Robotic Trash Concierge

George Gao, Jack Girel-Mats, Zachary Mason

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—The modern workplace has evolved such that many office workers collaborate in the same area. However, this has resulted in many small trash bins scattered across the office. These bins regularly overflow, forcing janitors to make repeated trips to hundreds of tiny trash-bins scattered across an office space. This is not only inconvenient and unhygienic for office workers, but it's a point of great inefficiency for janitorial staff to check hundreds of possibly empty bins. We have devised a solution using an autonomous robot capable of taking out an office worker's trash can by bringing it to a centralized dumping ground, where janitorial staff will empty dirty bins and place clean bins back into their proper locations. This method streamlines the trash collection process and eliminates the need for janitors to check hundreds of tiny bins across the office space.**

***Index Terms*—Autonomous, LiDAR, Robot, Roomba, ROS, SLAM, Trash Bin**

I. Introduction

The work environment has continuously evolved from era to era, and along with it problems and solutions. The modern workplace is no exception. Offices have become increasingly open, and collaborative, with tech campuses at the forefront of such changes. They have introduced architectures with open seating and shared desk space. It however introduces a copious amount of trash bins into the office, one for each desk. This places an excess burden on janitorial staff to accomplish the dull, monotonous work to take out all the scattered bins.

We've noticed that the scale of the problem and its monotonous nature, this was an appropriate application of automation. Thus we have designed a solution as such.

Our proposed solution introduces a robot designed for the flexible nature of open office spaces, with a goal of greatly reducing the workload on custodial staff. Our robot will be able to map out an open office area, and track where trash bins are located. Using a tracking system, the robot will be able to navigate to the trash bins, and bring them to a centralized dumping ground to be taken out by janitorial staff. Janitorial staff will then return clean trash bins to their original locations. This reduces the running around custodians have to do as now the trash is centralized in a common area.

While there are no competing products on the market, we realize that some technologies can be retrofitted to solve the same problem, notably other robotic delivery systems such as Proteus Robotics warehouse robots [6]. However, each such solution was not designed for an office space specifically, and would be clunky to use.

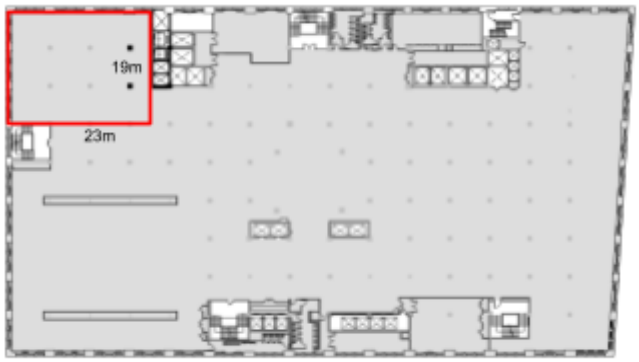

Fig. 1. Floorplan of Meta's 770 Broadway office in NYC [9].

II. Use-Case Requirements

We first introduce unit metrics that our project must meet. We extrapolate that the average worker will be out of the office work area by 6 p.m., so we chose a one hour buffer for the starting time of our robot, 7 p.m. We also take note that night custodians should still be in the office by 12 a.m and thus our robot must be finished by then, giving us a working time of five hours. Within these five hours a robot must travel at a speed of 0.21m/s. This comes from our use-case study into the dimensions of the average working area of a Meta office space (Fig. 1). The room we looked at came from Meta's 770 Broadway NY office, which had a working area of 19m x 23m, and assuming a common starting ground to the middle of the room (as the average distance for all trash cans) we calculated a round distance time of 42m. For a working area with 90 trash cans for 5 hours; we reached our 0.21m/s average speed after adding some buffer time.

The next use-case requirement we want to meet is to have an 85% trip success rate, with a successful trip defined as successfully navigating to a trash can and bringing it back without spillage or human collisions. The 85% comes from two sub requirements in series: a 90% bin docking success rate, 95% non-collision rate, and a 99% no spillage during transport. We think 90% is viable for bin docking due to the numerous ways a bin could be moved or mishandled by humans in between cleaning periods. A 95% non human collision rate is also acceptable due to the work period of the robot being designed for an after hours setting, alongside its menial movement speed of 0.21m/s, meaning collisions are both unlikely to happen and inconsequential if they were to happen. Nonetheless we don't want to inconvenience workers, thus we decided 95% would be acceptable in this case. For a given bin collection trip, there

should be at least a 99% rate of transport without the bin tipping over. We deem this to be acceptable since spilled bins nullify our system, and our work area of 90 trash cans means 1 bin could tip over, in which case it wouldn't be too burdensome for a custodial staff member to take care of manually, since they may already be performing a separate task nearby. Considering these success rates in series, we follow the derivation

$$R_s = R_{bin\ docking} \times R_{non-collision} \times R_{bin\ spillage} \quad (1)$$

to reach the final success rate of being 85%. We deem 85% success rate to be acceptable due to the still drastic reduction in labor that it entails, cutting the number of trips to be a trivial amount. Furthermore, due to how 85% encompasses human collisions, the number of trips that succeed in purely bringing back trash cans would be much higher. Following the same equation as 1, except getting rid of the non-collision rate, we achieve a success rate of 89%. In terms of our use case study this means around 10 left-over bins, a drastic reduction from 90.

Another requirement for our system is it should be able to lift a 10 pound, 7-gallon standard trash bin. Given the office environment where bins are expected to be emptied nightly, bins won't fill up too much throughout the day, as it's mostly going to be food waste and packaging that gets thrown out, along with the occasional document, none of which are that heavy. Bins themselves weigh 2-3 pounds, so this allows for 7–8 pounds of waste to be removed at a time.

We also want to be mindful of the ethical, health, and safety implications of our project. Our robot is going to be working alongside other humans in an office environment, so it's important to keep the safety of the office staff central to our project's goals. This includes adding obstacle detection, emergency stop procedures, and making the robot easy to see. Additionally, we want our robot to help janitorial staff not replace them. We understand that as robotics become more and more advanced, some jobs will eventually be replaced, but we hope that our design lives symbiotically with janitors, as they will be responsible for managing the autonomous system, as well as maintaining its operations.

