Trash-E: Autonomous Litter Picker Upper

Thomas Greco, Christian Mayo, Chrizzell Sanchez, Alex Rizk

Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816

***Index Terms* — Inspired by a cartoon character from a Disney film, existing products such as the Roomba, and the want to automate trash pickup, Trash-E was born as an electrical and computer engineering project that incorporates new technologies to make a simple task more exciting. Trash-E is a mobile autonomous robot that can go around and pick up anything that is classified as litter. Using Computer Vision (CV), the robot is able to see and identify trash, which is determined by training the robot to recognize objects. This is possible by feeding the robot hundreds of pictures of items that we want recognized. Currently, Trash-E is trained to recognize Red Solo Cups, which it is able to pick up and carry. Trash-E also incorporates Light Detection and Ranging (LiDAR), which is a laser scanning method, to scan its environment. The LiDAR sensor works in tandem with the Robot Operating System (ROS) in order to implement Simultaneous Localization and Mapping (SLAM), which allows the robot to map the environment and avoid obstacles while it is moving around looking for litter.**

***Index Terms* — LiDAR, Ultrasonic Technologies, SLAM, ROS, Deep Neural Network, Robot.**

I. Introduction

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, parties, sporting events, and the like. 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. Trash-E the autonomous litter picker upper can pick objects up using an arm apparatus, pincers to grip the objects, and it has a place to store the objects. Trash-E is driven by two stepper motors which are controlled autonomously through the microcontroller. The robot is also able to recognize litter through a camera using CV and an algorithm to train the system. It is also able to map its environment by using LiDAR and Ros in parallel. All of the intensive processing is done by the Jetson Nano. To be mobile the robot is battery powered. These either need to be recharged or replaced, but there's no way to make the robot run continuously without stopping at some point.

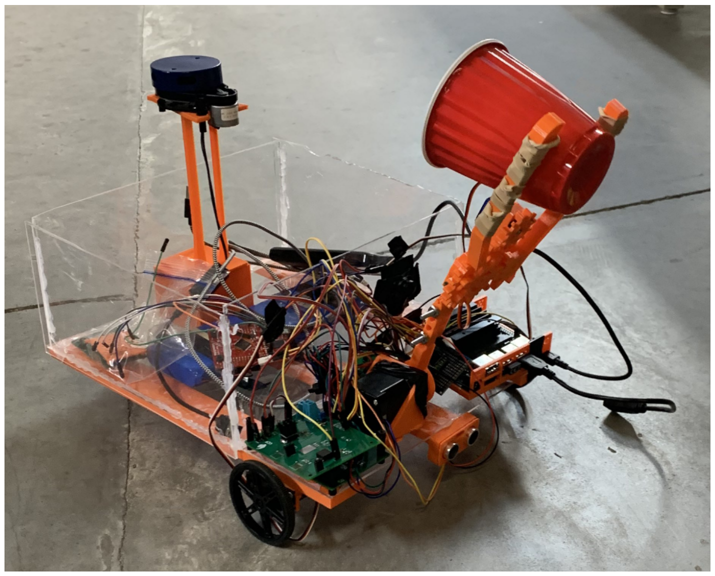


Fig. 1. Trash-E picking up a red plastic cup.

II. System Components

A. Minicomputer

The Nvidia Jetson Nano is the brain of Trash-E, it will operate the object detection as well as the SLAM software simultaneously. The Nvidia Jetson Nano was chosen for its high memory capacity, and high performance compared to its competition. It contains 4 GB of LPDDR4 memory which allows Trash-E to handle multiple complex tasks at once. It also has a Quad-core ARM A57 CPU running at 1.43 GHz and a 128-core Nvidia Maxwell GPU which gives the Jetson Nano its high performance and allows Trash-E to function optimally. The operating system is the Linux Ubuntu 18.04 distribution which makes it compatible with many packages and libraries available for computer vision and SLAM.

B. Microcontroller

For the microcontroller we chose Texas Instrument's TM4C1232H6PMI7 for its abundance of clock signal options. It contains six 16/32-bit GPTM blocks and six 32/64-bit Wide GPTM along with additional features that would aid in controlling the several motors needed for Trash-E to operate. If the Jetson is the brains, the MCU is the body controlling all of the motors. Furthermore, its max clock frequency is 80MHz which easily allows it to listen to commands in real time. Plenty of interface options are also included making for easy communication with peripherals.

