The Embellisher

Authors: Ritu Pathak, Ella Lee, Hirani Sattenapalli
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—Unclean streets in urban and underdeveloped areas lead to increased pollution, sanitization hazards, intensive human labor, high cleaning costs, and public safety concerns. This paper presents a robotic system capable of identifying garbage on sidewalks, picking up and collecting said garbage, navigating autonomously, and avoiding obstacles. Although there exist commercialized automated robot cleaners in water settings, for land there does not yet exist a solution beyond proof of concept. We aim to provide a solution using object detection providing 0.95 mAP, navigation and pickup processes at 90% efficiency, and successfully perform trash pickup within 45s.**

***Index Terms*—Design, Robot, Object Detection, Object Collection, Path Planning, Obstacle Avoidance**

1 INTRODUCTION

Unclean streets in urban and underdeveloped cities and towns are normal to where residents in such areas no longer view such areas with antipathy, but rather ambivalence. We observe these roads and sidewalks are often littered with trash, mud, and plant waste. Pedestrians have to deal with discomfort from unpleasant smells and sights of garbage on top of safety issues posed by slippery roads. Subsequently, we saw the urgency of promoting a more pedestrian friendly environment. We aim to develop a prototype of a cleaning robot that can autonomously navigate streets and pick up trash.

The robot will be engineered to navigate sidewalks while also being able to identify and clean up the streets along the way. Our robot is targeted to local and state governments, the authority in charge of street cleaning programs, who not only have to deal with the increased pollution present in the streets, but must delegate intensive human labor to help mitigate the existing amount of pollution. The costs for cleaning programs which delegate these tasks are high[2], but of course, it must be done because the trash is a sanitation hazard to the public. This robot can target these needs by identifying garbage on sidewalks and picking up and collecting garbage to reduce pollution, navigating autonomously to decrease the level of human labor, work within a restrictive budget to reduce cleaning costs, and avoid obstacles to maintain public safety.

There are a couple existing solutions which try to do this, but have not reached commercial success. One such example is DustClean; however, it is not widely deployed due to high manufacturing costs. Furthermore, its primary use is for small particles rather than large pieces of trash. Our robot aims to have manufacturing costs under $600 while having a trash storage compartment that is more scalable to various loads of trash. For the purposes of this design, we specify our trash components to be soda cans, plastic water bottles, and crumpled paper.

2 USE-CASE REQUIREMENTS

To resolve the problems mentioned in the previous section, we see the need for a robot that is able to autonomously identify garbage on the streets, pick it up, and collect it while avoiding obstacles. To achieve these goals, we aim to define the following use-case requirements:

First, The robot should be able to identify trash objects with at least 95% mean average precision for YOLOv7 and 80% for tiny YOLOv7. For reliable operations, which we claim to be two standard deviations away from the mean, a high precision of at least 95% is needed to classify trash objects. However, the reason the Tiny YOLOV7 can have less precision is that it can run inferences faster than the standard YOLOv7 due to its compact size. This sacrifice is necessary for high performance to identify trash objects quickly and collect them. However, both would have 100% recall rates because having no false negatives would avoid misclassification of living creatures as trash items and further safety issues such as injuries and psychological distress. This will promote public health, in particular for pedestrians and animals within the environments the robot is operating in.

Then, the robot must avoid obstacles with a 95% success rate and navigate the streets without disrupting humans or animals. As mentioned above, public health and safety are the most crucial factors to consider when designing a street-cleaning robot as it should not put pedestrians into risks of injuries or property damage. 95% represents two standard deviations away from the mean. By maintaining performance within this parameter, the robot will be able to operate reliably and consistently in normal conditions without unexpected interference. This predictable behavior ultimately elevates the user experience and increases user comfort during interactions with the robot.

The robot should pick up the given trash items at least 90% of the time. Reaching a 100% pick-up rate was unrealistic as the process slows down with the robot double-checking that the item is properly picked up. For real-world use purposes, a 90% success rate with some small error margins captures most trash while also maintaining a relatively efficient work pace. In effect, the robot can promote public satisfaction while also raising awareness of hygienic living conditions and public health through clean-up tasks.

In addition, the robot should be able to operate in ar-

eas bigger than 4 feet by 8 feet. These testing spaces model the rectangular shapes of real-life streets, which enables an expansion of experimentation to full-scale in future deployments. Additionally, testing at a smaller scale is practical in terms of project time and budget and ensures an application that benefits the communities once it is commercially designed and produced.

The robot must pick up trash in less than 45 seconds to maximize cleaning efficiency and minimize disruptions to the environment. The robot will be situated in the bustling streets with heavy foot traffic in real life. If it spends too much time picking up the objects and completing the tasks, it may block people passing through the area and cause inconvenience. Also, optimizing the robot's pickup time helps reduce unnecessary power consumption and the recharging time of batteries, which decreases overall operational costs.

Also, the robot must be operated within a span of 2 to 4 hours. We want the robot to be unobtrusive but able to collect sufficient amounts of trash at the same time to deposit all at once within the given timeframe limit so that it does not have a prolonged presence on the streets.

Finally, the battery should last about 2 hours. The existing commercial cleaning robots, such asDeebot X1 Omni [6] and Roomba by iRobot [7], have a maximum capacity of running for 2 to 3 hours. A battery life of at least 2 hours will ensure high performance in cleaning tasks, where the robot can operate without needing to swap batteries in the middle of the shifts and complete the cleaning tasks without long-term idling to recharge the battery. This also avoids any possible inconvenience to the general public, as the idle robot sitting on the street could disrupt any activities going on in the spaces.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

The block diagram provided in the Appendix, Figure 4, gives a diagram of the computation devices that we are using, interconnections, and all of the hardware and software components. The Jetson Nano Orin will be running the object classification pipeline. The Nano takes in camera feed from the e-CAM50_CUNX and is connected to the Raspberry Pi 4 via ethernet. The Raspberry Pi will use a dc stepper motor hat to connect the four motors used for motion, 3 motors used for the pick up mechanism (conveyor belt and rotating device), and the ultrasonic sensor. The provided legend gives details on the specifics of the components used.

The flow chart, as referred to in Figure 1, shows an overview of the robot's operation. First, it starts in sleep mode. Then, it rotates the camera, and if it detects the trash component, it centers itself around the trash object, initializing the motors of the roller and conveyor belt to go straight towards the garbage component by increasing speed. Then, it confirms the object has been collected by checking whether the component is invisible in the camera feed, then moves back to its original reset state. If no trash is detected, on the other hand, the robot moves 3 feet forward while incrementing the counter. During this movement, if an obstacle is detected, then the robot should move in a straight direction to avoid the obstacle. If the counter hits 4, then it automatically goes into sleep mode.