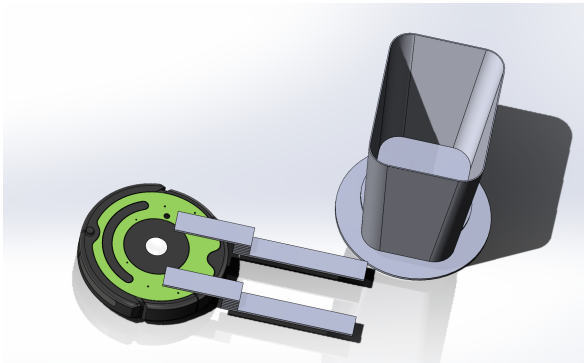

Fig. 2. Basic CAD model showcasing bin pickup and drive systems.

III. Architecture and/or Principle of Operation

The high level architecture of our solution can be found on Fig. 9 at the end of this document.. Now we will describe each sub-system in detail.

A. Bin Pickup System

The pickup apparatus will consist of an arm powered by stepper motors to create a stable and reliable system to minimize the janitorial staff's workload overhead.

B. Mapping

In order to create an autonomous system, we need to have knowledge of the office space that the robot will be operating inside. To accomplish this we will use SLAM to precreate a map of the office environment. This map will contain elements such as desks, chairs, walls, and bin locations.

C. Bin Tracking

In order to keep track of and locate the bins in an office environment, each bin will be given a unique tag. During mapping, every bin's location will be recorded into a database. This database will then be queried during pickup for bins' status and coordinates.

D. LiDAR System

Our robot needs sensing capabilities to navigate flexible office environments. These environments may have obstacles and varying layouts. The LiDAR sensor will provide the robot with a representation of its surroundings, and be used for mapping and obstacle avoidance.

E. Path Planning System

Once our robot has created a map of its environment, and enters the pickup routine, it will need to plan a path from its location to its desired bin pickup location. The path planning system will be responsible for creating a path from the map of the office environment. This system will also implement a local path planning strategy to accommodate new obstacles.

F. Drive System

For the robot, we will be using an iRobot Create2 (Roomba). This will allow for a solid base on which we can mount various components, as well as being supported by interfaces to send custom movement commands.

G. Camera System

The roomba will be outfitted with two cameras, both oriented to be facing opposite directions, 180 degrees away from each other. The front camera will be responsible for identifying bins during pickup. As our robot's pickup arm is mounted onto the rear of its body, our second camera is mounted on the rear. The purpose of the back camera will be to gather information for proper alignment of the robot to the bins before the arm is activated for pickup.

H. Computer Vision System

Our camera feeds will be piped into OpenCV to accomplish bin identification. We will calibrate our camera to be able to calculate the distance and orientation of bins from our robot. These will be used during mapping for bin location acquisition, and during pickup for bin alignment.

The LiDAR feed will be used for mapping of the office environment, as well as obstacle avoidance during the bin pickup routine.

I. State Coordinator

Our robot has many tasks: mapping, path planning, bin identification, bin alignment, bin pickup, and so on. The state coordinator will have the final say on signals sent to the drive system and bin pickup system. It will also be responsible for identifying situations in which janitorial staff are needed, such as in the case of the roomba getting stuck.

IV. Design Requirements

Our project's design requirements are derived from our use case requirements, and specify the performance of various subsystems that will meet our use case requirements after system integration.

To design a robot that fits within the constraints of open office environments, we used the Meta NYC office to estimate the size restrictions for our robot. Based on our findings the robot's height needs to be less than 30 inches and it needs to be less than three feet wide and and 3 feet long to fit between and under desks.

During the robot's pickup routine, it may encounter obstacles and humans, which could result in trash spills and collisions. We determined that a system for obstacle detection and avoidance was required. Additionally, it is important for the robot to be easily identifiable by office staff.

Given the use cases' movement speed requirement and cleanup time period, we need to ensure the robot has enough power to complete its tasks. The Roomba has an advertised runtime of 2 hours, but this is with the cleaning motor active, which uses a significant amount of power. Since we're only using the drive motors, the overall power use is significantly reduced and should allow us to meet the 5 hour requirement. However, the robot's computation system, bin pickup system, and sensing systems will require an additional battery attached to the Roomba's body.

Our camera system should be able to process data with at least 15fps (frames per second), as this is recommended for applications that move at less than 2m/s for proper CV applications [1].

For accurate bin identification and environment tag usage, we specify that the computer vision subsystems must be able to correctly identify with 90% accuracy from greater than 2 meters away in distance. 2 meters away would mainly be used for sanity checking the location that the robot is moving towards is correct, and is not that important. Thus lower accuracy is acceptable. The accuracy must reach 95% within 1 meter away but greater than 0.15 meters away from the camera, barring any obstructions when in proper office lighting conditions, as this would be the distances the docking procedure would be working within. We want to be as accurate as possible due to the inconvenient nature of fixing such mistakes of moving the wrong bin. However, due to the video stream-like nature of our application, both 90% and 95% accuracy over the course of 15fps means it's virtually impossible over the course of a docking procedure to continuously mistake the tag. Furthermore, the vision system must be able to do accurate distance sensing within 1 meter away with an accuracy of 90% with accuracy in this case defined by how far off the camera position is to the labeled tag with the following equation:

$$1 - (d_r - d_s) / d_r \quad (2)$$

with d_r being the real distance and d_s being the sensed distance. We apply a 90% accuracy to be sufficient due to the relative nature of docking to a bin, which would be the main use case of distance sensing. Due to how close bins are and low movement speeds, movement can be compensated to be safely within distance bounds to properly pick up bins with the given location accuracy.

In order to lift the 10 lb bin vertically, a great enough force needs to be generated, and the bin needs to be lifted high enough off the ground to prevent any catches on the ground. Any floor imperfections are assumed to be relatively small (<1in). The bin weighs approximately 44 N, and should be lifted up to 1.5in (~4cm) total, allowing for 0.5in of vertical misalignment during the docking process.

To properly traverse the office space and reach bin locations, our mapping of the area should be qualitatively accurate. Then during pickup, we need to have a localization accuracy of at most .5m in order to not bump into mapped obstacles and end up in the desired destination.

V. Design Trade Studies

A. Trash Bin Modifications

In order to support moving a 10 pound trash can in the office environment, we considered many bin modifications, including the addition of wheels, a fixed pedestal, bidirectional latching mechanism, omnidirectional base plate, or side-mounted flange.

Because our Roomba may not have perfect positioning and drive accuracy, any bin that requires fine positioning would make it difficult to meet the 90% docking success rate. This meant a bidirectional latch system would not be feasible since it requires alignment with the bin at a specific angle. Similarly, we decided against using wheels on the trash bin itself since any physical contact by the Roomba could shift the bin's position, thereby using up valuable time by having to re-adjust, and perhaps even causing a

docking failure if the guidance algorithm was slightly off.