C. Motors

Motors used in our robot would be the Adafruit 169 micro servo, Adafruit 154 standard servo, and Nema 17 Bipolar 59Ncm. The micro servo is positional and can apply a torque of 2.5 kg-cm. It is used in the gripper for the opening and closing actions and is more than capable of gripping light litter objects with the aid of rubber bands to provide a high friction surface. Two standard servos with continuous movement and torque of 3.2kg-cm are able to drive the wheels. Lastly, the stepper motor is controlled by an A4988 stepper motor driver in order to move the gripper arm up and down. All Servos are driven by a PWM signal while the stepper motor is given directions by the microcontroller.

D. Chassis

Trash-E's chassis consists of a relatively simple and cheap design consisting of a flat platform with a bucket on the rear end, driving wheels on the front, castor wheel in the back, and a straight robot arm front and center. The front end of the chassis has a flat open section for hardware to be placed. Each piece of hardware has a case to hold it in place, it also allows for it to be glued down. For picking up the trash, the front part of the straight arm is connected to a gripper and the rear is being held by the stepper motor that is encased in its own bracket. LiDAR is contained within the bucket on its own elevated stand in order to scan the surrounding room. In addition, the batteries are contained within the bucket to conserve space for the PCBs and Jetson.

E. Ultrasonic Sensor

The SainSmart HC-SR04 is used for the ultrasonic sensor on Trash-E to detect its distance from an object in front of it. It has an effective ranging distance of 2 cm up to 400 cm.

F. Lidar

The YDLIDAR X2 is used as the lidar on Trash-E that will provide the laser scan data to the Jetson Nano for operating SLAM and creating maps of the environment Trash-E is in. The YDLIDAR X2 provides 360° omnidirectional scanning with a range up to 8 meters. It is a relatively small unit that will fit properly on top of Trash-E and has low power consumption.

G. Camera

The camera used on Trash-E is the SainSmart IMX219 Camera Module. The camera is designed for use with the Nvidia Jetson Nano connected via a camera serial interface (CSI). The camera is used to provide a video feed for computer vision. The camera is capable of up to 1080p video recording at 30 frames per second.

H. Batteries

The batteries used for Trash-E are 4 16.8V 2600mAh Lithium-Ion battery packs that were donated by Smart Charging Technologies, a company that one of the engineers currently works for. These battery packs were placed in parallel, to allow for a higher capacity, to meet the specification of being able to run for 1 hour.

The different components such as the motors, microcontroller, and ultrasonic sensor all have different voltages, so regulators were used to step down to the correct voltages.

I. Regulators

The regulators used on the PCB are the LM1084-ADJ 5A regulators. These regulators were chosen because they have high current capability, where our robot has up to 10A at max load, and were the only ones available that met our needs at the time of purchase. Two regulators are used in parallel to step down to 5V, and a third regulator is used to step down to 3.3V. This allows our robot to be fully mobile.

Although the regulators have high current capability, due to the nature of linear regulators, they were unable to power the Jetson Nano, which has power consumption up to 4A, without triggering the over temperature control. Furthermore, we were unable to find USB-C controllers that fit the specifications to power the Jetson Nano. Therefore, we opted to use a 10,000 mAh external battery bank to power the Jetson Nano.

III. System Concepts

A flowchart is useful for understanding the main processes of the robot, and to visualize the system as a whole.