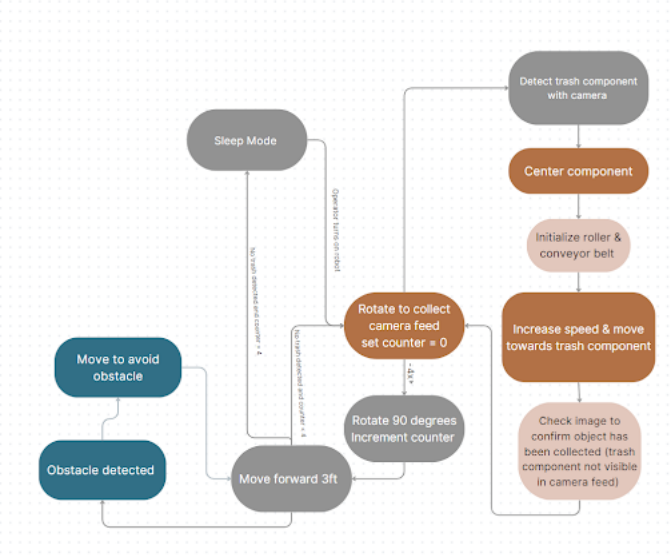

Figure 1: Flow chart for the system.

4 DESIGN REQUIREMENTS

Our design requirements are derived from our use case requirements.

Our first requirement is based on our ML subsystem performance, namely, we want our inferencing pipeline to support a camera frames per second (FPS) rate of at least 15 and return bounding boxes surrounding trash objects at a confidence interval of 0.68, a precision of 0.95, and recall of 1.0. These metrics are very important and reflect what we want our user experience to be like. For example, we want to support 15 FPS because we want our object detection to be as fast as possible, especially if the robot will be spinning in space to find other objects. We would rather be limited by physical spin ability than compute power, and 15 FPS seems like a good metric for that. We want the confidence interval to be at least 0.68 so that we are not accidentally detecting non-trash objects, and 0.68 is roughly one standard deviation above the mean. Our precision and recall metrics are consistent with our use case requirements.

An additional design requirement based on our Movement and Pick-Up Mechanism subsystems is that we want the speed of our robot to be roughly $0.92 ft/s$ when navigating to a detected trash object. We use the equations below to justify our quantitative metric.

$$c = \sqrt{a^2 + b^2} \tag{1}$$

where a and b are sides of a right triangle and c is the

hypotenuse, which is also the longest possible distance the robot may have to travel. Since our largest testing environment is 4ft x 18ft, from (1) we get the longest possible distance to cross to be approximately 18.4ft. To stay within our time bounds, we need to allocate some time to reach the trash and to pick up the trash. We allocate 20s to reach the trash and 25s to pick up the trash and account for obstacles which may be in the path.

$$S = \frac{D}{T} \tag{2}$$

where S is the speed in ft/s, D is the distance in ft and T is the time in s. We thus calculate our speed as $S = \frac{18.4ft}{20s} = 0.92ft/s$.

Another design requirement we made note of is that the power supply must supply at least 28000 mAH to last approximately 2.9 hours when supplying 5V @ 3A to both the Jetson Nano Orin and the RPi4. This is important because both devices need 5V @ 3A of input to run successfully. We refer to (3) as to how we built this relation.

$$\frac{mAH}{1000 * Amps} = Hours \tag{3}$$

where mAH is the battery capacity, $Amps$ is the current and $Hours$ is the overall time, where each variable reflects their units as well. We note (3) holds in systems where the input and output voltage are the same. Based on (3), we see if we have 28000 mAH and need to supply a current of 3A, we end up with roughly 2.9 hours. 28000 mAH is a fairly common battery capacity which results in a battery life of at least 2 hours and thus will allow the robot to roam for at least 2 hours, leaving room for energy lost due to heat dissipation.

5 DESIGN TRADE STUDIES

During the design of our product, we considered multiple options for each aspect of the robot, before making the most appropriate design choice. The primary features that we ran trade studies for are split into hardware, software, and construction/ build components. They are detailed below:

5.1 Hardware

5.1.1 Computation Device

When considering the computation device meant to run the object classification, we looked at several different products. Inspired by the tech talks released near the beginning of the capstone project, we thought heavily about the AMD Kria KR260 robotics starter kit. The Kria KR260 starter kit provides users with an Arm Cortex processor that can run up to 1.5GHz and a dual core Arm Cortex R5F real-time processor up to 600MHz [1]. It also provides a 4GB 64-bit system memory, 4 gigabit ethernet RJ45 ports, 4 usb interfaces, and also includes expansions referring to a 40-pin raspberry pi hat header with 26 input/ output ports. We also considered different Jetson Series. First, we looked at the Jetson Xavier and compared it against the AMD Kria KR260. Comparisons have shown that the Kria KR260 has a 5x productivity gain and up to 8x better performance per watt against the Jetson Xavier AGX[8]. The primary drawback with the Kria KR260 was the cost: it would take up a bit more than half of our allotted budget ($350). This was a heavy drawback, especially when we had to consider the budget needed for the construction of the robot.

As there were several Jetson Series in inventory, we decided to consider some of the other starter kits. The Jetson Orin Nano delivers up to 70 and 100 TOPS of AI performance, especially in the smallest Jetson form factor. The Orin Nano performs almost 3 times better than the Jetson AGX Xavier and the Jetson Xavier. The Jetson Nano Orin uses an 8-core Arm Cortex 64-bit CPU[8]. The Jetson Agx Xavier has an AI performance value of around 32 TOPs and provides an 8-core ARM Carmel. We also considered the Jetson AGX Orin, which had a much higher AI performance value (275 TOPS); however, it is priced at a significantly higher rate ($2000). Hence, we chose to use the Jetson Nano Orin as it was the most optimal choice considering cost, performance, and computational power.

5.1.2 Motor control Computation Device

The motor control computation device is meant to control the movement of the robot overall (the four motors that will be used to move the robot), avoid obstacles in the process (control the sensors needed to detect obstacles), and the motors needed for the pick-up mechanism (three motors are used for this and will be detailed in the construction section)[3]. We primarily considered three main devices: the Raspberry Pi, Arduino, and the Jetson Nano Orin. With the Jetson, the idea was to run all computation in parallel with the object classification. Running all of the computation on one device (the Jetson Nano Orin) did not seem ideal as this would slow down performance significantly and could lead to lags/ delays and higher latencies. Hence, we decided to avoid using the Jetson for motor control, and instead use a separate compute device - either the Raspberry Pi or Arduino.

