-ww&ts=1636819429333&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253Fdistld%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftm4c1232h6pm
- [7] https://www.ximea.com/support/wiki/apis/Jetson_Nano_Benchmarks
- [8] <https://www.rohm.com/electronics-basics/dc-dc-converters/linear-vs-switching-regulators>
- [9] <https://www.design-reuse.com/articles/42191/low-dropout-ldo-linear-voltage-regulators.html>
- [10] <https://www.analog.com/en/analog-dialogue/articles/low-dropout-regulators.html>
- [11] https://www.ti.com/lit/ml/slup239a/slup239a.pdf?ts=1635839833910&ref_url=https%253A%252F%252Fwww.google.com.hk%252F#:~:text=There%20are%20two%20types%20of,maintain%20a%20regulated%20output%20voltage.
- [12] https://www.ti.com/lit/an/slva079/slva079.pdf?ts=1635911592573&ref_url=https%253A%252F%252Fwww.google.com%252F
- [13] <https://www.digikey.com/en/maker/blogs/introduction-to-linear-voltage-regulators>
- [14] <https://www.solarschools.net/knowledge-bank/renewable-energy/solar/how-a-pv-cell-works>
- [15] <https://components101.com/motors/nema17-stepper-motor>
- [16] IEEE standard for systemC: <https://paginas.fe.up.pt/~ee07166/lib/exe/fetch.php?media=1666-2011.pdf#page=36&zoom=100,120,614>
- [17] Battery safety: https://webstore.iec.ch/preview/info_iec62133-2%7Bed1.0%7Db.pdf
- [18] Software engineering: <https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29119:-1:ed-1:v1:en>
- [19] C standard link: <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1124.pdf>
- [20] Robot safety: http://www.paragonproducts-ia.com/documents/RIA%20R15_06-1999.pdf

- [21] [models/tf2_detection_zoo.md at master · tensorflow/models](#)
- [22] <https://www.mathworks.com/discovery/slam.html#slam-with-matlab>
- [23] <https://www.mathworks.com/help.nav/ref/posegraph.html>
- [24] <https://www.youtube.com/watch?v=1GvSPzWagaM>
- [25] <https://www.cnet.com/home/kitchen-and-household/this-is-why-your-roombas-random-patterns-actually-make-perfect-sense/>
- [26] <https://learn.sparkfun.com/tutorials/serial-communication/uarts>
- [27] <https://www.st.com/en/microcontrollers-microprocessors/stm32-ultra-low-power-mcus.html>
- [28] <https://www.ibm.com/cloud/learn/deep-learning>
- [29] <https://steelkiwi.com/blog/python-for-ai-and-machine-learning/>
- [30] <https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a>
- [31] <https://www.tibco.com/reference-center/what-is-a-neural-network>
- [32] <https://www.freecodecamp.org/news/want-to-know-how-deep-learning-works-heres-a-quick-guide-for-everyone-1aedeca88076/>
- [33] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
- [34] <https://becominghuman.ai/how-to-label-image-data-for-machine-learning-and-deep-learning-training-414686d0d1ee>
- [35] <https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks>
- [36] <https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html>
- [37] https://github.com/NVIDIA-AI-IOT/tf_trt_models
- [38] <https://github.com/tzutalin/labelImg>
- [39] <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [40] <https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Nqs/AMb.18ZAm39ERetIY08r3QeLPnB5WRTfTOI0Sr3GAs>