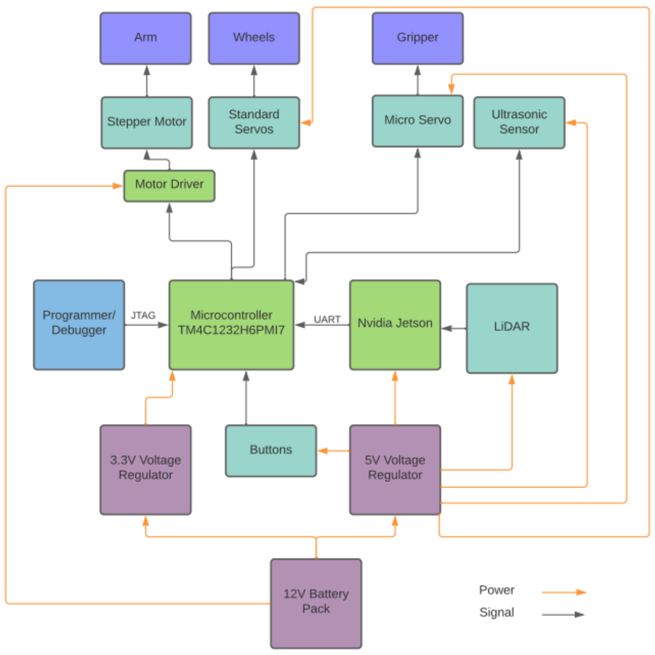


Fig. 2. Block diagram showing the overall design of Trash-E.

As can be seen from the flowchart in Fig. 2, the microcontroller controls all of the physical aspects of the robot. The microcontroller needs data from the Jetson to know what it should be doing.

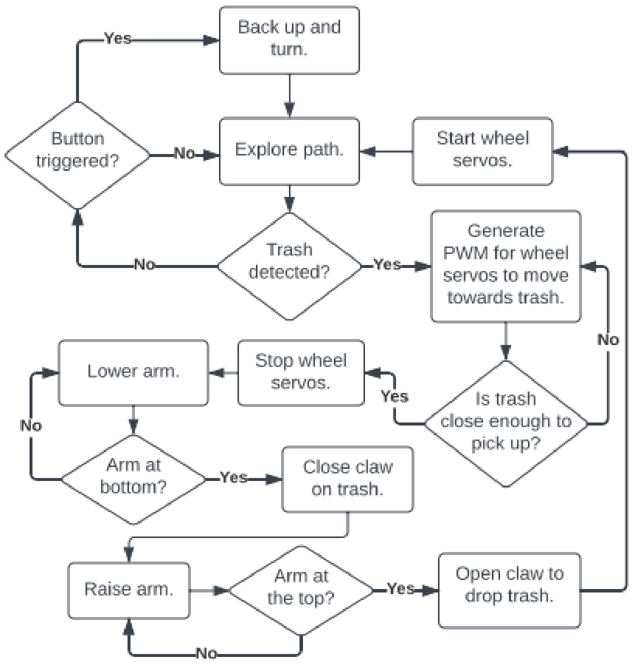


Fig. 3. Flow chart showing the overall process of Trash-E functionality.

As shown in Fig. 3 above, the system of Trash-E is cyclical. From the moment the program starts, Trash-E will continuously look for trash to pick up. Once it detects a trash object, it will run the process of picking it up and then it will go back to looking for trash.

IV. Chassis Design

Since Trash-E is fully autonomous it will not have access to a wall outlet and will need to rely on its own battery. The greatest contributor to power consumption in terms of the chassis would be the weight of each component. To combat this, we chose lightweight materials, or materials that could be designed to be lightweight such as acrylic and PETG. The body of the chassis is made up of three .093” x 11” x 14’ acrylic sheets, and two are cut up into four separate pieces to form the bucket that will contain the collected litter. Three PETG reinforcement bars with 5% infill are used to provide the main acrylic sheet platform rigidity by super gluing them underneath. Brackets for the servos were also made in order to keep them in place under the front end. Attached to the servos are Pololu wheels for standard size servos. These wheels directly fit flush onto the gear of the servo making for a solid hold on the wheels. A castor wheel is also placed back and center to save on weight and give full range of motion to Trash-E. Moreover, using adhesives gives greater freedom in where we could actually place the hardware on Trash-E. In fact, the majority of the chassis is held together by adhesives in order to save additional weight from screws.

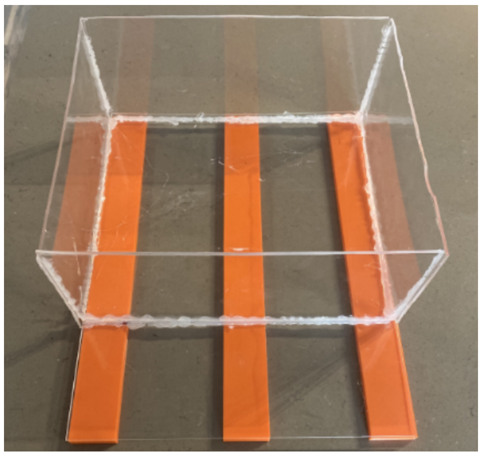


Fig. 4. Chassis design with three PETG reinforcement bars.