Once it came down to the Arduino/ Raspberry Pi, this was a fairly straightforward decision. In addition to being able to control seven motors and one sensor, we required our additional computational device to run ROS. The Robot Operating System is a very useful set of software libraries and tools meant to build robot applications. As part of our capstone project, we wanted to use ROS to add complexity, provide additional software for us to learn, as well as simplify part of the motor control and robot interface as prior research demonstrated that ROS is a great way to work with and build robot applications. Arduinos work well controlling motors and provide a simple microcontroller, but significantly limit our scope for extended robotics applications. Raspberry Pis have great processing power (around 1.6 GHz) while the Arduino runs at around 16MHz[13]. Hence, as Raspberry Pis can run ROS and

provide their own operating systems, we decided to move forward with the Pi.

Once we decided on using the Pi, our main considerations were between the Raspberry Pi 5 and Raspberry Pi 4. The trade offs, in terms of computation and performance, between the two are that the Pi 5's CPU runs at a higher clock rate, and supports a GPU at 800 MHz (compared at 500 MHz for the Pi4)[5]. The CPU and GPU of the Pi 5 are noticeably more efficient than the Pi 4's. The Pi 4 does have a lower power consumption (which would be a pro in terms of increasing battery life)[4]. However, after testing out the Pi 5, we realized that the Ubuntu distribution that is supported by ROS versions that have been released, are not compatible with the Pi 5 (only supports Ubuntu 23.10). The ROS distributions that have been released are compatible with Ubuntu 22.04 and below; however, there is no planned ROS release for Ubuntu 23.10. From research, we noticed that ROS works really well on Ubuntu Linux; hence we decided not to look for another operating system. Overall, we decided to use the Pi 4 for our motor control computation device as it can support ROS distributions.

5.1.3 Sensors:

The robot is meant to avoid obstacles while it is moving in its defined environment. Obstacles, earlier, have been defined to be larger than 1.5 ft which is the approximate height of the robot. A sensor is needed to detect these obstacles and we considered several different types of sensors while choosing the most suitable sensor. Our first choice was to use object classification (the tiny yolo v7 model) to detect objects in the environment.

There are several difficulties with the option of using Object classification to avoid obstacles. Most object detectors, including the Tiny Yolo models, are trained for a specific resolution of input and do not support multi-scale training. As there is unpredictability in terms of the obstacles that the robot will encounter on a daily basis, training the model to ensure the use case requirement of 95% obstacle avoidance is an exceedingly difficult task. This will also require large datasets and computational power, and there could be imbalances in the different categories of the objects that the model must be trained on (to cover all the ranges of obstacles the robot will potentially encounter - human beings, buildings, large boxes etc.).

As the usage of object classification for obstacle avoidance is highly unrealistic considering our time to train and the computational power of our compute devices, we decided to consider the usage of sensors to detect objects (of ¿ 1.5"). We chose to use an ultrasonic sensor which uses sound waves to measure the distance to an object. This works efficiently to detect objects at distances from 7 inches to around 11 feet (this can change depending on the sensor used). Other sensors we considered include: An electromechanical light switch, which requires physical contact with the target object (inefficient in defined environment). A pneumatic which uses compressed air and a sensitive diaphragm valve to detect the presence of objects (this is not always reliable as the instruments must be functioning perfectly well to detect objects). Magnetic sensors that are actuated by the presence of a permanent magnet within a sensing range.

The sensors described above and others do not precisely give a distance measurement and are not perfectly reliable. Hence, we decided to use an ultrasonic sensor.

5.2 Software

5.2.1 Object Classification Model Architecture

We settled on the Tiny YOLOv7 model over other models, since it's designed to run on lightweight compute devices, which is helpful for keeping our costs low. However, because it's compressed to a lower size, its accuracy is lower when compared to other pre-trained models, such as R-CNN and non-tiny YOLO models. We felt this was an appropriate trade to make for our use case however, since the model is still empirically ¿90% accurate with a recall ¿70%. Ideally we want this model to reach 100% recall sacrificing some amount of precision to avoid false negatives as much as possible. Since our robot just needs to be autonomous and perform reasonably well at picking up surrounding trash, we feel these metrics are sufficient. Furthermore, we will use the YOLOv7 after the Tiny YOLOv7 detects something, to keep our false positive rate low without sacrificing too much detection speed. We also could have trained a TensorFlow model, and then used TFLite for the model to be optimized on edge devices; however, research during one of our member's internships showed the speed of TFLite models to be substantially slower than Tiny YOLOv4 (30 FPS vs 20 FPS), and the Tiny YOLOv4 has been shown to be slower and less accurate than the Tiny YOLOv7. Additionally, unlike TensorFlow models, YOLO models are able to detect several objects in one image pass, which is useful for trash detection. Aside from the Jetson Nano Orin, we were considering other lightweight edge devices, mainly the Raspberry Pi. However, the Jetson Nano Orin has a more powerful GPU unit, making it more suitable to ML projects requiring high performance compute. We were additionally thinking of using AMD products, but similarly the Jetson Nano Orin seemed more suitable to our purposes with more readily available resources.

5.3 Construction/ Build

5.3.1 Pick-up mechanism

The type of pick up mechanism to use for the robot was an important component of the project as this defines a major portion of the time it would take to collect a single object. The primary mechanisms we considered include the scoop mechanism, a robotic arm, and a roller mechanism (that consisted of some rotating device that would pull components into the robot).

The scoop mechanism consists of a rectangular box that is close to the ground. It is connected to the robot with a few motors that will control its movement. Once a trash

component is located, the robot will move towards it, and the scoop will move outwards (away from the robot) and pull the trash component into the storage area. The primary drawbacks with this mechanism are that it has high latency (takes a significant amount of time for all motors to be moved accurately - around 90 seconds to 2 minutes) and has greater complexity in terms of attaching the motors in specific locations to have the scoop move in properly. There is also the potential error of not catching the trash object in the scoop (ex. pushing away the object).

The robotic arm mechanism consists of a robotic arm used to pick up each trash component. There is a high latency associated with the robotic arm (around 2-3 minutes) to have the arm move to the object, pick it up, and place it in the storage area. This mechanism also faces similar drawbacks of potentially being unable to pick up the object or missing the object (due to its shape). The timings for the mechanisms were approximated from past projects that used a scoop/ robotic arm for different applications.