Another idea was to use a pedestal that we could place the trash bin on top of, where the Roomba could drive underneath and lift the bin for pickup. The issues with this solution are twofold. One is that a trash bin elevated off the floor restricts the position it can be in for a pickup, so employees would not be able to move a trash bin around unless they also moved the pedestal. The other is that elevating a trash can off the ground by the height of the Roomba (approx. 4in after attachments) immediately adds 25% to the bin height, which increases the risk of tipovers that we are trying to avoid. It also complicates the design unnecessarily since the cameras and LiDAR would have to be mounted low to the ground where they have less visibility. This leaves the side-mounted flange and omnidirectional base as the last two bin attachments to consider. Both these designs would offer greater variability in pickup positioning, but the side-mounted flange would require a more complicated lift system as explored in part B below. This resulted in our selection of the omnidirectional base plate shown in Fig. 3 as the best option that would allow us to meet use case requirements.

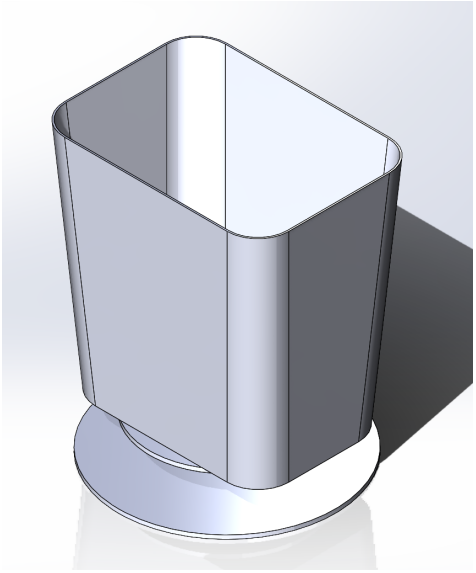

Fig. 3. CAD model showing the omnidirectional base selected for the bin modification.

B. Docking Arms

Since the Roomba is a circular object with no existing attachment points to mount a lift system on, we had to design our own. In part A above, we explored the different bin modifications and arrived at two potential solutions. Because 7-gallon trash cans are rectangular, the apparent width varies based on the position around it, so our arms would have to have to similarly change width if using a side-mounted flange. This would require actuation, which reduces the overall strength and could jeopardize our 10 pound load requirement, in addition to increasing complexity. For this reason, we chose to use fixed arms in conjunction with an omnidirectional base.

C. Bin Lift System (BLS)

Since we decided against the use of wheels, the bin must be physically lifted off the ground for transport. There was really only one practical option here, which was to use stepper motors, since they offer high torque and fine control. To generate lift motion, we considered using either a cam that would spin a partial rotation or a worm gear to produce linear motion. We decided to use a cam mounted to the stepper motor through gears (see Fig. 8) since it allows for more customization of torque and lifting profiles, as well as being lower-cost and more easily manufactured.

D. Onboard Processor

We explored performing offboard and onboard data processing, with a combination of microcontrollers and PC's, including the RPi and Jetson Xavier AGX. The vision and navigation subsystems for our robot require a significant amount of data to be processed, which may not happen at the 15fps laid out in our design requirements on the RPi, leaving the Jetson or a PC as the only options to perform computation. A PC would have to be wirelessly networked into a microcontroller onboard, which increases the chance of failure and reduces autonomy since the Roomba would be reliant on data processing and communication from multiple nodes. Because the Jetson already has GPIO and plenty of computational power, it was the logical choice to meet our design requirements.

E. Onboard Vision System

To actually sense our surroundings, we had to use some form of camera, LiDAR, or both. LiDAR has demonstrated success in room mapping, especially with acquiring depth information, but cannot provide the fine positioning data required for navigation to a bin for successful docking. As such, we decided to use both a LiDAR and camera. These will be mounted on top of the Roomba to have the best vantage point possible.

We could have gone with a single camera that faces the docking mechanism, but this means its view is obstructed (see Fig. 4) half the time (anytime we tow a bin), and cannot be used for positioning. Because the bin drop-off point needs to be at a certain location, we decided to use two cameras facing in opposite directions that allows for visual navigation all the time.

As for the LiDAR, there were two models in the ECE inventory that we could have used, but to avoid a similar issue as the camera where a carried bin causes a significant obstruction, we decided to use the 360-degree version due to better handle obstructions.

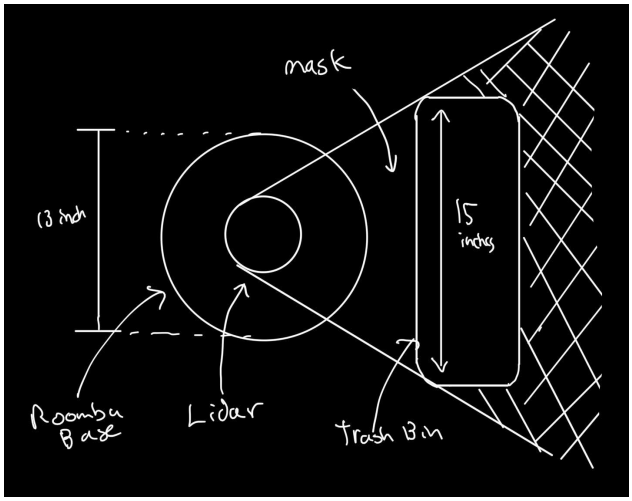

Fig. 4. Figure showing masking requirements on LiDAR due to bin obstruction during transport.

F. Environment Mapping

The Environment mapping function went through several iterations. Initial research showed built-in MatLab libraries for LiDAR mapping to be sufficient for our application. We had planned to export the MatLab library into C and port it onto the onboard processor. However, after further research and testing, the library seemed to be more for stationary applications instead of live exploration. While this application could work due to the slow nature of our robot, we decided to transition to ROS open source navigation libraries. There were two libraries that we considered, hector_slam[5], and gmapping_slam[3]. Both were viable, but we decided on using gmapping_slam because the hector_navigation library is not supported on the version of ROS (Noetic) we are using[4].

G. Robot Path Planning

Like the environment mapping system, MatLab was the first choice for path planning as it supported built-in libraries that can navigate based on the given output of the mapping library. However, due to swapping the mapping system to ROS, we decided that transitioning the map input to one that would work with the MatLab library would be inefficient and impractical. Thus we have decided to stick to a unified system for path planning and mapping. Keeping libraries within the same system decreases development time and has more supported features due to increased synergy and compatibility. We decided on using the ROS navigation stack as it is fully supported by our ROS release unlike hector_navigation [4].

H. Bin Tracking

To track the location and status of each bin, we will be using a database in conjunction with ARuco tags. There were many options for tagging the specific bins. We first considered using April Tags due to their inherent uses in LiDAR applications, however due to the ease of use and compatibility of ArUco tags with the openCV library (which is the main onboard camera vision computation system), we decided to use them instead. We also considered different options of the stateful tracking of bins themselves. We needed some system to keep state in between runs, as such, we considered some system that writes to an external hard drive through some sort of file I/O interface, or using a database. File I/O would've involved writing our own file system interface and developing our own encoding / decoding scheme or using a higher level abstraction combined with some python library such as pickling. This would've provided much stronger flexibility in handling bin storage for future applications but we decided that such flexibility was a low priority for the current project. Thus we decided on a more inflexible state tracking system involving a database, and building an abstraction interface library on top of that. The database streamlines fetching and writing concurrently and has sufficient features for the current goals of the project. Furthermore, there are many built in interfaces in python to interact with a database of choice.

VI. System Implementation

A. Trash Bin Modifications

The trash bin itself will be omnidirectional as discussed previously, and necessitates a circular interface. To allow for variations in robot positioning while maintaining stability, we will be using a large disk with a slightly smaller diameter than the length of the trash bin, affixed to a larger circular base plate via a central column that also provides room for the lift arms to slide underneath as shown in Fig. 5. All of these components will be made with laser-cut ¼" plywood, and joined with a combination of glue and metal hardware. The upper and lower disks will be 10 inches and 14 inches in diameter, respectively. Because the lower disk has a larger footprint than an original bin, we actually increase stationary stability further.

Fig. 5. Side view of the modified trash bin showing the bin, upper disk, central column, and lower disk (from top to bottom).

B. *Docking Arms*

Our docking arms look like forklift arms, but will actually be static. They will be a box-like structure to provide rigidity, while allowing for components to be mounted inside, namely the stepper motors, gears, and cams for the BLS. The arms will also have small omnidirectional wheels at the end, since a bin extended out on the arms will not be able to stay upright independently. These wheels will roll all the way across the base plate on every docking sequence.

The arms will be fixed to the top of the Roomba via existing screw holes, and have drop-down members on the front to fit under the upper bin disk, while simultaneously reducing the risk of a tip-over.

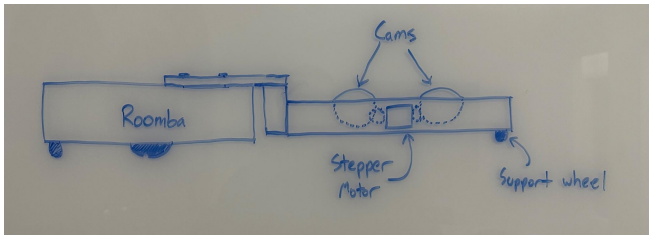

Fig. 6. Cross-sectional diagram showing the docking arm and components housed inside.

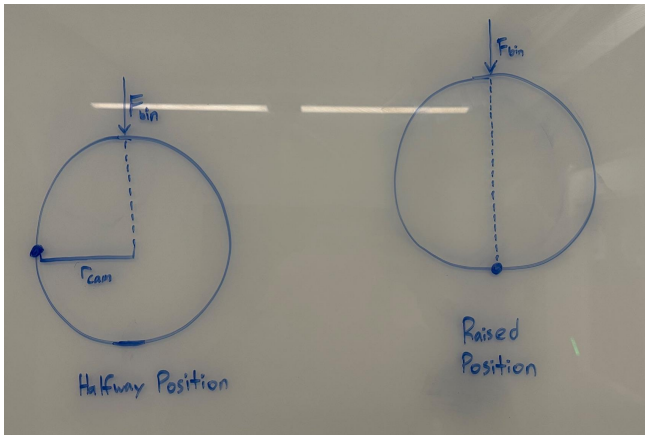

Fig. 7. Diagram showing cam rotation in minimum (raised) and maximum (halfway) torque positions.

C. *Bin Lift System (BLS)*

The lift system uses stepper motors, gears, and cams to achieve our 44 N lift capacity and 4 cm lift height. We are using 2 NEMA 17HS4023 stepper motors, which have a reported torque output of 13 N*cm, while being in a small form factor that will be able to fit in our docking arms. The required torque to lift the bins is given by the following equation:

$$\tau = F \times r \tag{3}$$

where F is the force perpendicular to a radial from the axis of rotation, and r is the radius at which F is applied. Based on Fig. 7, we can compute the required torque output of the lift system per motor, which is:

$$\tau_{out} = \frac{1}{2}(F_{bin} \times r_{cam}) \tag{4}$$

$$\tau_{out} = \frac{1}{2}(44\ N \times 2cm) \tag{6}$$

$$\tau_{out} = 44\ Ncm \tag{7}$$

The gear equation given below will determine what input-output tooth ratio is required to meet the torque requirement. This is because a low gear ratio (<1) will provide an increase in output torque

$$Ratio = \frac{N_{in}}{N_{out}} = \frac{\tau_{in}}{\tau_{out}} \tag{8}$$

Using this equation, we can calculate the required ratio:

$$Ratio = \frac{N_{in}}{N_{out}} = \frac{13\ Ncm}{44\ Ncm} = \frac{0.3}{1} = 3.38 \tag{9}$$

This ratio means for every 3.38 turns of the stepper motor, our output cam should rotate once. To add a buffer that allows for component friction, we select a gear ratio of 1:4. The cam will be circular and 4 cm in diameter, but mounted off-center with its edge on the center of rotation of the output gear as shown in Fig. 8. For each motor, we will have 2 output gears, each with a cam, allowing for a total of 4 contact points to lift from. This ensures stability of the bin and will prevent tipovers from occurring during the pickup process. The gears and cams will be 3D printed since it allows for a high degree of customization, and the infill can be adjusted to balance cost and strength. This also allows us to manufacture the cam and output gear as a single piece which makes integration far easier.

One final part of the BLS is going to be a resting bar for the cam just beyond the vertical position. This will allow for static stability without having to constantly power the stepper motors, helping us to meet our 5 hour runtime requirement.

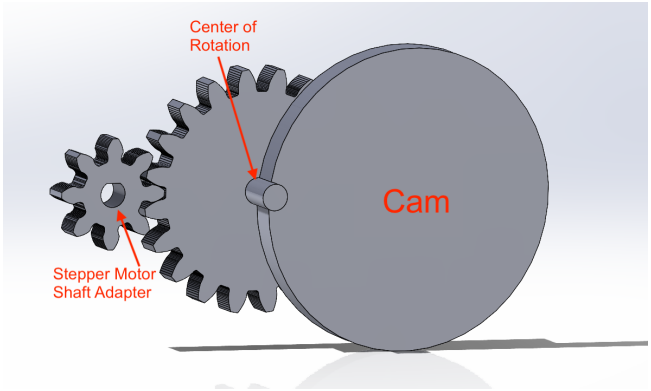

Fig. 8. CAD model of single gear system with cam shown.

D. *Drive System*

Our drive system is built on top of the iRobot Create2. This robot is a Roomba modified with a serial connection for manual control. Using an included USB to serial cable, we will control the Roomba's movements from our NVIDIA Xavier AGX using the pyCreate2 library [7]

which supports tethered driving [12]. To coordinate with other ROS software, the off the shelf ROS module diff_drive_controller[2] will be used, as it implements a differential drive controller as necessitated by the design of the Roomba. Additionally, an emergency stop system will be added to the drive system to end all movement if necessary.

E. Onboard Vision System

The vision system uses Python's openCV library to handle bin identification, location as well as rudimentary distance sensing. The vision system greatly depends on the camera used, and thus should be calibrated on the software side to handle distortion effects [13]. The camera is used to identify ArUco tags in the environment and communicate ArUco coordinates in relation to the robot to the central logic hub. ArUco tags also provide unique identification numbers which can be read from with openCV. This system also uses the ArUco tags to differentiate between the different possible orientations of a trash-bin in relation with the robot itself (by assigning a different ArUco id modulo the number of orientations).

Our LiDAR system is implemented using the Slamtec RPLIDAR A1M8. The LiDAR will be used for mapping, localization, and obstacle detection, and be connected directly to the Jetson Xavier AGX using an included usb adapter. Due to the height restriction of our robot, which is necessary to drive under desks, the mounting position of the LiDAR will bring the trash bin in alignment with the LiDAR sensor. This creates an obstruction that the LiDAR will interpret as a wall, messing up localization. To avoid this issue, a mask will be applied to the LiDAR's output. This decreases its FOV, but this is a reasonable tradeoff as it makes localization and implementation much more straightforward.

F. Environmental Mapping

We will be using SLAM to create a 2D map of the robot's office environment. Specifically, we will be using the ROS module gmapping_slam which constructs a map from 2D laser scans provided by the Slamtec RPLIDAR A1M8 and odometry data provided by the Roomba. In order to create the initial map, a one time setup is completed, which requires a manual exploration of the office space by remote controlling the roomba until a satisfactory map is created.

G. Robot Path Planning

Much like our mapping system, we will be using an off the shelf ROS stack called "navigation"[8]. The navigation stack is a conglomeration of ROS modules each that pertain to a specific task. Critically, it includes the acml package, which is responsible for localizing the robot, the global_planner package which creates the path from robot to bin locations, and the basic_local_planner package which handles following the overall path while taking into account the environment and obstacles. The navigation system's output will then be routed into the state coordinator system for control of the robot.

H. State Coordinator

To coordinate all of the tasks that the robot needs to accomplish, a central decision maker is necessary. To keep our software stack unified, this system will be implemented as a ROS node. It will take in movement directives from the manual control system, bin alignment system, and bin pickup system, then in combination with the state of the system, output the required movement commands to our drive and bin pickup systems.

I. Bin Tracking

Tracking bins is a stateful protocol, and must be maintained between sessions. It also must be scalable and have concurrency control for future scaling priorities. As such, a light-weight database with standard library python support is perfect for our use case. Thus, we chose to go with SQLITE, which is part of the standard python library. The database tracks bins locations in the mapping environment, whether the trash needs to be taken out, as well as each bins's identification number. The database interfaces with the backend logic system on the onboard processor through a custom python interface that handles bin tracking logic, including tracking bin locations on drop-off, pick-up, and spillage of bins, as well as successfully emptied bins.

J. Electronics Power System

As for our power system, the primary concern is delivering enough current at the appropriate voltages to keep all components active, while avoiding brownouts or current-limiting conditions. See the table below for the power requirements of our various components. Note that the currents are maximum ratings, and are likely less under normal operating conditions.

TABLE I. SYSTEM POWER REQUIREMENTS

Component	*Voltage (V)*	*Peak Current (mA)*
Nvidia Jetson AGX (10W mode)	12	~850mA
USB Camera (x2)	5.0 (USB)	70
360-degree LiDAR	5.0 (USB)	100
TB6600 Drivers (x2)	3.3, 12	1500

With this analysis, we can determine that the peak current needed is 4250mA, which is the sum of the Jetson and stepper drivers, but not including the USB devices. This is because the 10W Jetson power mode includes the running of USB peripherals, which itself has a rated max output current of 500mA. The TB6600 drivers use 3.3V on the pulse, direction, and enable lines, which is perfect since the Jetson has 3.3V GPIO pins. With these requirements, we believe a battery >20Ah of capacity and a 12V DC output will more than meet enough to meet our needs for a 5-hour

runtime. This is especially true because the steppers only run for a brief period of time when the BLS is active, and are otherwise in an idle mode with very low current consumption.

VII. Test, Verification and Validation

A. Trash Bin Modifications

The 30 inch height requirement will be measured from the floor to the uppermost point of the robot. A constraint of 3 feet has also been placed on the width and length of the robot. By meeting these standards we ensure the robot can fit between the aisles ensuring the robot can physically get to all bins.

B. Battery Life

The robot must last for the average working session. Currently, its intended use involves finishing the task within a five-hour time frame. The primary test to confirm this will drive the robot while carrying a full bin for 2.5 hours followed by movement until the robot runs out of battery. This mimics the full 5 hour use time.

C. Vision System Accuracy

To test the minimum requirements of the vision system we will set up the standalone vision system implementations and place tags within various distances. We will place an ArUco tag at distances of 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0 meter away from the camera lens, and do 10 trials each, to target an accuracy of 99%. Further testing will be done in the same method from 1.1m - 2m (in 0.1m increments) with 2 trials each to target a 95% accuracy. Such accuracy will allow us to make sure the use case requirement of proper bin identification is met. We will also test at each of the listed distances to ensure that distance accuracy is also met. We will use equation (1) to calculate the average accuracy amongst all trials, to ensure we hit the 90% accuracy laid out.

D. Movement System

The move system in place must be able to hit the minimum of 0.21 m/s when traveling in straight lines. To make sure that our implementation hits this baseline, we shall take videos of the robot moving forward across a distance of 3 meters. We would then confirm if it can be completed in 13 seconds (this includes slack such that our robot must be moving on average faster than 0.21m/s).

E. Bin Pickup

The pickup system must be able to lift a trash bin that is 10 lbs at maximum. We will test over the course of 10 trials whether our system can lift a trash bin weighing 10 lbs 1.5 inches off the ground without spillage. This will be used to validate the use case of bin docking without spillage.

F. Mapping System

We will create and analyze a map of the HH1307 laboratory space over three trial runs to qualitatively measure the accuracy of the resulting map. This map should clearly portray the rooms walls and other large obstructions such as desks.

G. Localization System

After creating a map of HH1307, we will place the robot in 10 different locations, manually move the robot for 30 seconds to warm up localization accuracy, and then measure the accuracy of the perceived position vs. the actual position of the robot.

H. Path Planning System

To test the path planning system, two points in HH1307 will be picked 15 ft apart, one for the robot's starting position and the other for the bin location. Then, the robot will create a round trip path to the destination and back, which will be followed by the robot. We will measure the success rate, (getting <.5m to the trash bin location where cameras take over, no collisions), over 5 trials. Then we will pick two new points, and rerun the test.

Next we look towards verifying our use-case requirements are met, through a series of integration tests between our subsystems.

A. Bin Docking

Using the vision system that has strong accuracy in bin identification and distance sensing within a 1 meter distance metric, we want our implementation to successfully dock with a full 10 lb trash bin. The robot must be able to attach the docking mechanism successfully and lift up the bin according to the design specifications outlined, hence the full 10 lb trash bin used. We repeat the test from various orientations of the bin for a total of 20 trials. A successful dock involves the robot being able to identify the correct bin being picked up, as well as the lift up of the bin to 1.5 inches off the ground without spillage.

B. Non-Collision

We require that during a trip between two destinations 10ft directly apart from each, no collisions occur 95% of the time. We will trial 20 trips between two points with a walking human in each test, in which we want the robot to stop before hitting a human if the human crosses its path.

C. Full Integration Testing

We require a successful trip to happen 85% of the time, and using all the subsystems that have met our specifications, we will run a full integration test 20 times to ensure a 85% success rate is hit. Bin Docking and collision rates must be met such that a successful trip is defined as no collision happening with a trash bin reaching the final destination without spillage. This includes proper path planning to be within <0.5 meters of the bin that is targeted for pickup, successful docking, and bring back. Single trips

will be trialed individually with a single bin to bring back, from various locations and orientations.

Finally, one full integration test will be done on rotating bins for a total of 12 bins in the HH1307 room. Rotating bins means after a bin has been taken to the dumping ground, we manually replace the tags and bring it to another location that will be targeted to emulate a room full of bins. The robot must finish getting to all of the bins within a time span of 20 minutes. This time span is extrapolated from the ratio of bins that must be completed in the META NYC use case study of 90 bins over 5 hours with a round trip time of 42 meters, compared to the smaller room of HH1307 and 12 bins and an average round trip distance of 18 meters. Using this ratio we solve that to roughly accomplish essentially the same task as our use-case study in the HH1307 room, the robot must finish the task in 17.4 minutes. We round to 20 to account for bin rotation and thus having a human continuously moving in the environment causing unnecessary stoppages.

VIII. Project Management

A. Schedule

Our schedule will follow the Gannt chart showcased by Table III. It is color coded based on the team member that is primarily responsible for the task, with color combination involving shared tasks. Tasks are usually done in one week periods with a steady progression of subsystem bring up prior to March 3rd 2023, then integration following March 11th 2023. Various amounts of slack have been built into the schedule to account for unforeseen circumstances, and the color coding for responsibilities acts as a guideline for bring-up. This means multiple and different members may step in to help accomplish tasks in the scheduled weeks even if not required by the chart specifically.

B. Team Member Responsibilities

Team members' responsibilities are split by team member backgrounds.

Mason has a strong hardware background, giving him default responsibilities on hardware-related tasks. This includes 3-D modeling, designing, and building the docking mechanism. His embedded background also means he has partial responsibility in embedded communication between subsystems of the robot.

George and Jack both come from software-oriented backgrounds, relegating most software tasks split amongst them. While Jack has the primary responsibility of LiDAR bring up, and George is responsible for the bin tracking backend with a database; more complicated software systems such as path planning, CV, and scheduling logic have shared responsibility between them.

Integration is the partial responsibility of all team members due the embedded communication and software and hardware nature of the project.

While individual team members have primary responsibilities, each is responsible for the success of the project as a whole. This means that we each are responsible for helping in the case that challenges arise for tasks.

C. Bill of Materials and Budget

Please refer to Table II at the end of this document for the bill of materials and their associated costs.

D. Risk Mitigation Plans

Our robot will have numerous systems, any of which could fail. It is vital that we have backup systems and implementation plans in place in the event that this occurs.

During the initial stages of our project, we ran into connectivity issues with the NVIDIA Xavier AGX. The Jetson would sometimes disconnect from our secure shell session making it difficult to develop. These issues were resolved by reflashing the AGX and switching to an ethernet cable instead of a usb cable for SSH. We have many software components to tackle, so the developer workflow is vital to the success of our project. If we were to run into an unrecoverable situation with the AGX, our contingency plan is to utilize the Xavier NX which prior teams have found less problematic.

We also developed a backup plan for the LiDAR system. We are currently using the Slamtec LiDAR A1M8, which accurately identifies solid objects such as desks, frames and walls, but may not correctly identify other objects like desk chairs. Therefore, we have also acquired the Intel RealSense LiDAR Depth Camera L515 as a backup. In case our original LiDAR does not provide accurate obstacle detection, our implementation will pivot to the L515.

Another risk that we have addressed is the Roomba's power supply. The Roomba itself is powered by an internal battery pack. This battery pack is old, and may have lost performance. To meet the use case requirement of running the Roomba for 5 hours, we have located aftermarket battery replacements that provide additional range.

Finally, to address the risks in the SLAM and navigation stacks, we analyzed past projects that have implemented robot localization and path planning using overhead camera systems. We believe that this technology could be adapted to our problem space. Additionally, our team has previous experience with this type of system.

IX. Related Work

Autonomous robots that are used to move items around are not unique, especially as robotics rapidly develops. Thus, there are quite a few predecessors to our implementation albeit designed for different use-case scenarios and purposes.

Perhaps the first implementation when we think of autonomous item movement is Amazon warehouse robots. Amazon, the world's leading online retailer, deploys over 520,000 drive units of its proprietary robot named Proteus. They are warehouse robots designed with navigation and perception features to move GoCarts (warehouse storage containers) [6]. These are designed for warehouse usage which includes rugged industrial design choices that are

unfit for the office applications that our project is geared towards.

Similar to swarm warehouse robotics, there was a project that tested the feasibility of swarm applications called PARROT. The implementation used several robots to transport pallets in parallel, offering large speedups [11]. Although there are similar aspects in perception and path planning, we note that their implementation focuses on scale of parallelization in a proof-of-concept fashion, and is not applicable nor actually usable in the office environment.

Another application that is similar in use case and technologies to our implementation are the food delivery robots we see on the streets, with the most notable example called Starship. Such robots use different perception technologies than ours, and they do involve heavy path planning. They are also designed for much more rugged outdoor environments and as such their form factor would not fit well in the open office environment.

In a familiar vein, a robot that uses similar technologies and has to do with garbage pick up and cleanliness rests in Recycle Bot, an autonomous robot that uses LiDAR, CV and a RRoomba drive system to identify plastic bottles on the ground to pick up and store on its onboard container [8]. While thematically similar and using similar technologies, our use case is quite different from theirs, and as such the mechanical portions of our designs differ greatly.

We notice that there are a great deal of applications that are geared towards moving something around to make human lives easier. We find that as such our implementation is but another iteration, or rather step, in the frontier of robotics.

X. Summary

In all, our design uses tried and tested systems and combines them in a way to provide a solution that can greatly streamline and improve the logistical challenges of maintaining large open offices. The autonomous nature of the robot means that one time setups in the office greatly improves efficiency and gets rid of a large menial task for custodial workers, and can be easily scalable to cover multitudes of areas.

Nonetheless, while the benefits scale greatly, we face challenges in implementation both due to the technical nature of the task as well as our inexperience. We foresee large challenges with working with ROS as well as integrating the open source resources we are using. Control logic will be a challenge when integrating data pipelines from multiple sensors, including optometry, CV, and SLAM applications. Another technical challenge we face is having our docking mechanism align correctly, as this would require fine-tuned controls as well as having the mechanical strength to lift up the 10lb trash bin outlined in our use-case requirements.

However if we can overcome these challenges we end up with a flexible, and robust implementation that can be applied to thousands of offices across the world. In addition, the modular nature of software and robotics means that scaling the implementation and adding additional use cases is not only feasible, but easy to do. We believe our solution to be one that can provide great efficiency and betterment of the modern workforce, and is another step in the robotic revolution.

Glossary of Acronyms

BLS – Bin Lift System
CV – Computer Vision
LiDAR – Light Detection and Ranging
MQTT – Message Queuing Telemetry Transport
OBD – On-Board Diagnostics
ROS – Robot Operating System
RPi – Raspberry Pi
SLAM – Simultaneous Localization and Mapping
acml - Adaptive Monte Carlo Localization

References

[1] “Camera Basics for Visual SLAM”, Accessed on 3/1/2023, [Online]. Available: https://www.kudan.io/blog/camera-basics-visual-slam/#:~:text=The%20ideal%20frame%20rate%20for,fps%20based%20on%20the%20application.&text=There%20are%20ways%20to%20increase,based%20on%20the%20use%20case.

[2] “diff_drive_controller” Accessed on 3/3/2023, [Online]. Available: http://wiki.ros.org/diff_drive_controller

[3] “gmapping” Accessed on 3/3/2023, [Online]. Available: http://wiki.ros.org/gmapping

[4] “hector_navigation” Accessed on 3/3/2023, [Online]. Available: http://wiki.ros.org/hector_navigation

[5] “hector_slam” Accessed on 3/3/2023, [Online]. Available: http://wiki.ros.org/hector_slam

[6] Jed John Ikoba, “Amazon announces the Proteus, a fully autonomous warehouse robot”, Accessed on 3/3/2023, [Online]. Available: https://www.gizmochina.com/2022/06/23/amazon-proteus-fully-autonomous-warehouse-robot/

[7] “iRobot® Create® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600”, Accessed on 3/3/2023, [Online]. Available: https://www.irobotweb.com/-/media/MainSite/Files/About/STEM/Create/2018-07-19_iRobot_Roomba_600_Open_Interface_Spec.pdf

[8] Meghana Keeta, Serena Ying, Mae Zhang “RecycleBot” Accessed on 3/3/2023, [Online]. Available: http://course.ece.cmu.edu/~ece500/projects/f22-teama4/wp-content/uploads/sites/213/2022/12/Keeta_Ying_Zhang_final_report.pdf

[9] Meta Floor Plans, Accessed on 3/3/2023, [Online]. Available: https://www.vno.com/office/property/770-broadway/3311677/landing

[10] “navigation” Accessed on 3/3/2023, [Online]. Available: http://wiki.ros.org/navigation

[11] Prithu Pareek, Omkar Savkur, Saral Tayal, “P.A.R.R.O.T: Parallel Asynchronous Robots, Robustly Organizing Trucks” Accessed on 3/3/2023, [Online]. Available: http://course.ece.cmu.edu/~ece500/projects/f22-teama2/wp-content/uploads/sites/211/2022/12/Capstone_Design_Report-2-compressed.pdf

[12] “Python Tethered Driving”, Accessed on 3/3/2023, [Online]. Available: https://edu.irobot.com/learning-library/python-tethered-driving-with-create-2

[13] Fernando Souza, “3 Ways To Calibrate Your Camera Using OpenCV and Python”, Accessed on 2/24/2023, [Online]. Available: https://medium.com/vacatronics/3-ways-to-calibrate-your-camera-using-opencv-and-python-395528a51615

Description	Manufacturer	Model	Quantity	Cost (Dollars)
NEMA 17 Stepper Motor	Twotrees	Twotrees-16565	3	$27.59
TB6600 Stepper Driver	OUYZGIA	200205003	2	$19.99
7 Gallon Trash Bins	Continental Commercial	B08PDV3YY7	2	$23.08
USB C to 4x USB A adapter	Keymox	B0835L59N2	1	$9.55
24Ah battery, 12VDC & USB out	SinKeu	HP500S	1	$99.99
Jetson Computing Device	NVIDIA	Xavier AGX	1	~$1900.00
2D, 360 degree LiDAR system	SlamTec	A1M8	1	$99.99
Programmable Roomba	iRobot	Create2	1	$199.99
1080p Camera	TedGem	CE0140_01	2	$27.80
3D printed components	Zachary Mason/TechSpark	N/A	Variable	Variable
Laser cut components	Zachary Mason/TechSpark	N/A	Variable	Variable

TABLE II. BOM

Team B4: Trash Concierge

18-500, Spring 2023
George Gao, Jack
Girel-Mats, Zach
Mason

Project Start: Mon, 1/30/2023
Project End: Sun, 4/30/2023

George
Jack
Zach

TASK	ASSIGNED TO	PROGRESS	START	END
Initial documentation				
Project Proposal	Zach, George, Jack	DONE	1/30/23	2/4/23
Website Setup	Zach, George, Jack	DONE	2/3/23	2/4/23
Pathfinding and database				
Database setup	George	0%	2/10/23	2/15/23
SLAM/LIDAR bringup	Zach, Jack	0%	2/17/23	2/22/23
Data capture testing	George, Zach	0%	2/22/23	2/27/23
Simulate pathfinding	George, Zach	0%	2/12/23	2/17/23
SLACK		0%	2/27/23	3/20/23
Movement and local bin detection				
Roomba movement	Jack	0%	2/13/23	2/17/23
Local bin identification	George, Jack	0%	2/17/23	2/20/23
Movement towards bin	George, Zach	0%	2/22/23	2/27/23
SLACK		0%	2/27/23	3/20/23
Bin/robot hardware				
Bin attachment design	Zach	0%	2/10/23	2/15/23
Roomba fixture design	Zach	0%	2/15/23	2/22/23
Docking system test	Zach	0%	2/22/23	2/27/23
SLACK		0%	2/24/23	3/20/23
System integration				
Automatic bin docking	Zach	0%	3/20/23	3/27/23
Movement with SLAM	Jack, George	0%	3/20/23	3/27/23
Bin pickup w/ database	George, Jack, Zach	0%	3/27/23	4/3/23
Multi-bin pickup	George, Jack, Zach	0%	4/3/23	4/10/23
Bin replacement	George, Jack, Zach	0%	4/10/23	4/17/23
SLACK			4/17/23	4/28/23

Jan 30, 2023 · Feb 6, 2023 · Feb 13, 2023 · Feb 20, 2023 · Feb 27, 2023 · Mar 6, 2023 · Mar 13, 2023 · Mar 20, 2023 · Mar 27, 2023 · Apr 3, 2023 · Apr 10, 2023 · Apr 17, 2023 · Apr 24, 2023

TABLE III. GANTT CHART

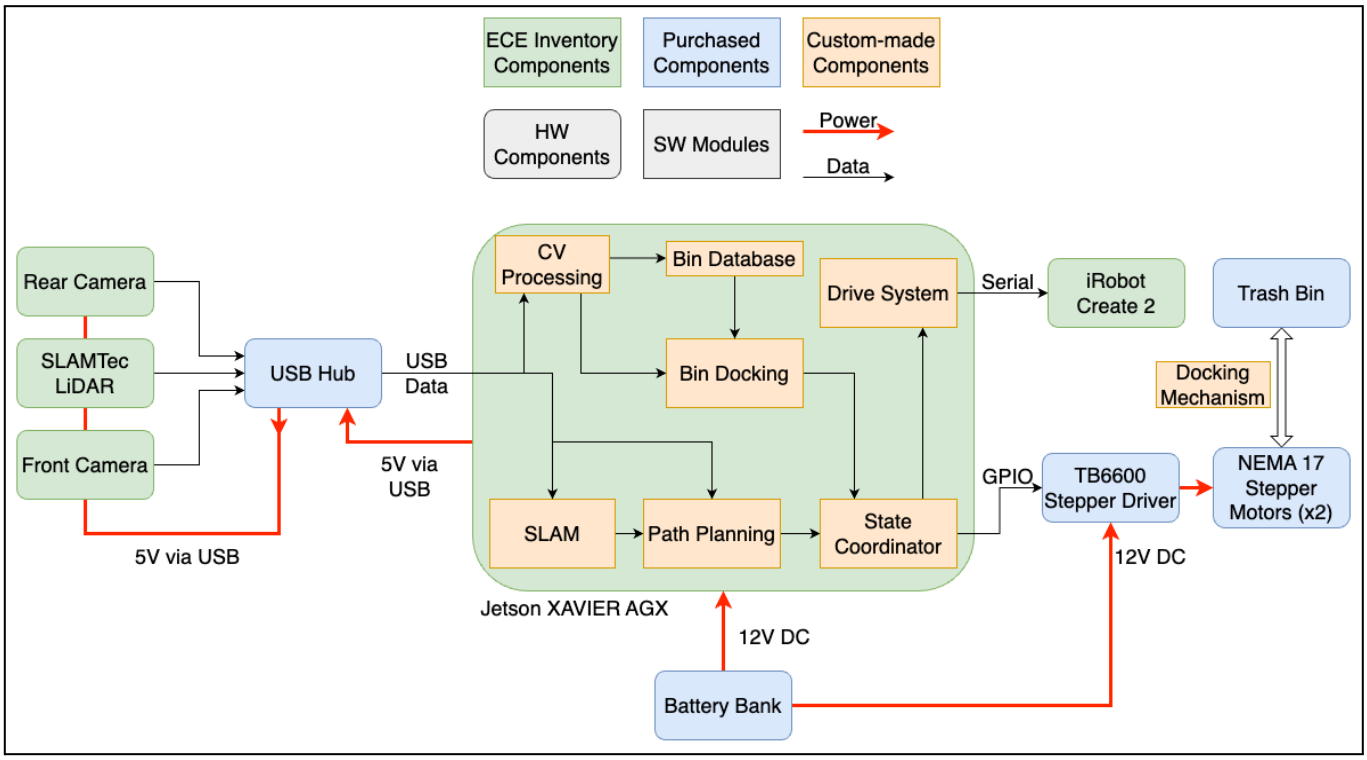

FIG 9: SYSTEM BLOCK DIAGRAM