In order for Trash-E to pick up litter we used a gripper and arm light enough to not stress out the stepper motor.

The arm was made to be as short as possible, so as not to strain the stepper motor. Additionally, the arm also is light and strong enough to hold up the gripper along with whatever litter is picked up. Two PETG adjustable rails compose the arm, they can be adjusted longer or shorter allowing for a modular design. The two rails are then fastened together with a nut and bolt with the gripper attached on the end.

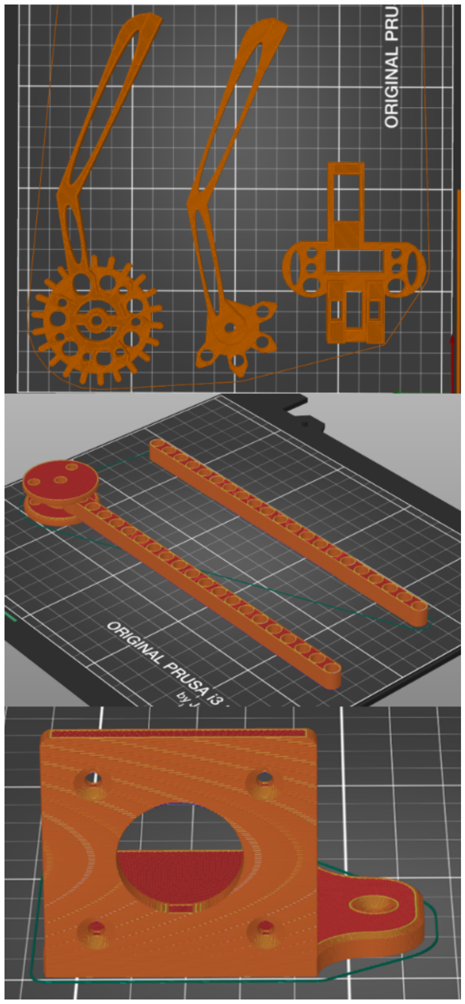


Fig. 5. Rendered images of slicer software estimated prints of gripper (top), adjustable rail arm (middle), and stepper motor bracket (bottom).

The gripper uses a minimal amount of material for its design with infill being at a 5% gyroid pattern. It is driven by a single micro servo through a gear mechanism that allows for the pincers to open 180 degrees, if trained to it would be able to collect most small sized litter items . In order to have litter stay in place while being gripped we used generic rubber bands. Additionally, a bracket holding the stepper motor is used to keep the arm in a stationary position where it has room to operate in a 100 degree arc above it.

Mounting the lidar in the back center of the bucket area, it is elevated with a 8 inch stand to give enough clearance above the 5 inch walls of the bucket. The ultrasonic sensor is also covered so it can be placed below the gripper arm in order to give enough clearance to detect the cup. Casings for the various PCBs are designed to allow for maximum airflow in order to dissipate heat while being able to be mounted anywhere on the Chassis. Lastly, a stand elevated 8 inches from the base of the robot is needed to give the LiDAR scanner enough clearance to generate a clear map of the room.

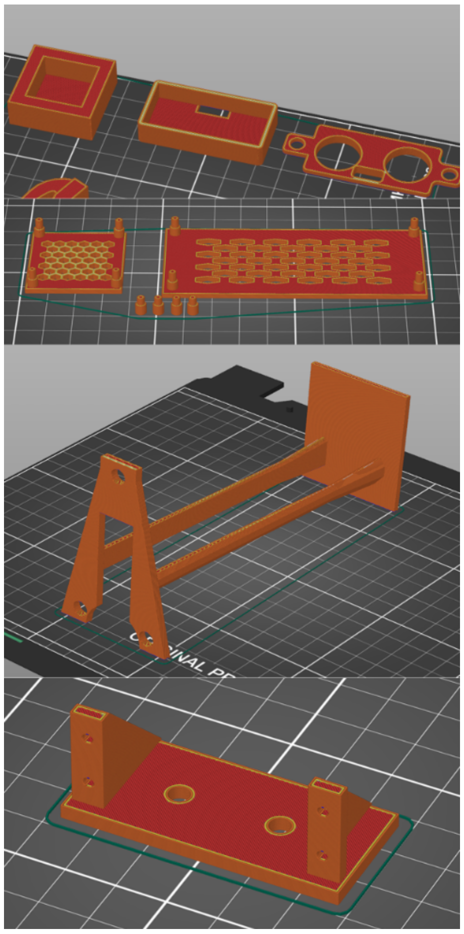


Fig. 6. Rendered images of slicer software estimated prints from top to bottom of ultrasonic sensor case, PCB stands, LiDAR stand, and servo brackets.