The final mechanism we considered was the roller/ rotating device. There were several types of implementations we saw of this. One of the implementations was the use of several Caster wheels that sit close to the ground. The rotor moves these wheels and pulls in the trash components from underneath. Due to the force of the wheels, they are shot back to the storage compartment. This mechanism takes around 20 to 30 seconds to collect the component. However, the primary drawback is the danger it presents to small animals/ young children near the robot. The other implementation we considered was to use a rotating shaft (which is provided in our design sketch), that will roll in the trash and be placed in the storage compartment through a conveyor belt. We are expecting the trash to be collected inside the robot around 20-30 seconds, and around 20 seconds more for the trash to be placed in the storage area via the conveyor belts. The roller has a three inch radius and does not move as fast as the Caster wheels, and so presents a safer intake mechanism. This design is described in greater detail in the implementation section.

Overall, we decided to use the rotating shaft mechanism as it takes less time to collect trash components, in reference to the other mechanisms described, and is a safer mechanism that can be justified for use in the real world.

5.3.2 Types of Wheels

The types of wheels we used for the robot were an important consideration primarily when thinking about real world applications of the capstone. Although this project is intended to be used as a proof of concept and meant for immediate deployment, we thought it necessary to consider types of wheels for different terrains. Our primary considerations were tracked wheels and Mecanum wheels.

Tracked wheels: run on continuous tracks instead of wheels and work well with soft, low friction, and uneven ground. They provide good traction on soft surfaces, but aggressive tracks can damage paved surfaces and are prone to a higher set of failure modes (snapped/ derailed tracks).

Mecanum wheels: provide traction and stability for omnidirectional movement (rotation around a fixed axis). Using a variable wheel size can provide flexibility with the size of the robot.

After considering the defined environment and recognizing that most of the places we targeted in our use-case requirements had street trash on paved roads, we decided to use Mecanum wheels. In addition, mecanum wheels provide greater flexibility in terms of design as they are less prone to different failures and can be tested well in our defined testing environments.

5.3.3 Build Material

In terms of the actual construction of the robot, we considered several different materials. We looked through past deployed and prototyped street and waterway trash collecting robots, and recognized that most were made out of plastic/ were custom made through a manufacturing company. As this posed difficulties in terms of cost and time, we decided to build the robot out of aluminum extrusions (provide a very strong and sturdy foundation/ frame for the robot) and acrylic boards. We will be 3D printing the actually rotating device, as some of the newest 3D printers on campus have produced some reliable and sturdy products. The rotating device must be strong and durable (to collect the trash components), and must match our defined dimensions. Hence, 3D printing the rotating shaft, using the most recent 3D printers was the best option for us. In addition, we will be using acrylic boards over laser-cut wood to cover our electronics and fill our base as there is a greater availability of these boards in campus workstations. Overall, we considered our materials (on campus and online - budget) in great detail while defining the final design sketch and the dimensions.

6 SYSTEM IMPLEMENTATION

Our implementation is split into three major components: detecting trash, motor control, and avoiding obstacles. For our overall pipeline, we plan to use a camera (specifically the e-CAM50_CUNX) to take a picture of every direction the robot faces, as the robot will have a simple path finding algorithm. Specifically, the robot will take a picture of what lies in front of it, if the robot sees trash, it will move forward, else it will turn 45 degrees to the left. If the robot has turned a full 360 degrees, it will also move forward. Also, it will be smart enough to avoid moving forward in a direction which has obstacles. A basic CAD model of our overall system can be seen in Fig 2. It includes trash storage separated by an acrylic board from the conveyor belt and roller pickup mechanism. It includes mock-up wheels and a blue box for all hardware components.

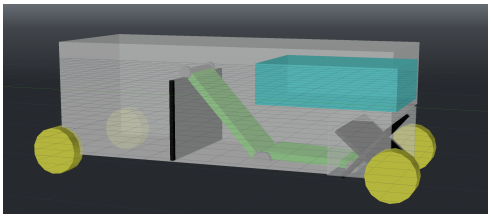

Figure 2: CAD Model

6.1 Object Classification

In order to detect trash, we plan to tune two pre-trained object classification ML models on a custom dataset including classes for soda cans[9], plastic bottles[10], and crumpled paper[11]. We will combine these three datasets together to form our custom dataset, using Roboflow to re-annotate classes as needed. In case the existing soda can dataset performs poorly due to its intensive use of backgrounds, we also plan to build our own soda can dataset with pictures of soda cans taken from an iPhone. We will annotate these using Roboflow on backgrounds more representative of our use case. The two models we plan to use are Tiny YOLOv7, which works well on lightweight compute devices but sacrifices accuracy for speed, and YOLOv7, which is very accurate but runs slower. We will be running both on the GPU of the Jetson Nano Orin. We have been training both models using Google Colab Pro on the A100 GPU. Once we run out of compute units, we plan to switch to using the GPUs on the ECE wing clusters.

To summarize our flow: (1) the input of the ML subsystem will be a constant image stream captured by our e-CAM50_CUNX camera connected via GPIO pins to the Jetson Nano Orin. (2) We plan to use TensorRT-Torch to run inference using the Tiny YOLOv7 model trained on our custom dataset on. (3) If the Tiny YOLOv7 model detects trash components above a certain threshold for a particular image, we will then run the same image through the Tiny YOLOv7 model to reduce false positives. (4) The output of this subsystem will be bounding boxes around detected trash objects above a certain threshold.

The pipeline for this subsystem can be visualized through the left half of the architecture diagram, provided as reference in Fig 4.

6.2 Movement and Path Finding

In order to control motors used for movement and trash pickup, we plan to use the Raspberry Pi4, since we believe the CPU unit on the Jetson Nano Orin won't be enough to do everything. We plan to use ROS to control all of the motors (7 DC motors in total throughout the full system). We do not plan to use a complicated path finding algorithm to navigate to trash, rather we plan to simply see if we detect trash in one of 8 directions, if we do, we move forward to it, otherwise, we move straight relative to the robot, unless we detect an obstacle. We plan for our robot to have four Mecanum wheels with one motor per wheel, in order to facilitate movement in 8 directions. Additionally, using Mecanum wheels should allow our robot to spin in place easily while surveying different directions.

To summarize our flow: (1) the input of the path finding subsystem will be bounding boxes from the ML subsystem. (2) If there are non-empty bounding boxes and no obstacles, use ROS to have the four wheels move forward, else use ROS to spin the robot 45 degrees. (3) If the robot has spun 360 degrees, then the robot should move forward. (4) This subsystem will not have an output.

There are two other considerations we have for our path finding algorithm that need more research: (1) using QR codes to model our boundary of our space (2) remembering the locations of detected trash objects. As these are both aspects connected to obstacle avoidance, we will elaborate on these components in that section.

6.3 Pick Up Mechanism and Display Subsystem

For our pick up mechanism, we plan to have 3 motors in total, with one controlling a roller like mechanism and the other two controlling two conveyor belts. Fig 3 is provided for reference.

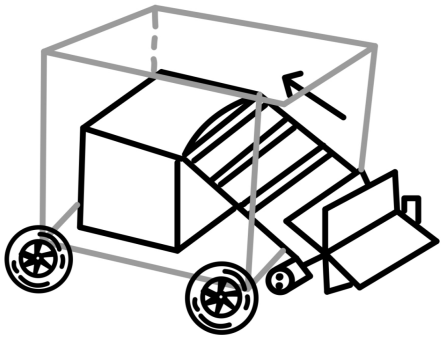

Figure 3: Pick Up Mechanism Subsystem

We thought this method would work the best for our use case, picking up soda cans; plastic water bottles; and crumpled paper, due to the shape of these objects. The roller works as a "one size fits all" mechanism, as long the objects are large enough, and the conveyor belt pulls the trash objects into the trash storage component of the robot. We also plan to have a HC-SR04 sensor near the top of the storage component to detect when a trash object has been collected. The Raspberry Pi5 will use this information to update a digital display of trash objects collected and to detect when the robot should stop.

To summarize our software flow: (1) the input of the display subsystem will be positive detections from our HC-SR04 sensor. (2) The RPi4 will update the display counter when receiving input. (3) This subsystem stops the full system once a certain number of trash objects have been collected.

6.4 Obstacle Avoidance

In order to avoid obstacles, including the boundary, we plan to include an ultrasonic proximity sensor, the HC-SR04, at the front of our robot, as high as possible, connected to the Raspberry Pi4. This way, the robot would be able to detect obstacles roughly of its height, which is our limited scope for this project. This is to mimic stationary hazards such as traffic cones denoting construction and/or potholes, along with stationary humans. As designated by our movement subsystem, if this sensor detects an object, the robot will turn 90 degrees to the right. The same obstacle should not be detected after the turn. In other words, we currently plan to have the boundary be the same height of the robot. We are still in the process of researching using QR codes to model the boundary of our space, and have designated this goal to be post MVP.

Additionally, we are also researching potentially remembering the locations of past object detections to find a better way for the robot to know if an area has been cleaned or not, particularly in the case of obstacle avoidance. We have been unable to come up with a good way to do so however, and thus to get a working product, we have also decided to make this a post MVP goal.

To summarize our flow: (1) the input of the obstacle avoidance subsystem will be positive detections from our HC-SR04 sensor. (2) The RPi4 will use ROS to stop existing robot movement and spin the robot 90 degrees to the right, becoming the new "forward" direction. (3) This subsystem does not have an output.

6.5 Public, Social, and Economic Factors

We considered public health, safety, and welfare in detail when defining our design and scope. Considerations of public health, sanitation, and pollution that results from street-garbage/trash were our main sources of intention and drive behind the product. The end goal of the robot is to increase public health and welfare by collecting street trash and in the process, inform the public about the importance of maintaining clean streets.

In terms of public safety, we considered this in great detail while finalizing our design. After we completed major portions of research for the robot, we finalized on two designs: a roller mechanism that consists of 6-7 Caster wheels and a rotating fan that works in combination with two small conveyor belts. We chose to use the conveyor belt and rotating fan as our main pick-up mechanism for the robot. The main reason for this design decision was that it provides greater safety for stray animals, pets, or small children that might accidentally put their hands into the robot. Although this will be significantly advised against, this is a potential scenario we must be aware of. The roller with the Caster wheels poses a significant threat to stray animals/small children as there is no in-expensive way for the wheels to stop immediately when in contact with skin/-fur. The wheels also squish/squeeze the trash component as they pull it into the storage area; hence, it presents a danger to any stray animals/small children. The rotating fan will not move as fast as the wheels and runs on a DC motor, hence it will mostly get stuck (if a stray animal runs into it).

In terms of social factors, our design will not exclude certain groups of people when active for tasks. The robot is meant for use for society at large without excluding demographics, although is best purposed for underdeveloped communities and urban areas. It will also benefit the government as they can exhibit their continuous contribution to technology and innovation to the residents, as they would be the ones paying for the robots to go to the streets. They would also help environmental organizations in reducing waste while minimizing costs for cleaning with our autonomous garbage cleaning robot. It will also span cultures as it can adapt to different cultural elements with the addition/removal of features and settings by preferences. The robot also aligns with the existing sustainability policies set by governments

In reference to economic factors, in regards to production, our product aims to have low enough cost to be useful to state/local governments. Specifically, the product is meant to last for a long time and be relatively self-sufficient, with a one time cost per purchase rather than a salary based cost necessary for humans. However, since each robot can only cover so much land in a couple hours, the consumers must buy several. In terms of distributions, none of our components are involved in a supply chain shortage, so mass production would not be an issue. It may take some effort to release the robots into the street however, such as some initial setup cost to know which area to focus on. In terms of when these products should be produced, this product is useful year round but is not meant to run in inclement weather. Therefore, we anticipate the robot to have more usage in the summer, and thus believe the supply of robots should be higher in the summer season.

7 TEST & VALIDATION

To ensure our garbage-cleaning robot meets the use-case and design requirements, we aim to evaluate the accuracy and validity of the following two components. Following these validations, we will test the entire system with three integration phases to measure its performance levels thoroughly.

7.1 Tests for Object Classification

The performance metrics for the object classification are false positives and false negatives. To do so, we will connect a camera E-CAM50-CUNX to Jetson Nano Orin and run machine learning models to see which objects Orin is able to detect and identify as trash items. First, there will be 15 correct trash components provided. We calculate the accuracy rate and false negative rate (model classifying a trash component as non-trash items). Similarly, we also provide 15 non-trash objects and get false-positives (models clas-

sifying non-trash components as trash items) to measure the accuracy rate. We set the number of items to be 15 because 15 is a doable and testable number to reflect the actual accuracy rate correctly, while 10 was a small number to be sampled from.

7.2 Tests for Movement

For movement, we evaluate the robot's ability to effectively collect objects and clean the given area without colliding into obstacles through three test cases: no trash components, 5 trash components, and 10 trash components. The robot will be placed in an assigned area (most likely to be a track) and start cleaning tasks. During the operation, we will test if it successfully avoids obstacles and record the number of obstacles it avoids. Also, we will keep track of the robot's operating time for movement efficiency.

There are two edge cases to be highlighted. First, the robot will be instructed to spin around 5 to 10 feet 2-3 times and stop if no object is detected. Also, if the robot is low on battery, it would find a way to return to the starting point or look for a charging station.

7.3 Tests for Integrations

There are three integration testing phases. In the first phase, we will test pick-up mechanisms with obstacle avoidance and record the number of trash items the robot collects. In the second phase, we test path planning and camera rotation with object classification and record the number of objects the robot is able to detect and reach to pick up the items. In the last phase, we are planning to combine the first and second integration phases and run multiple iterations with them. Just as before, we will record the time the robot takes to complete the cleaning tasks and the number of components it has picked up.

Each phase includes the following three tests to validate the robot's performance and improve its functionality as we go through to satisfy requirements for object classification, trash pick-up, and obstacle avoidance.

For the first test, we will use a 4ft by 10ft area containing 3 to 5 soda cans. Under this test setting, we aim to record and evaluate if the robot is able to identify soda cans above 95% of mAP for YOLOv7 and 80% for tiny YOLOv7 as well as 100% of recall rate as well as pick up the required amount of soda cans 90% of the time.

Then, our second test expands to an area of 4ft by 14ft and the mix of crumpled papers with existing soda cans to test the robot's performance with varied types of objects. During the evaluation, we will calculate the number of cans and papers it collects and see if the robot avoids obstacles more than 95% of the time.

The last test will include an area of 4ft by 18ft, which represents the small model of the actual streets, the addition of plastic water bottles, and the placement of large obstacles. This reflects a more realistic cluttered environment with object variations. Just like the previous tests, we will collect data for object classification, robot collision, and pickup rates.

8 PROJECT MANAGEMENT

8.1 Schedule

Please refer to Fig 5 for the full schedule.

8.2 Team Member Responsibilities

Our team member Ritu is in charge of training machine learning models and run inferences on YOLOv7 and tiny YOLOv7. Also, she plans to complete writing codes for the overall ML pipeline and testing the model using new pictures with the camera module.

Ella will be working on building the robot and testing the integrated parts. She is in charge of defining path finding algorithm and communication between Jetson Nano Orin and Raspberry, as well as building CAD models for pick up mechanisms.

Hirani is responsible for writing codes for pick up mechanisms and robot movements and using ROS to develop software for motion control. Also, she oversees creating connections between hardware and software components during the system integration.

8.3 Bill of Materials and Budget

Please refer to Table 1 for the comprehensive list of the materials.

8.4 Risk Mitigation Plans

Our current plan is to use existing datasets as much as possible to avoid time-consuming processes with custom datasets. However, there is a risk of low performance as some of the pre-existing models are low-quality. To mitigate this, we aim to evaluate whether it is acceptable to lower accuracy to prioritize faster detection or to focus on precision rather than speed by only using the standard YOLOv7.

Another major risk we could encounter is with regard to our pick-up mechanism. Currently, we are using a conveyor belt and a roller to push the trash items in and move them to the back of the robot. However, this mechanism has only been implemented under water, not on the ground, which we are testing. Subsequently, we might run into unforeseen problems that are hard to debug. Hence, we came to have a backup pick-up mechanism—a scoop and a roller—prepared so that once we cannot move forward with our current design, we can pivot to the backup in a timely manner.

For navigation, a simple path-finding algorithm will be used to reduce the overall complexity of the robot. Our main concern previously was how to move the robot when it encounters an obstacle on the way to pick up a trash item. To mitigate this, the algorithm will be kept simple

as follows: when an obstacle is detected along the movement, the robot pauses and rotates 90 degrees to move on to a new path. This helps maximize the probabilities of the robot working as intended by avoiding unexpected localization errors.

Also, there is a risk of identifying obstacles (humans or animals) that are smaller than the robot as trash items. We simply limit obstacles to being taller than the robot so that we do not accidentally classify them as trash objects and avoid those under the threshold height of the robot. Under this controlled testing, we can solely concentrate on classifying trash items vs. non-trash items without interference with the obstacles.

The final risk we identified is that no one on the team has experience with ROS. However, this will be mitigated by learning the ROS framework through online resources and seeking out help from the student organization RoboClub.

9 RELATED WORK

There have been several trash collecting robots that have been deployed/ prototyped in the past. Although the in-take mechanisms differed depending on whether the robot operated on land/ water, we found some similarities in the trash classification methods. One of the implementations of the water way cleaning robots, used a camera mounted on the chassis of the robot and image data to classify whether the object as garbage or non-garbage[12]. This specific design used ultrasonic sensors to detect the object near the robot, then used image data to classify. This makes sense for water-way cleaning robots, as there are no significant expectations for obstacles. Almost all objects are either classified as trash or non-trash.

Most of the on-land projects we have seen have not been deployed. A majority of them have been prototyped/ planned out, but not tested in the real world. They vary in terms of the intake mechanisms and the classification mechanisms (mostly used tiny yolo or lidar system). Our design will be testing a unique combination of water-way intake mechanism combined with a roller mechanism with the usage of tiny yolo for object classification.

10 SUMMARY

The Embellisher aims to promote public health and safety by detecting and collecting various trash items thrown out on the streets, such as soda cans, water bottles, and crumpled paper. Using object classification, The Embellisher will be able to differentiate garbage from non-trash items, pick them up without any human interference, and avoid obstacles. Through motion control, it will be able to navigate the given environments without hassle. Ultimately, the production of the robot with such functionalities will allow effective clean-up tasks and elevate the lifestyle of the public in the surrounding environments.

Going forward, our team expects to develop the robot's capability of garbage collection and refine its performance through interactive and progressive testing with variations of trash items. While it may be difficult to integrate the system and debug unpredictable challenges, we are confident that our risk mitigation plans will guide us through to resolve any issues. Overall, we hope The Embellisher is a promising prototype that raises social welfare by relieving pervasive environmental issues and benefiting the community as a whole.

Glossary of Acronyms

- CAD - Computer Aided Design
- mAP - Mean Average Precision
- RPi – Raspberry Pi

References

[1] *349 AMD NVIDIA Jetson AGX Xavier devkit vs. Kria KR260 Robotics Starter Kit.* https://www.tecnohub.org/2022/05/349-amd-nvidia-jetson-agx-xavier-devkit.html. visited on 2024-03-01. URL: https://www.tecnohub.org/2022/05/349-amd-nvidia-jetson-agx-xavier-devkit.html.

[2] Budget and Legislative Analyst's Office. *Policy Analysis Report.* https://sfbos.org/sites/default/files/BLA_Report_Street_Cleaning_Cost_Survey_062518.pdf. visited on 2024-03-01. 2018. URL: https://sfbos.org/sites/default/files/BLA_Report_Street_Cleaning_Cost_Survey_062518.pdf.

[3] Tausif Diwan, G. Anirudh, and Jitendra V. Tembhurne. *Object detection using yolo: Challenges, architectural successors, datasets and applications - multimedia tools and applications.* 2022. URL: https://link.springer.com/article/10.1007/s11042-022-13644-y#citeas.

[4] Emmet. *Raspberry Pi 5 vs Raspberry Pi 4 Model B.* https://pimylifeup.com/raspberry-pi-5-vs-4/. visited on 2024-03-01. 2023. URL: https://pimylifeup.com/raspberry-pi-5-vs-4/.

[5] Emmet. *RASPBERRY PI 5 VS RASPBERRY PI 4: THE DETAILED DIFFERENCES COMPARISONS.* https://www.elecrow.com/blog/raspberry-pi-5-vs-raspberry-pi-4-what-are-their-differences.html. visited on 2024-03-01. 2023. URL: https://www.elecrow.com/blog/raspberry-pi-5-vs-raspberry-pi-4-what-are-their-differences.html.

[6] *How Long Does A Robotic vacuum cleaner Battery Last*. https://www.ecovacs.com/au/blog/how-long-does-robotic-vacuum-cleaner-battery-last. visited on 2024-03-01. 2023. URL: https://www.ecovacs.com/au/blog/how-long-does-robotic-vacuum-cleaner-battery-last.

[7] *How long does the battery last or what is the cleaning time for Roomba?* https://homesupport.irobot.com/s/article/26647. visited on 2024-03-01. 2021. URL: https://homesupport.irobot.com/s/article/26647.

[8] Prabu Kumar. *NVIDIA Jetson Orin vs. other NVIDIA Jetson modules – a detailed look*. https://www.e-consystems.com/blog/camera/technology/nvidia-jetson-orin-vs-other-nvidia-jetson-modules-a-detailed-look/. visited on 2024-03-01. 2022. URL: https://www.e-consystems.com/blog/camera/technology/nvidia-jetson-orin-vs-other-nvidia-jetson-modules-a-detailed-look/.

[9] personal. *Soda Can Object Detection Dataset*. https://universe.roboflow.com/personal-g5mzf/soda-can-object-detection. Open Source Dataset. visited on 2024-03-01. 2023. URL: https://universe.roboflow.com/personal-g5mzf/soda-can-object-detection.

[10] Ros. *trash-plastic-bottle-detection Dataset*. https://universe.roboflow.com/ros/trash-plastic-bottle-detection. Open Source Dataset. visited on 2024-03-01. 2022. URL: https://universe.roboflow.com/ros/trash-plastic-bottle-detection.

[11] School. *Crumpled Paper Detection Dataset*. https://universe.roboflow.com/school-e5lvf/crumpled-paper-detection. Open Source Dataset. visited on 2024-03-01. 2023. URL: https://universe.roboflow.com/school-e5lvf/crumpled-paper-detection.

[12] Abhay Patel Shreya Phirke and Jash Jani. *Design of an autonomous water cleaning bot*. 2021. URL: https://www.sciencedirect.com/science/article/abs/pii/S2214785321028972.

[13] Aleksandra Szczepaniak. *Raspberry Pi or Arduino – when to choose which?* https://www.leorover.tech/post/raspberry-pi-or-arduino-when-to-choose-which. visited on 2024-03-01. 2023. URL: https://www.leorover.tech/post/raspberry-pi-or-arduino-when-to-choose-which.

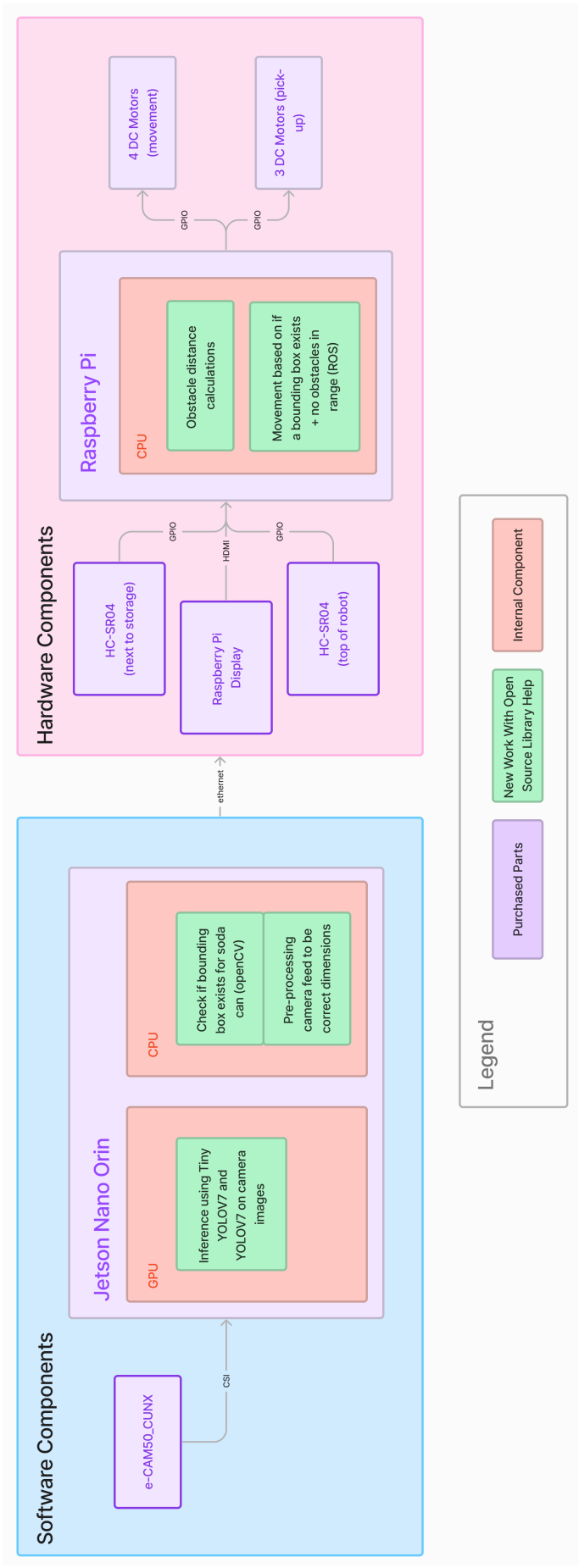

Figure 4: Block diagram of the entire system

Object Classification	Task Owner	Start Date	End Date	Status
Decide an object classficiation ML model + datasets (3hrs)	Ritu	2/5	2/12	Complete
Google Collab Setup + Running tiny-yolo (5hrs)	Ritu	2/12	2/14	Ongoing
Train model on collab (8hrs)	Ritu	2/14	2/25	Ongoing
Run inferencing on Jetson Nano Orin (5hrs)	Ritu	2/26	3/1	Ongoing
Write codes for overall ML pipeline on Jetson Nano Orin (12hrs)	Ritu	3/1	3/7	Not Started
Connect and set up the camera (2hrs)	Ella	2/12	2/19	Ongoing
Test the model using new pictures from the Raspberry Pi camera (4hrs)	All	3/11	3/17	Not Started
Motor Control				
Research ROS (1-2hrs)	Hirani	2/5	2/12	Complete
Set up Raspberry Pi (installing ROS) (3hrs)	Hirani	2/19	2/23	Ongoing
Define software requirements for motion and obstacle avoidance (3hrs)	Hirani/Ella	2/23	3/1	Ongoing
Learn & use ROS to develop software for motor control (5hrs)	Hirani	2/24	3/1	Ongoing
Write code and iteratively test the requirements (6-9hrs)	Hirani	2/26	3/1	Not Started
Set up actuators for the pick up mechanism (3-4hrs)	Ella	2/19	2/26	Not Started
Write code for pick up mechanism (3-4hrs)	Hirani	3/11	3/18	Not Started
Test motion, obstacle avoidance, and pick up mechanism using simple test cases (8-12hrs)	Hirani/Ella	2/26	3/11	Not Started
Integration/Testing				
Finalize on design sketch and dimensions of the robot structure (3-5hrs)	Hirani	2/5	2/12	Complete
Build CAD model (2hrs)	Ella	2/23	2/28	Complete
Research Path Planning	Ella	2/23	2/28	Complete
Sourcing Build Materials	All	2/26	3/1	Ongoing
Build/ cut out acryllic for base structure (2-3hrs)	Ella	3/11	3/14	Not Started
Assemble/wire wheels, Raspberry Pi, and actuators for pick up mechanism (4-6hrs)	Ella/ Hirani	3/11	3/14	Not Started
Test the integrated parts (5-8hrs)	Ella	3/14	3/18	Not Started
Research communications between Raspberry Pi and Jetson Nano Orin (1 hr)	Ella/ Hirani	2/19	2/26	Ongoing
Make connections between Raspberry Pi, motor control, and Jetson Nano Orin (3hrs)	Hirani	3/11	3/16	Not Started
Test motion control works well with the object classficiation using simple test cases (4hrs)	All	3/14	3/18	Not Started
Final Testing				
Test the system with one soda can (1-3hrs)	All	3/11	3/18	Not Started
Test object classification and pick up mechanism using 3-5 soda cans (4hrs)	All	3/18	3/25	Not Started
More soda cans (5-8) and variations (adding crumpled white paper)	All	3/18	3/25	Not Started
Use 5-8 soda cans, crumpled paper, empty water bottles, cardboard boxes	All	3/25	4/1	Not Started
Slack				
Object Classification				
Decide an object classficiation ML model + datasets (3hrs)	Ritu	2/5	2/12	Complete
Google Collab Setup + Running tiny-yolo (5hrs)	Ritu	2/12	2/14	Ongoing
Train model on collab (8hrs)	Ritu	2/14	2/25	Ongoing
Run inferencing on Jetson Nano Orin (5hrs)	Ritu	2/26	3/1	Ongoing
Write codes for overall ML pipeline on Jetson Nano Orin (12hrs)	Ritu	3/1	3/7	Not Started
Connect and set up the camera (2hrs)	Ella	2/12	2/19	Ongoing
Test the model using new pictures from the Raspberry Pi camera (4hrs)	All	3/11	3/17	Not Started
Motor Control				
Research ROS (1-2hrs)	Hirani	2/5	2/12	Complete

Spring Break

Figure 5: Gantt Chart

Table 1: Bill of materials

Description	Model Number	Manufacturer	Cost per Quantity	Quantity	Total Cost
Jetson Nano Orin	900-13767-0030-000	NVDIA	0	1	0
Raspberry PI 4	400	Raspberry PI	0	1	0
Micro SD Card	-	-	0	1	0
Micro SD Adapter	UNICASD01	UNI	9.99	1	9.99
Camera - e-CAM50_CUNX	e-CAM50_CUNX_H01R2	e-con Systems	0	1	0
Pin Converter	LA094	Onyehn	7.59	1	7.59
Ultrasonic Sensor (2pc)	HC-SR04	WWZMDiB	5.99	1	5.99
Display/ Monitor for Raspberry Pi	8595698868	ELECROW	49	1	49
Displayport to VGA converter	DP2VGA-A-B-V2	Moread	7.99	1	7.99
Displayport to HDMI converter	WARRKY-AV-DH01	Warrky	8.38	1	8.38
USB-A to USB-C cables	TC-UAC2015R	etguuds	5.98	1	5.98
USB-C to USB-C cables	-	-	0	2	0
Battery for Jetson Nano Orin & Raspberry Pi	B02P	CONXWAN	24.29	1	24.29
Battery for motors	-	-	0	7	0
Mecanum Wheels (4pc)	B0C1B68993	Yahboom	29.99	1	29.99
DC Motors	-	-	0	4	0
6” x 18” Roller	-	-	0	1	0
4 small PVC pipes (1.5” diameter & 18” wide)	-	-	0	4	0
DC Motors	-	-	0	3	0
Material for roller (12” x 18”) and (18” x 18”)	-	-	0	0	0
L-shaped Interior Corner Connectors	Tongling-JC-20	BLCCLOY	0.682	20	13.64
Acrylic boards (dim)	-	-	0	12	0
2020 Series Alum. Extrusions	-	-	0	4	0
Connectors for motor	2H1020	IXGNIJ	1.799	10	17.99
Adafruit DC & Stepper Motor Hat	2348	adafruit	22.5	2	45
Total					225.83