V. Software detail

The main goal of Trash-E is to autonomously navigate an environment while simultaneously looking for trash objects that it was trained for and picking them up. In order to accomplish this task, object detection and simultaneous localization and mapping are needed.

A. *Object Detection*

The majority of the object detection software was written using Python and libraries such as PyTorch, Jetson-Inference, and TensorRT. PyTorch is a popular python computer vision library. Jetson Inference is an Nvidia real-time inference deep neural network vision library for the Jetson platform. This library handles a lot of object detection functionalities on the Jetson Nano such as video capture, displaying detection boxes, and running inference of the model and returning detection data. TensorRT is an Nvidia library used for optimizing neural networks on the Jetson platform. The object detection model architecture is the SSD MobileNet V1 architecture. It was trained for detecting red plastic cups using PyTorch in a labeled dataset using hundreds of our own images.

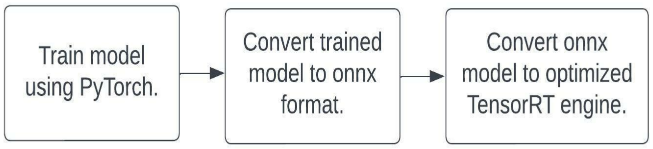

Train model using PyTorch.
Convert trained model to onnx format.
Convert onnx model to optimized TensorRT engine.

Fig. 7. Process of optimizing a trained PyTorch model with TensorRT.

After training, the object detection model is converted to onnx format. Onnx which stands for open neural network exchange is an open format that represents neural networks and allows any neural network framework to be able to convert its own neural network to a format that can be understood universally. Any software that takes in neural networks in the onnx format essentially allows any framework such as TensorFlow and PyTorch to work with it. This is why TensorRT requires the onnx format, so that it can be compatible with any neural network that is able to be converted successfully to onnx. Once the model is converted to onnx format then it is optimized with TensorRT, which creates a TensorRT engine that can run the object detection model. This conversion drastically improves performance of object detection inference on the Jetson Nano. In real-time, the Jetson Nano is performing inference of the object detection at approximately 30 frames per second with a 720p resolution video feed.

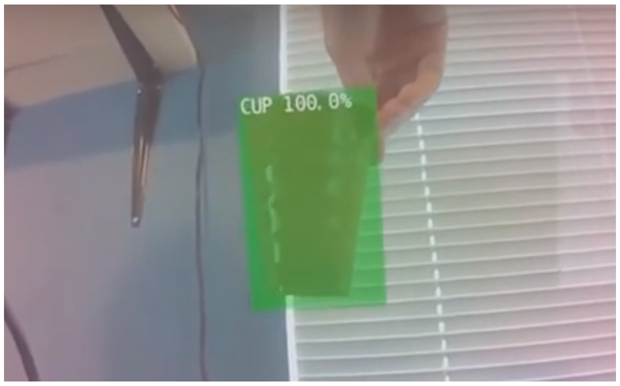


Fig. 8. A red plastic cup being detected by the camera on Trash-E using the object detection model.

The object detection code written for Trash-E does more than just detect the objects in its view. In order to have proper and optimal functionality, Trash-E must approach the closest object in its view. The closest object should have the biggest detection box as shown in Fig. 8, therefore an object's closeness is determined by the size of its detection box area. During the object detection process, the Jetson Inference API provides data such as the detection boxes that include other information such as the class name, confidence score, area, and its center. This data is used to create a list containing all the current detections and sort them by area from greatest to least. Once the closest object is determined, that object is then tracked. Our code uses the object's detection box center to determine where it is on the camera view relative to the 720p resolution, 1280x720. The center value contains a tuple (x, y) which contains the pixel coordinates on the x and y axis. In Trash-E's case, movement is only relevant on the x-axis, therefore the y value is ignored. Using the center x value we determine whether the object is located on the left, right, or center. If the value is less than or equal 750, the object is to the left. If the value is greater than or equal to 800 then it is to the right. If the value is in between the previous two pixel markers then it is centered. The pixel markers 750 and 800 are not exactly in the middle of 1280, this is because the Jetson Nano camera is not placed on the center of Trash-E. It is instead placed more to the left to allow for arm movement. Therefore the values are slightly skewed to accommodate for what the true center is.

B. *Simultaneous Localization and Mapping (SLAM)*

Trash-E would be able to navigate through an environment solely with object detection. However, this assumes that there is always a plastic cup in sight. In the situation that there is no cup in sight or the object detection model is unable to detect a cup, Trash-E would not move and essentially be stuck until a cup appears in its view. The solution to this problem is implementing a way for Trash-E to navigate an environment autonomously. The best way to do this is to implement simultaneous localization and mapping or SLAM for short. SLAM allows us to navigate an environment autonomously using a lidar.

To implement SLAM, the Robot Operating System (ROS) is used. Despite its name, it is not actually an operating system. ROS is an open-source set of software libraries and tools that help build robotic applications. ROS has support for python allowing our code to work seamlessly with our object detection code. The ROS Melodic version is installed on the Jetson Nano since it is compatible with our Ubuntu distribution.

When Trash-E is booted up, a map is generated using hector_slam. This ROS package allows us to create a map of an environment using the lidar onboard Trash-E.

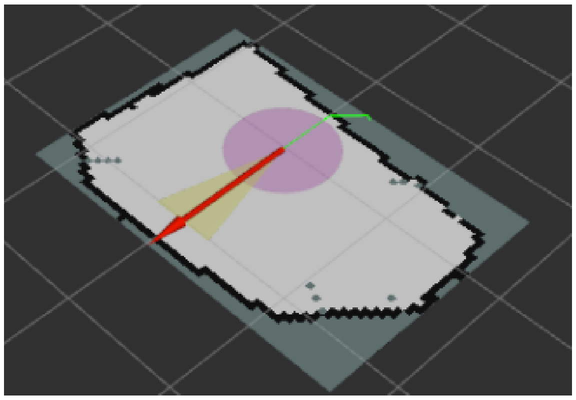


Fig. 9. Generated map by hector_slam of testing arena.

A serial code is sent to the microcontroller so that it knows it is in map generation state. Once the map is generated another serial code is sent to signal the map generation state is done. Then many ROS nodes are initialized for working with amcl, a ROS package that is a localization system for a robot moving in 2D. These nodes facilitate the localization part of SLAM, which involves sending the navigation goals and pose of Trash-E in the environment map to another ROS package called move_base. The navigation goals are where Trash-E should navigate towards and the pose is its orientation in the map. The move_base package when given a goal will attempt to reach it with the robot by creating a path using costmaps that are based on the map we generated of the environment.

Trash-E is navigated with data published by move_base called cmd_vel which contains velocities in different axes for robot movement. The only thing that matters in our case is the angular z velocity, which when negative signals a turn to the right, when positive a turn to the left, and when zero means no turn and continues forward. During SLAM operation, different goals will be generated at different sides of a map as each goal is reached in order for it to explore as much of the map as possible.

C. Guiding Trash-E Using Object Detection and SLAM

The final purpose of both these processes is to guide where Trash-E needs to go. Both these processes can't guide Trash-E at the same time therefore we have a state machine in place that determines which process should be based on certain criteria. UART is used to send data from our python program to the microcontroller so that it could control the servo motors on Trash-E.

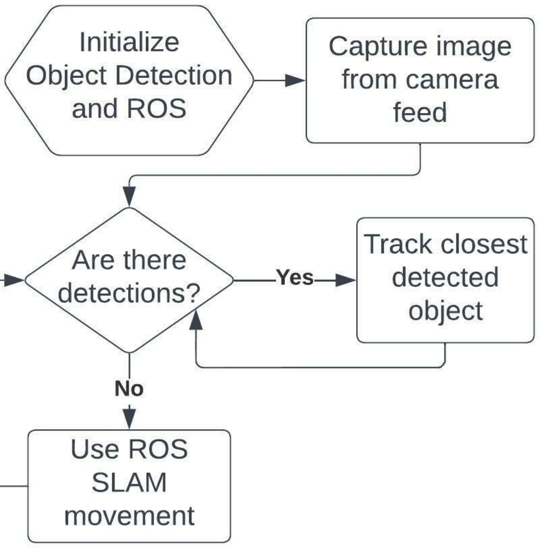

Initialize Object Detection and ROS
Capture image from camera feed
Are there detections?
Yes
Track closest detected object
No
Use ROS SLAM movement

Fig. 10. Process of choosing whether object detection or SLAM control Trash-E movement.

Object detection has priority over ROS SLAM for guiding Trash-E as long as there are cups being detected by the detection model. Earlier it was mentioned how the program determines where in the camera view the object is located. Based on the result of that calculation, we send the microcontroller one of three serial messages that tells it the cup is either to the left, right, or center. When there are no cups in sight, the jetson inference API returns an empty detection list. This condition switches control over to our SLAM software and the cmd_vel data from move_base determines whether Trash-E turns left, right, or goes forward once we check the z velocity value. Based on the z value, we send the corresponding serial message to the microcontroller to tell it to turn left, right, or go forward just like the object detection does. If at any time during SLAM control the object detection detects a cup, it will immediately take control of guiding Trash-E.

D. Embedded Software

With the TM4C1232H6PMI7 providing plenty of GPIO ports, timers, and access to the Tivaware library implementation of Trash-E's features were straightforward to implement. It outputs to several locations such as the motors and communicates with the Nvidia Jetson Nano while reading from an ultrasonic sensor.

All of the embedded software begins with configuration/initialization of peripherals. For example, in order for us to output signals to the motors we first have to enable whichever pins are needed and specify whether they're going to be outputting data or receiving. Furthermore, the timers have to be configured to generate a PWM signal for the continuous servos driving Trash-E, to adjust the speed we simply increase the positive width of the duty cycle. Similarly, the stepper motor outputs a pseudo PWM signal in which the GPIO pin is driven high and low with delays in between to conform to the 1µs duration, 50% duty cycle, of the A4988 stepper motor driver, this can be seen in Fig. 11. Driving the pin high and low this way allows for tracking of arm position letting us know if it's in the up or down position with some slight variance.

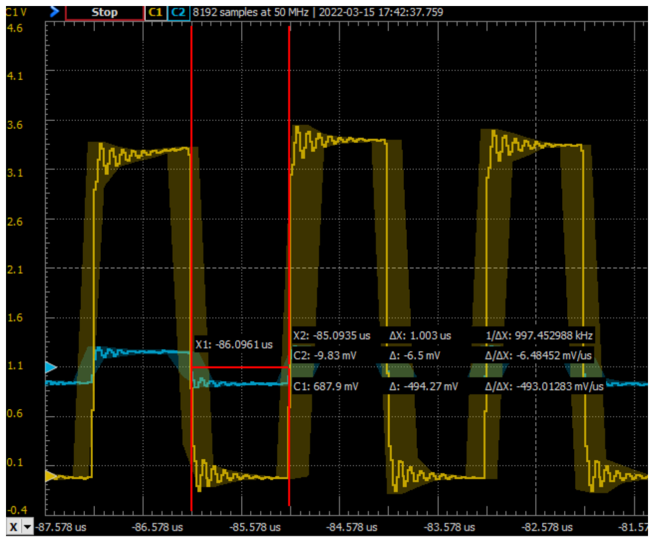


Fig. 11. Waveform showing the 1µs and 50% duty cycle waveform driving the stepper motor through the A4988 driver .

The 9g micro servo is controlled in a similar manner to the continuous servos where it is opened and closed with a PWM signal generated from a timer.

To read data we followed a similar process mentioned in the previous paragraph for configuration. The SR-04 ultrasonic sensor is connected to a GPIO pin that reads the data it puts out. In order to do this we output a high signal to the trigger pin to get ready to read the digital output of the sensor. After asking to read a high pulse is sent through the ECHO pin on the sensor, the length of the pulse gives the distance measured. A timer in periodic mode runs as soon as the rising edge of the pulse is received and stores the duration in a counter variable. Using this duration we can take the frequency of the clock, divide it by a second to get the value of the counter variable in nanoseconds. Then, multiplying it by the value of the counter and dividing by 58 gives the distance in centimeters; this information was given by the SR-04 documentation [1]. While the object detection software is detecting a cup, the ultrasonic sensor on Trash-E was programmed to detect trash up to a certain distance in order to give enough clearance for the arm to be lowered in front of the trash and for the claw to grab it. When the detection software is not detecting a cup and the ultrasonic sensor detects something right in front of Trash-E, it will cause it to reverse away from a potential collision.

Two external buttons are located on the front side of Trash-E's chassis at opposite ends. These buttons are connected to GPIO pins. The purpose of these buttons are for Trash-E to avoid being trapped against an obstacle due to its square shape. When the left corner button is clicked, this will cause Trash-E to receive a signal to reverse and turn right. If the right corner button is clicked a signal will be sent to reverse and turn left.

Finally, communication from the Nvidia Jetson Nano to the TM4C1232H6PMI7 is entirely done through a UART pin. Communication goes in one direction where the Jetson decides what to do and the command is received by the MCU. Several general instructions can be given that tell Trash-E what to do, and it handles the outputs needed corresponding to that instruction given. This includes the motor control and interaction with any peripherals. From there it follows the flow chart in figure 2 on what to do and in what order.

VI. Board design

The microcontroller and power system of Trash-E was implemented on three different boards stacked on top of each other. This decision was made to optimize the limited space allocated for the PCB. The triple stacked PCB allowed for heat dissipation of the linear regulators, by adding planes that were as big as possible, while still being able to fit on Trash-E. It also allowed for having smaller steps between each voltage, to allow for more heat dissipation and to not trigger the over-temperature controls.

USB-C capability was implemented on the boards, in the case that the engineers were able to get access to a controller that fit the Jetson Nano requirements. However, due to supply chain issues, controllers were unavailable for the project.

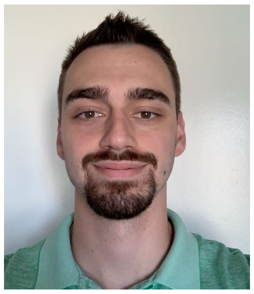
VII. Conclusion

Building Trash-E over the two semesters allocated was a very challenging situation, which gave valuable experiences to every member of the group. Throughout the project the group was tasked with meeting strict deadlines, conducting meetings that were efficient, which allowed the group to maximize time working on the project, and working as a cohesive group.

There were many challenges throughout the process, such as dealing with supply chain issues, where one day a part that the group wanted to order had plentiful stock, and within the week the part would be out of stock, with a lead time of over a year. This also affected components such as the Jetson Nano, where a 4GB version could only be found at more than double retail price. Another challenge faced by the group was the online aspect. With Covid-19 affecting the world, development of the robot had to be done individually by the engineers, with online meetings for progress check ups. Occasionally the engineers were required to meet in person to pass off physical parts and to build the chassis of the robot.

The group aspect of the project was similar to how a company might function, with different engineers working on different parts of the same project. This allowed the group to experience taking on responsibilities that they were interested in, and to work collaboratively to ultimately complete Trash-E.

Engineer Biographies



Thomas Greco is a 22 year-old graduating computer engineering student at the University of Central Florida. Thomas has accepted a job with L3 Harris in Melbourne, FL, as a software engineer.

Christian Mayo is a 22 year-old graduating computer engineering student at the University of Central Florida. After graduation, Christian will be taking a job with Qualtrics in Seattle, WA, as a software engineer.

Alex Rizk is a 22 year-old graduating computer engineer at the University of Central Florida. After graduating Alex will be starting an embedded software engineering job with Lockheed Martin in Marietta, GA. In the future, he plans on pursuing a career in the fields of Robotics and Mechatronics.

Chrizzell Jay " CJ" Sanchez is a 23 year-old electrical engineering student , with a focus on Power and Renewables at the University of Central Florida. CJ has accepted a job with Affiliated Engineers in Gainesville, FL where he will perform MEP consulting.

Acknowledgement

The authors wish to acknowledge the assistance and support of Smart Charging Technologies, who donated batteries to be used in the robot, and allowed the engineers to use the lab to test and construct the robot. As well as the generosity of colleague Noah Zinn for lending an Nvidia Jetson Nano 4GB Developer Kit.

References

[1] Elec Freaks. (n.d.). Ultrasonic ranging module HC - SR04 - SparkFun Electronics. Retrieved April 7, 2022, from https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf