Cleanbot 3000

Ori Dabach, Mario Valdez, Javier Reyes, Ramisa Aanan, Jose Flores, Jeffrey Broquist, Alejandro Cruz, Daniel Lopez, Rojelio Bejar, Cristian Ticas, Robert Alwin, Miguel Lucero, Adrian Gunatilake, Morris Blaustein, Sarah Blazic, John Dizon, Melissa Luna, Elmer Espinoza, Ernesto Silva
Advisor: Professor James Flynn
California State University Northridge
April 19, 2022

Abstract - simulations and live experiments to test algorithms, create software based models, and organize future goals.

1. Introduction

New:
1st paragraph:
The CleanBot 3000 team

2nd paragraph:

The objective of the navigation team is to develop an algorithm using ROS (robot operating system) to simultaneously map and navigate the JPL cleanrooms. Its purpose is to ensure that cleanbot 3000 can efficiently sanitize the cleanrooms using the UVC (Ultraviolet type C) LEDS while avoiding all equipment and hardware in the cleanroom.

2. Mapping Navigation

2.1 Previous Work

In the previous semester, the Navigation team was able to incorporate SLAM (simultaneous localization and mapping) with navigation goals and path planning to get a functional navigation stack for simulations. The ROS Gazebo simulator was used in conjunction with RVIZ to run a turtlebot3 model using SLAM and navigation goals through a simulated world. In order to keep the Turtlebot away from all obstacles and walls, the inflation radius parameter, in the navigation stack, was implemented and analyzed. A coverage path algorithm involving map segmentation was researched as a potential solution to optimize the amount of resources needed to clean the entire space of the JPL clean room.

2.2 Plan of Action

The plan of action for the Mapping and Navigation team for the Spring 2022 semester was to work with ROS Visualization (RVIZ) and Gazebo simulator to run SLAM and navigation goals to visualize a turtlebot3 either simulated or real. After extensive research, the navigation team identified three algorithms that best meet the requirements set out by JPL: Rapidly Exploring Random Trees (RRT), Particle Swarm Optimization (PSO) and A-star (A*). After further research, it was concluded that the A-star algorithm should be chosen for the Cleanbot's coverage path planning algorithm as it's well documented, has high search efficiency, and has low computational cost. The navigation team decided to use the mobile robotics simulation toolbox in MATLAB for the purpose of simulating multiple path planning algorithms to test for efficiency and avoid wasting time in

developing an unsuitable algorithm for the CleanBot. Due to the discovery of longevity issues with the toolbox when simulations were run, the decision was made to continue running the simulations in ROS instead of Matlab. A prototype simulation of the A-star algorithm was built in ROS.

2.3 Research

The CleanBot 3000 requires a path planning algorithm to accomplish its 1.1 requirement set by JPL: CleanBot 3000 shall navigate the clean room. To generate a path planning algorithm, it's necessary to identify the type of coverage path planning (CPP) it will use: either a point-to-point path planning or full-area coverage path planning [1]. Since the purpose of the Cleanbot is to clean as much of the surface area of the room as possible, a full-area coverage path planning algorithm would be best. CPPs can be categorized into classical and heuristic-based algorithms [2]. The main differences between the two types are in reference to their ideologies. Heuristics are used to obtain an acceptable solution in a timely manner, but their tradeoff is that they do not provide the best or most precise solution as classicals do [3]. From the three preferred algorithms PSO and A-star are of the heuristic type while the RRT is considered to be a classical-type algorithm[2]. A brief overview for RRT, PSO, and A* will be discussed followed by the reasoning that led to the decision of choosing A* as the main algorithm for the CleanBot 3000.

Rapidly Exploring Random Tree (RRT) is a sampling-based planning algorithm within the heuristic category of algorithms. RRT functions by using a random generation of nodes in a tree structured form to make a map of the environment [2]. They have good performance in a variety of environments including dynamical, multi-dimensional, partially structured, and unstructured environments [4]. The algorithm can also effectively search in high dimensional spaces and handle kinodynamic planning [2].It requires no model of the environment or the need to preprocess the space, however it does have the disadvantages of poor stability while searching and blindness [5].

Particle swarm optimization (PSO) is an evolutionary algorithm under the swarm intelligence category, a subcategory of the heuristic-type algorithms [2] [6]. Overall, PSO is based on the social dynamics of a population [6]. Its simplicity and easy implementation has led to its use by many researchers [6]. PSO consists of swarm agents, where each agent is judged on its performance and influences adjacent agents [6]. While PSO has global searchability in the initial stage, the swarm can be easily trapped in obstacles, leading to a slow convergence rate during the late search process [2]. PSOs performance has the possibility of reaching a rapid deterioration state when dealing with multi-dimensional environments [2].

A* is a popular graph-search algorithm in the heuristic-type algorithm [2]. It is widely used due to its performance and accuracy and therefore has a lot of documentation supporting it [1]. The algorithm traverses environments by finding the shortest path between multiple points, called nodes, within a local section of the room instead of processing the entire map

[2]. This is accomplished by minimizing the number of turns and reducing the processing time when searching for a path [2]. While the algorithm does have difficulty covering areas near obstacles and is also likely to revisit a previously traveled area over a new one, these lesser setbacks can be overcome by adding modifications to the algorithm [2].

After extensive research it was hypothesized that due to the current makeup of the cleanbot and the requirements set by JPL the A* algorithm was chosen for the CleanBot3000. At a first glance comparison of the other algorithms shown in Fig. 1, it would seem that A* is not the ideal choice since it does not do well with backtracking nor can it utilize local path planning efficiently. However, based on these JPL specifications:

- CleanBot 3000 shall be able to locate its position in the cleanroom.
- CleanBot 3000 shall be programmable with "no-go" areas which specify the location of the flight hardware.
- CleanBot 3000 shall be able to detect and communicate with other CleanBots.
- CleanBot 3000 shall have a swarm of robots that are a TBD distance away from each other.

the A* algorithm proves to be the best option largely in part to its low computational cost and efficiency in reaching its goal. The use of a low computational cost algorithm prolongs the sanitization run time as the onboard processing consumes less energy. This in turn reduces the impact processing the algorithm has on the power budget of the robot. Furthermore, a low cost algorithm is less likely to contribute to a software failure when all the supporting hardware, i.e. motor controllers and sensors, share space with the same main processing hardware. Moreover, as more robots are added to the project additional network hardware will require computational resources to allow intercommunication between the robots. In addition to the low computational cost, the A* algorithm is determined to have a rapid search time which allows for greater sanitization coverage within the same time frame. Future tests are needed to validate that the A* algorithm will be the most efficient. As an added benefit, A* algorithm is shown to be at a high level of maturity which will allow for a faster deployment as the development phase ramps up [2].

Comparison of coverage path planning algorithms

Performance Metrics	Sampling-based planning	Graph Search Algorithm	Evolutionary algorithms
Fast searching time			
Collision avoidance	✓	✓	✓
Complete coverage	✓		
The shortest path between two points	✓	✓	
Non-backtracking			
TSP optimization			✓
Large-scale structural coverage (SCP optimization)	✓		
Good real-time performance	✓		
Low computational cost		✓	
Dynamic environment	✓	✓	✓
Global path	✓	✓	✓
Local path	✓		
Experimentally sufficient		✓	
Maturity level	✓	✓	✓

Fig. 1. Comparison of various coverage path planning algorithms: Performance and analysis [2].

2.4 Implementation

2.4.1 MATLAB

2.4.1.1 Overview

MATLAB is a well-known tool for research and development throughout the science and engineering industry. Within

MATLAB are many specialized packages, known as toolboxes, for working on specialized areas within the STEM world. To work with autonomous robotic systems, MATLAB has some available toolboxes that aid in the development of said systems including Robotics System Toolbox, Navigation Toolbox, Model Predictive Control Toolbox, Reinforcement Learning Toolbox, and Stateflow. Aside from the MATLAB-provided toolboxes, there are third-party toolboxes that offer specialized services not found in MATLAB-sponsored packages. One such package is the Mobile Robotics Simulation Toolbox. The subsequent section will detail the functions found in Mobile Robotics Simulation Toolbox as well as show the outcome of a simple coverage algorithm implemented using the tool.

The Mobile Robotics Simulation Toolbox was built on the MATLAB 2019b version and requires the following packages: Simulink, StateFlow, Robotics System Toolbox, and Navigation Toolbox [7]. From the documentation, the tool can model the kinematics, sensors data, to create a visual representation of a robot in motion. To get an accurate simulation for the Cleanbot 3000 the tool was given the specifications of the Turtlebot3 from the Robotis website, along with the Lidar specifications from the SLAMTEC website for the RPILIDAR A1.

2.4.1.2 Robot Motion

The initial parameters taken into consideration in the mobile robotics simulation toolbox are the physical specifications that are defined for a Turtlebot3. The turtlebot has a two-wheel differential drive system where each wheel moves independently. The wheel radius is 33 millimeters with a wheelbase of 160 millimeters measured from left wheel center to right wheel center [8]. Fig. 2 shows the wheel radius, R, and wheelbase, L, along with the other kinematic parameters [7]. From the wheelbase and wheel radius the program creates a DifferentialDrive object to control the motion of the robot, including the forward and backward motion as well as the angular speed required for turning the robot. The linear velocity, v in units of m/s, is represented by (1), the angular velocity, w in rad/s, is given by (2), together they form the forward kinematic equations [7]. To turn the robot one wheel turns faster than the other one. The speed of each wheel is calculated automatically by the toolbox using (3) and (4) known as the inverse kinematic equations [7].

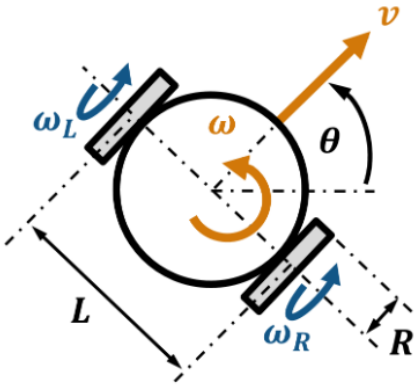


Fig. 2. Diagram of a two-wheel differential drive robot [7].

$$v = \frac{R}{2}(w_R + w_L) \quad (1)$$

$$w = \frac{R}{L}(w_R - w_L) \quad (2)$$

$$w_R = \frac{1}{R}(v + \frac{w_L}{2}) \quad (3)$$

$$w_L = \frac{1}{R}\left(v - \frac{w_L}{2}\right) \quad (4)$$

2.4.1.3 Sensor Data Acquisition

To detect objects including the building walls the Turtlebot3 uses lidar scan technology. The mobile robotics simulation toolbox requires the maximum range the lidar can reliably detect objects, the angle of observation and the number of points within the observation sweep. In this implementation the program was given a maximum distance of 12 meters, an observation window of 360 degrees and a total of 50 uniformly distributed observation points. This puts an observation point every 7.2 degrees for a complete 360 degree scan, Fig. 3 shows how the lidar scan points are represented in the MATLAB toolbox - the blue dashed lines. Within the figure the robot is represented by the blue blox with the red line relaying the robots forward facing direction. The lidar rays show the distance detected from each scan point at the current position of the robot.

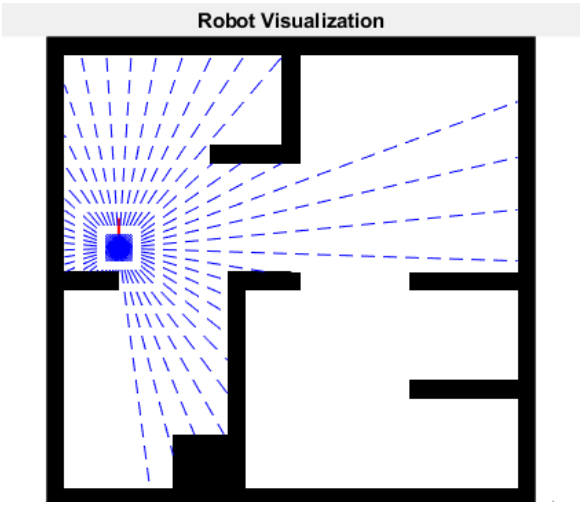


Fig. 3. Lidar on a robot showing the distance to obstacles.

2.4.1.4 Coverage Path Planning

As a starting algorithm, a linear drive program was implemented that would traverse the simulated space in straight lines alternating between vertical and horizontal motion, with respect to the room, changing direction when it reaches an object. Using this pattern, the algorithm covers the room moving one wheelbase distance of 160 millimeters after each turn. Following the JPL specifications of keeping the robot one meter distance aways from any hardware in the cleanroom, the algorithm is given a safety distance parameter of one meter. At the point the robot detects an object nearing the safety distance, detected by the forward-facing lidar scan points, the robot is programmed to stop at a distance equal to the safety distance plus the length of the wheelbase to do a 180 degree turn and return in the direction it came from, having moved 160 millimeters to the side - assuming no object is within the turning distance in either direction. Fig. 4 provides an outline of how the algorithm handles turning the robot to the right.

When the robot cannot turn in the direction it was going to continue covering unexplored space, the robot rotates 90 degrees in either direction to switch to horizontal motion. In this configuration the robot moves west to east and covers the room moving one wheelbase north after each turn.

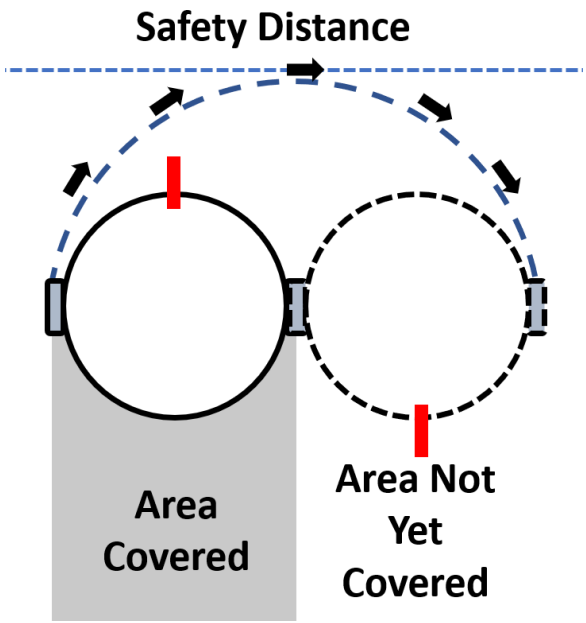
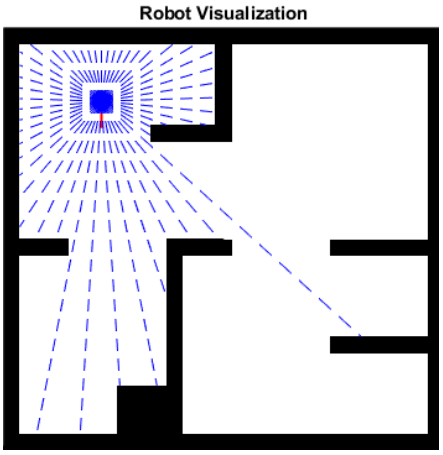


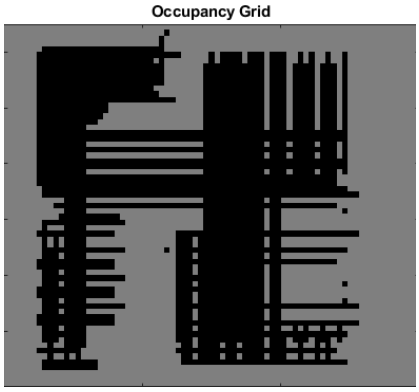
Fig. 4. The differential drive method of turning when it reaches the safety distance.

2.4.1.5 Coverage Path Planning

The algorithm described here was run for a simple pass with a resulting coverage map shown in Fig. 5. The figure clearly shows that the linear algorithm is capable of covering the majority of the room surface. Unfortunately, since there is minimal intelligence in the code the robot repeats areas and completely misses other regions. The code could be improved to do a better coverage of the room specially to keep track of the areas that it has already explored to reduce repeating the same areas. However, it was found that the code could not be easily exported to work in the ROS environment. This has to do with how the ROS platform is structured and the strict formatting that is required for each section that handles different parts of the robot system. The best course of option is to implement all algorithms in ROS from the very begining.



(a)



(b)

Fig. 5. (a) Sample map with the robot and (b) covered area after a single run.

2.4.2 ROS

2.4.2.1 Overview

The control of a robotic system requires a cluster of subsystems to work in unity to allow a robot to move and perform tasks. ROS is an open-sourced platform that integrates all the necessary components into a package that simplifies the build and control of a robotic machine [9]. This project used the ROS noetic release that works in conjunction with Ubuntu 20.04, the latest long-term supported version of the Linux

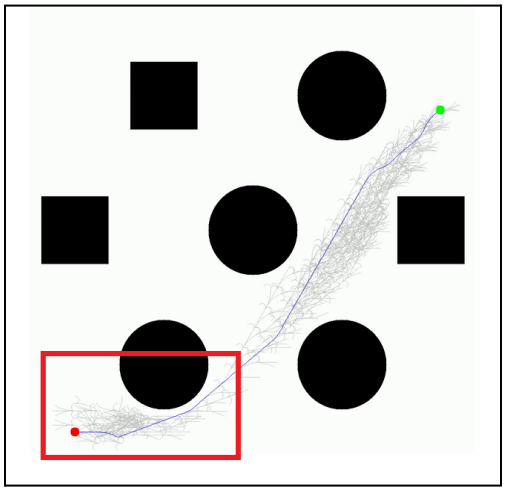
distribution operating system [9].

ROS allows abstraction of the hardware layer to allow high-level programming of the physical components including the sensors and motors that drive the machine [9]. ROBOTIS, the TurtleBot3 manufacturer, provides an open-source package that contains the necessary libraries to control the TurtleBot3 hardware. Additionally, an open-source ROS package created by Umang Rastogi provides a path planning implementation using the A* algorithm [10]. These packages were used as a base to build a custom-tailored solution to meet the requirements set by the JPL.

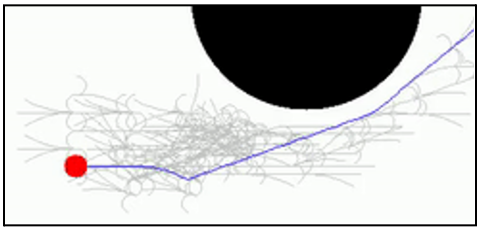
2.4.2.2 Algorithm Structure

To get as much area covered with the least amount of resources the TurtleBot is directed to move linearly as much as possible and only use A* when a new direction is needed. As obvious from the previous section, moving the robot forward takes less computation power than making the robot turn to another direction. Here, the direction the robot moves forward in is formulated using the lidar scan data. From this data, the furthest unobstructed point discovered is set as the goal point for the robot. Once the goal is set, the navigation stack is tasked with moving the robot to its destination.

The navigation process is primarily influenced by two key components: the occupancy map and the A* algorithm. As the robot traverses the room, the occupancy map is updated to identify the covered area which in turn updates the future goal position of the robot. In this process, the A* algorithm tests the areas around the robot to determine the best path to take to reach the goal position. After the goal is met, the robot once again processes the new scan data to determine the new goal position. Fig. 6 shows a sample map where the A* algorithm plots all the neighboring nodes along with the optimal path.



(a)



(b)

Fig. 6. (a) The a-star algorithm as pathfinder, (b) close up display showing adjacent nodes [10].

2.4.2.3 Grid Segmentation & Evaluation

As described in the Research subsection of the this paper the A* algorithm takes a region of space and splits it into a grid made of smaller fragments called nodes [1]. In the case of the TurtleBot, the region is bound to the space the lidar can detect. Each of the adjacent nodes surrounding the robot are then evaluated by the algorithm to find

the best path for the robots to take [8]. Fig. 7 shows the A* evaluation of multiple adjacent nodes when determining an optimal path. Evaluation of each node encompasses calculating the amount of resources required to move the robot to the candidate position, defined by g(n), the cost function, as well as calculating how the new position impacts the distance to the goal point, defined by h(n) [11]. The evaluation function is the sum of these two functions given as (5) [11].

$$f(n) = g(n) + h(n) \quad (5)$$

The move cost function, g(n), is best described by (6) where θ is the angle in degrees that the robot turns and the x and y lengths are defined as the Manhattan distance in [4] where the x_c is the current x coordinate and x_t is the target or goal x coordinate, the same with the y points. The use of the Manhattan distances is to simplify the calculations since the resulting value is only used in the evaluation. The heuristic portion, h(n), of the overall cost function, f(n), is formed with the information from the lidar scan. In other words, the robot estimates the goal point to be the furthest point away from the already covered area.

$$g(n) = |\theta| + |x_c - x_t| + |y_c - y_t| \quad (6)$$

As an example, the evaluation of g(n) for a node that requires the robot to rotate results in a higher cost of resources than a node directly in front where the robot only has to move forward to the adjacent node. Alternatively, if the same node is closer to the final goal position it would result in a smaller value to the h(n) function. Fig. 7 gives a sample cost map grid for adjacent nodes, where the robot is at the central square. As the reader can tell the front facing nodes have the lowest costs while the rear facing have the higher cost as these require more resources from the robot. The size of the grid is dependent on the maximum range of the lidar the robot is equipped with.

49			3			49
	48	93	2	93	48	
		47	1	47		
93	92	91		91	92	93
	138	137	181	137		
	138	228	182	228	138	
139			183			139

Fig. 7. The cost grid of adjacent nodes with the robot facing upward.

2.4.2.4 ROS build

To implement this algorithm in ROS the program requires customizing several of the built-in navigation functions including the local planner, global planner. costmap, and the map server. The map server works as the occupancy map that updates as the robot moves through the room. The map server is updated to capture the covered area similar to what was presented in Fig. 5 from the MATLAB subsection above. The costmap serves to help determine the best path but can also visually represent the data of Fig. 7 in RVIZ as well as keep track of costs as the robot moves in the room [7]. The local and global planners are the main workhorse because they are responsible for directing the robot through the map. For every lidar scan the local planner is tasked with determining the most suitable path for the TurtleBot [7]

On the other hand, the global planner takes data from the occupancy grid to determine where the next goal is to be placed. It is here in the global planner where the main algorithm executes and the more basic linear algorithm is executed in the local planner. A small but crucial parameter needs to be adjusted to maintain the requirement of maintaining, at all times, a distance of one meter between the hardware and the robot, the inflation radius. The inflation radius allows the developer to adjust the minimum distance the robot can reach to any object including walls [9]. After implementation, a 3D rendering is given in Gazebo to visually observe the behavior of the robot with the algorithm. Fig. 8 gives a snip of a simulation showing a virtual TurtleBot as it moves through a constructed room.

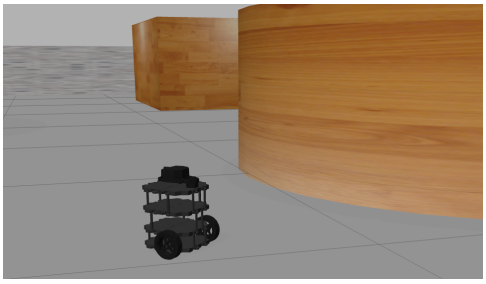


Fig. 8. Gazebo simulation showing a turtlebot in a virtual environment.

2.4.2.5 Conclusion

The algorithm presented in this paper combines the reduced computational needs of using a linear algorithm and the accurate convergence to a solution of the A* algorithm. The MATLAB section shows that with a linear algorithm high coverage is possible however the strategy makes use of multiple passes through areas that are already covered which reduces the efficiency and consequently requires a longer run time. By adding A* to the process a much more effective job can be accomplished. As more turtlebots are added to the project this algorithm naturally divides the area into small grids for each robot to cover without the need for much modification to the application.

3. Marker Localization / Systems

3.1 Previous Work

The Systems team was created this semester in order to integrate all of the Cleanbot's software and hardware systems together. In previous semesters, the Marker Localization team worked on locating markers known as ArUco markers so that the Cleanbot would be capable of avoiding flight hardware in JPL's cleanrooms. By placing these markers in the imaging system's viewpoint, they appear in the system's generated image and from there, can be used as reference points and to gauge distance between the Cleanbot and any obstacles. The team can create these markers in any size to mark all types of hardware and ensure that it's avoided by the Cleanbot.

During the Fall 2021 semester, the team successfully recognized the ArUco markers at greater distances and in dimmer light conditions than before with the use of an upgraded camera. The imaging system hardware that was used last semester was the Raspberry PI High Quality (HQ) camera with the 6mm Arducam lens [reference last semester's paper]. The camera was calibrated by installing camera calibration drivers and by using a large 8x6 checkerboard with 74mm squares [reference last semester's paper]. After calibration, various tests were composed for fiducial marker detection at different angles, distances, and heights. The fiducial detection software was run on four

Linux terminals confirming calibration, displaying the number of markers detected, and creating a map using the Cleanbot's position so that the markers could be detected and then visualized on RVIZ software. Finally, the camera's detection abilities were tested in various rooms with dim and bright lighting, near and far distances, and direct and peripheral angles. The camera detected the markers in all lighting conditions, the 9.5 x 9.5 inch markers were detected at a distance up to 30ft, while the 11 x 11 inch markers were detected at a distance up to 60ft. It also successfully detected markers at 19ft with a 27 degree angle and at 15ft with a 23 degree angle.

3.2 Plan of Action

In the Spring 2022 semester, the systems team focused on expanding the sensor network of the Cleanbot to include the Zynq 7000 SoC in addition to the Raspberry Pi. The Digilent Zybo Z7-10 development board was chosen because the team was familiar with it and able to program it with Xilinx software. The Zybo Z7-10 has 5 PMOD ports, which is ample room for the sensor network. The Raspberry Pi will still be needed to capture image, lidar and ultrasonic data. The Raspberry Pi will be connected to the Zynq 7000 via a UART serial connection and send data to be processed by the Zynq 7000 for the path planning and marker localization algorithm.

Currently, the Systems team has successfully implemented the TMP3 PMOD temperature sensor with the Zybo Z7-10 and is able to receive temperature data and display it to the UART serial console. Digilent provided an IP for the TMP3 PMOD as well as a software driver, which allowed the team to get the sensor successfully functioning with the Zybo Z7-10. One of the challenges faced was finding sensors that had IP and software drivers, so the team is researching which relevant PMODs have existing IP's and software drivers. The systems team will coordinate with the navigation team to incorporate the Raspberry Pi and its sensors with the Zybo Z7-10. The systems team will continue to use the UART serial console to monitor and debug application development. Consideration of requirements and efficiency will be taken when planning the PS-PL interface on the Zybo.

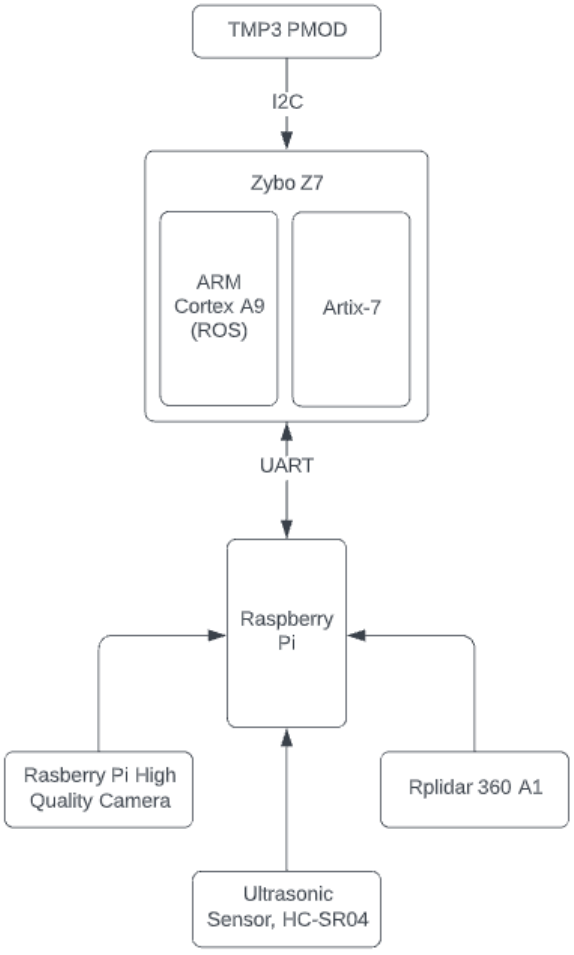


Fig. 9. Cleanbot sensor network system diagram

3.3 Research

The Marker Localization team decided to transition into a team that combines the software and hardware used by the Navigation team and Power team onto a single platform. Up to this point, they have worked mostly independent of each other, but the goal of the Systems team is to cover both teams.

To familiarize themselves with FPGA and SoC design, the team used the Zybo FPGA board due to most of the members having access to one. Xilinx Vivado and Vitis software were used to create hardware designs and boot files to load onto the FPGA. The Vivado software provides a platform for analysis and synthesis of hardware description language designs, allowing for the creation of hardware designs to run on the FPGA. The Vitis software is for writing the software that will run on the FPGA, also providing an environment for debugging the designs through software[13].

The team went through the Requirements and Specifications of the Cleanbot 3000 and found the requirements they determined should be handled by the Systems team. The first requirement of the control system for the Cleanbot is to monitor temperature, UV light, and battery levels at all times. The fault protection states that the Cleanbot 3000 shall shut down if the battery reaches TBD temperature when discharging or temperature when charging.

With this requirement, the team decided to find a temperature sensor that can read the temperature of the battery. looked into different temperature sensors and looked into I2C compliant sensors so that they may implement other I2C devices into the design for UV light and battery level monitoring.

Two sensors were found that would be fit for use as battery and UV temperature sensors. The first sensor chosen was the Digilent PmodTMP3, an ambient temperature sensor with an operating temperature range of -40°C to +125°C, with a temperature resolution of up to 0.0625°C [14]. The second sensor chosen was the TE Connectivity TSYS01, with an operating temperature range of -40°C to +125°C, with a temperature resolution of 0.01°C [15].

The TSYS01 was favored due to its higher accuracy, adjustable temperature range, and the possibility of measuring the battery temperature directly. It was ordered and a schematic was designed but the boot file could not flash onto the Zybo board. After some troubleshooting, it was decided to use the TMP03 sensor instead since there was more documentation on how to use it on the Zybo board.

The team was able to get the TMP3 working and they were able to receive temperature readings. With the TMP3 reading ambient temperature, the battery temperature will be measured by taking a "zero" temperature which would be the temperature of the battery when enclosed in the Cleanbot. The temperature sensor the team plans to use will function as a backup sensor in the case that the battery’s built-in temperature sensor were to fail.

The team looked into further use of the LogiCore IP AXI IIC Bus Interface that allows for interfacing multiple I2C-

compliant devices. I2C devices allow the team to implement the sensors onto a microcontroller or micro computer like a Raspberry Pi, giving the team more flexibility on what sensors they decide to use for measurements.Such devices include an UV light sensor to help keep track of the UV levels of the Cleanbot's UV-C LEDs.

3.4 TMP3 Temperature Sensor

To implement the temperature sensor using the Zybo Z7-10, the software Vivado and Vitis were used to create a block design and program the board. The team wanted to take advantage of Vivado intellectual properties (IPs) to simplify the implementation of the sensors. For this reason, the TMP3 PMOD sensor from diligent was chosen.

With the TMP3 acquired and the files for the Zybo downloaded, a block design was created using Vivado. The design included the Zynq processor IP, UART PMOD IP and TMP3 PMOD IP. The UART IP will not be necessary for the final design but was used for additional practice with the Zybo and software. After the block diagram was completed, a bit stream was generated that included things like the routing, hardware logic, and initial values for the memory to configure the FPGA.

When the design is exported, a .xsa or Xilinx support archive file is created. This is a proprietary file format specific to Xilinx software and tools. XSA is a container that contains one or more .hwh or hardware handoff files. These files hold the information from the block design which includes but is not limited to: internal connectivity information, memory map information of processes, IP information, BIT files, and more [16] .

With the .xsa file generated, Vitis was used to program the board using the file exported from Vivado. This was done using a C code example file given with the TMP3 documentation. It outputs the temperature in both celsius and fahrenheit with two degrees of accuracy through the UART to a terminal in Vitis. This information will be pivotal to monitoring the temperature batteries and UV lights to prevent damage from overheating and gasing.

4. Power & Mechanical Design

4.1 Previous Work

In the previous semester, Fall 2021, the team continued researching crucial components that were needed in the Cleanbot's design. After being notified that funding was unavailable, the team focused on ideal versus low budget plans. The team was not able to finalize any material as the dosimetry test experiment was still in the process of being finalized. The Solidworks team decided to model a 4 wheel chassis design while also incorporating a Turtlebot 3000. Some digital models were created to model the cleanbots potential. Members of the Solidworks team were informed and assisted in designing this chassis framework. As for the motors team, some additional motors were researched and added to the powers team documentation. Brushless DC motors with and without integrated speed controllers were researched. The wheels team researched better quality wheels and different companies that could possibly provide the components necessary for the design.

Previously, the batteries and charging team conducted research on low budget batteries that meet the requirements of the Cleanbot. The team created documentation to organize their findings. Many of the researched products provided the bare minimum for the robot's necessities, however, such products did not provide added benefits such as circuit protection or rapid charge and long lasting battery life. However, these products may prove to be useful in the prototyping phase of Cleanbot 3000 and aid in lowering the overall cost throughout the project's lifespan.

4.2 Plan of Action

For the semester of Spring 2022, the Power and Mechanical team focused on continuing the research in order to finalize the Cleanbot's components as well as introducing a sanitation sub team to interpret and apply the JPL dosimetry test data upon completion. All sub teams maintained a cost effective mindset while also researching quality and effective components that will adequately fit the robot. With funding currently unavailable, the team planned to continue cost effective research in order to aid future semesters when funding becomes available.

At the start of the semester, the team was notified that the JPL dosimetry test data was in the completion process which led to the decision to incorporate a sanitation sub team to interpret said data. With the completed test data, the power team as a whole would be able to obtain a more accurate idea of the robot's velocity and LED exposure dosage.

The Solidworks team would work on redesigning the chassis and create a two wheel caster ball design, as opposed to last semester's four wheel chassis model. The team planned to 3-D print some plastic models that give a physical visualization of the Cleanbot 3000 since a 3-D printer was made available. Using the Solidworks program to accomplish this model, the team planned to attach motors to the 3-D printed prototype in an attempt to have a mobile design.

The wheels team planned to incorporate additional research to the catalog of compatible wheels in the subteam's documentation.Unfortunately, a polyurethane caster ball wheel could not be found up to this point. This is a problem that previous semesters have not been able to accomplish as well. The wheels team has also planned to eliminate the research of Omni wheels, with rollers located around the circumference, from the group's interest.

As for the motors team, the plan was to continue researching Brushless DC (BLDC) motors and focus on finding adequate products that exhibit the lowest possible power consumption while adequately sustaining the Cleanbot. Some motors found, and listed in the motors documentation, still underperformed for what the team had in mind. The batteries and charging team would be researching cost effective and efficient ways into wireless charging of the bot.

4.3 Power System Design

4.3.1 Batteries/Charging

The batteries and charging team focused on selecting a battery which would adhere to the required specifications given by JPL. The team used a simulink model which

was created from a previous semester which was modified to be able to calculate the total power of the bot. The power team was dealt with multiple options on how the battery can be implemented on the bot. The selected battery was the NH3054QE34 by inspired energy. The chosen battery has a 6.8Ah capacity, when two are connected in series the total voltage would be 28.8V [17] which is sufficient to drive the UV-C LED modules. Multiple features such as having built in sensors for short circuit protection, discharge and charge protection also made this battery a suitable fit for this design. Under current estimates two of these batteries would be sufficient to run the Cleanbot for an hour. The charger which was selected initially from previous semesters was the EB330A due to its compatibility for the desired battery. The power team will look to find new ways of being able to wirelessly charge the batteries as well as making sure it is cost effective and efficient.

4.3.2 Sanitation

Sanitation is a newly revived subteam introduced as a subgroup of the power team. This subteam has been inactive for the past few semesters but was reactivated for the Spring 2022 semester. The reasons for this decision are to allocate efforts for the interpretation of dosimetry test results performed by JPL engineers and researching adequate sanitizing products that meet the requirements of the project. During the semester, the team started by recovering the latest research performed in previous semesters. The latest research included Klaran Light Engines UV-C LED modules[18]. This semester, the team decided to continue research based on the light engine modules. Klaran LE (Light Engines) are sanitizing modules that include board mounted UV-C LEDs that allow for convenient integration for a variety of applications [19]. For this reason, the LED modules should be easily integrated for Cleanbot's applications. Currently, Klaran offers three types of modules that each incorporate 3, 9, or 12 sanitizing LEDs. The team is heavily in favor of utilizing the 12 LED count module in order to maximize exposure area. Also, the team were notified by JPL engineers that the 12-count LED module was utilized in the dosimetry test process. The part number for this specific module is LE-24V-12V-HC and comes with standardized 24Vdc and 350mA LED drive voltage and current[18].

The dosimetry test experiment performed by JPL engineers was completed and received in April. The test results are displayed in Fig. 9 and Fig. 10. The JPL engineers conducting the experiment encountered an issue with the LED module. The module did not function properly and demonstrated only a single functioning LED. For this reason the experimental results were presented with respect to a single LED exposure[19]. The bacterial growth also lacked a control sample as the bacterial colonies were too numerous to count. However, the data still proves to be useful in determining critical bot parameters such as velocity and UV-C dosage levels.

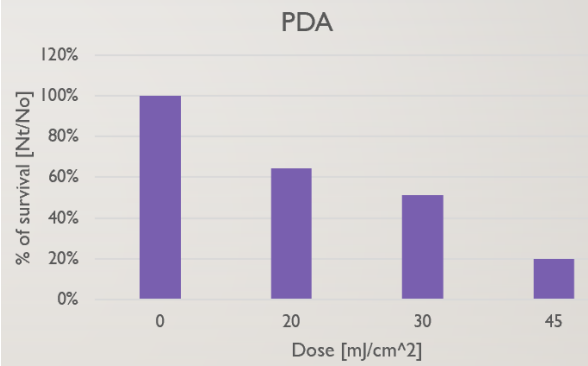


Fig. 9. % Survival of Fungal Population

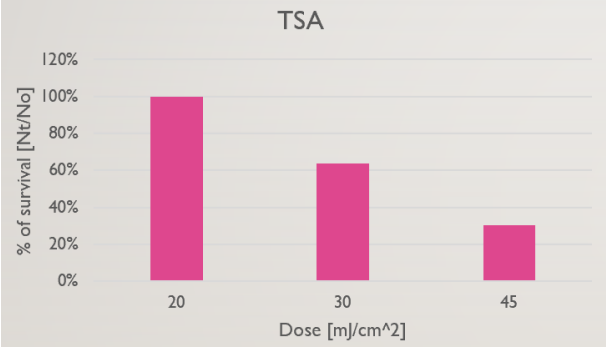


Fig. 10. % Survival of Bacterial Population

The data was extrapolated in order to determine the necessary dosage levels required for bacterial and fungal inactivation of 90%. After careful calculations, it was determined that approximately 52.14 mJ/cm^2 and 50.63 mJ/cm^2 are necessary for 90% sanitization of bacterial and fungal populations. Such levels of sanitation correspond to approximately a velocity of 9.67 m/hr if 4 LED modules are included in the final design. These findings lead the team to conclude that 90% sanitation levels may not be feasible during the desired run time. Possible solutions include testing a more powerful LED module, running multiple cleanbots simultaneously, or adjusting the desired levels of sanitation.

Other research topics for the sanitation team included circuit protection, cooling solutions, and power delivery methods. Although no definitive products were selected during this spring semester, the team compiled documentation regarding the extensive research performed on such topics. At the moment, pin heat sinks and dc boost converters seem to be the proper course of action in regards to cooling and power delivery.

4.4 Mechanical Design

4.4.1 Solidworks

Solidworks is a solid modeling engineering computer program. The reason is to create visualizations of the Cleanbot with real specifications. This semester the team focused on updating all new members with work done in previous semesters to ensure more progress can be made. The team has reviewed the components created from last semester and have begun creating a unique design of the cleanbot which can be seen in Fig. 11 as seen below.

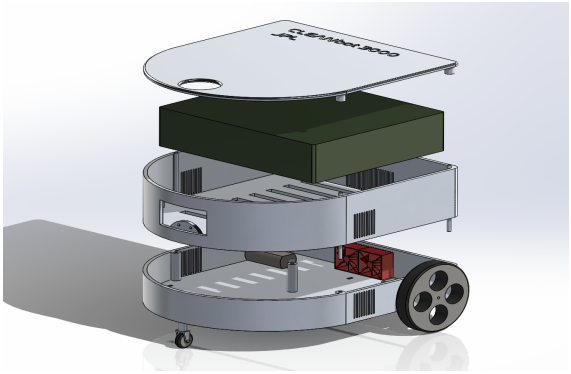


Fig. 11. Two Wheel Chassis Design

The team has since created a new design but had a lot of liberty in the design as the team have lost the TurtleBot. The student that owned the Robot has since graduated and has kept his turtlebot. As a result the solidworks team has taken steps back on the

design. The sub team has kept an open mind and decided on a two wheel caster ball chassis design different from last semester's four wheel design. 3-D printing some designs made by the members but have also obtained old prototypes from older semesters. The team has not fully decided on whether a two wheel or four wheel design will be kept as the older designs seem to be functional. This is very promising as the team have actual models that may be tested and analyzed. The team has created a model that helps with real life visualizations and has raised questions if this is the actual model the team wanted to move forward with or will the team be going with the older designs? The physical model printed out can be seen below in Fig. 12.

Fig. 12. 3-D Printed Model

As for the material used for the chassis, 80/20 Aluminum remains the material of choice per previous recommendations from JPL due its advantages. This material demonstrates great durability and does not have as much weight as other materials. It is desirable to maintain CleanBot's weight at a minimum while retaining solid durability. Furthermore, thermal analysis has not been done this semester but the analysis is used to find temperature distribution, temperature gradient, heat flow, and heat exchange between the different components of the robot. This could be useful information to add as the project progresses.

4.4.2 Motors

The motors team is responsible for keeping the Cleanbot mobile throughout NASA JPL laboratories. This semester the team has reviewed previous motors researched from past teams and have also researched additional motors. The team's hopes are to find the best performing motor that includes a high level of efficiency. This semester the team faced obstacles such as finding an effective motor that has the capabilities of the desired torque and power ratio while maintaining a relatively low power consumption. It seems that most Brushless DC motors have high power consumption but not enough torque that is necessary. Integrated controllers would be ideally included in these motors to connect to the control system used in the robot. Incorporating these controllers would give the robot the ability to program specifications. However, including the speed controllers separately would cause the motors system to occupy more space within the CleanBot. The team also needs to approximate weight for the robot before any further research can be made. The team has taken previous calculations performed by past groups into consideration and have continued looking for suitable motors in hopes to purchase some motors in the future to run more tests.

4.4.3 Wheels

The Wheels team this semester focused on suitable Polyurethane wheels that may be added to the design. The Wheels team has decided that the robot would yield two polyurethane wheels and one caster wheel design. Similar to previous semesters, the team has researched many wheel options and have found some companies that offer wheels that may be used in the final design. Polyurethane remains the material of choice with respect to wheels. These wheels possess the ability to maintain bacteria, dirt, and dust build up to a minimum as the CleanBot maneuvers throughout the JPL clean rooms. The team was notified that Omni wheels with multiple rollers around the circumference shall be excluded from the design. Omni wheels possess excessive openings and spaces where debris may have a chance to accumulate. An example of a suitable wheel can be seen in Fig. 13.

Fig. 13. Polyurethane Wheel [21]

The team has since ended the research of Omni wheels and will redirect their focus on other wheel designs.

As for caster balls, the team's efforts to find caster balls that are made of polyurethane have been unsuccessful so far. It seems that to obtain such products, custom builds may be required. In the future, the wheels team hopes to find and implement a suitable caster wheel in the final design. If future efforts remain unsuccessful, the four wheel design may be the best course of action.

5. Grant Research

6. Conclusion

As the Fall 2021 semestr.

References

[1] S. Zhao and S.-H. Hwang, "Path planning of ROS autonomous robot based on 2D lidar-based SLAM," in 2021 International Conference on Information and Communication Technology Convergence (ICTC), 2021, pp. 1870–1872.

[2] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms," IEEE Access, vol. 9, pp. 119310–119342, Aug. 2021.

[3] baeldung, "The difference between a heuristic and an algorithm," Baeldung on Computer Science, 19-Nov-2021. [Online]. Available: https://www.baeldung.com/cs/heuristic-vs-algorithm. [Accessed: 16-Apr-2022].

[4] J. Nieto, E. Slawinski, V. Mut, and B. Wagner, "Online path planning based on Rapidly-Exploring Random Trees," in 2010 IEEE International Conference on Industrial Technology, 2010, pp. 1451–1456.

[5] J. Xu, H. Xiao, and F. Huang, "Research on path planning algorithm of bidirectional rapidly exploring random tree improved by artificial potential field,"

J. Phys. Conf. Ser., vol. 2137, no. 1, p. 012051, 2021.

[6] S. I. A. Meerza, M. Islam, and M. M. Uzzal, “Optimal path planning algorithm for swarm of robots using particle swarm optimization technique,” in 2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE), 2018, pp. 330–334.

[7] MathWorks Student Competitions Team (2022). Mobile Robotics Simulation Toolbox (https://github.com/mathworks-robotics/mobile-robotics-simulation-toolbox), GitHub. Retrieved April 11, 2022.

[8] Robotis (n.d.) 1. Features. Robotis e-Manual. Retrieved April 14, 2022, from https://emanual.robotis.com/docs/en/platform/turtlebot3/features/

[9] Open Robotics. (n.d.). Getting Started. ROS Documentation. Retrieved April 14, 2022, from http://wiki.ros.org/Documentation

[10] Umang Rastogi. 2020 Implementation of A* pathfinding algorithm on a differential-drive robot. https://github.com/urastogi885/a-star-turtlebot. (2022)

[11] X. Li, X. Hu, Z. Wang and Z. Du, "Path Planning Based on Combinaion of Improved A-STAR Algorithm and DWA Algorithm," 2020 2nd International Conference on Artificial Intelligence and Advanced Manufacture (AIAM), 2020, pp. 99-103, doi: 10.1109/AIAM50918.2020.00025.

[12] S. Bobrowicz, “Zybo Z7 reference manual - digilent reference,” *Digilent.com*. [Online]. Available: https://digilent.com/reference/programmable-logic/zybo-z7/reference-manual?redirect=1.

[13]Horn, D. *What’s different between Vivado and Vitis*? Digilent Blog. https://digilent.com/blog/whats-different-between-vivado-and-vitis/

[14] “Pmod TMP3 reference manual,” *Digilent.com*. [Online]. Available: https://digilent.com/reference/pmod/pmodtmp3/reference-manual?redirect=1.

[15] Www.te.com. [Online]. Available: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet%7FTSYS01%7FA%7Fpdf%7FEnglish%7FENG_DS_TSYS01_A.pdf%7FG-NICO-018. [Accessed: 16-Apr-2022].

[16] “Documentation portal,” *Xilinx.com*. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1400-vitis-embedded/XSA.

[17]Inspired Energy, “Battery Specification,” NH3054QE34, 2019

[18] Klaran, “Klaran LE,” 120220, 2020

[19] Klaran, “Klaran WD Series UVC LEDs,” 120321, 2021

[20] Casters, RWM. “Polyurethane Wheels: Polyurethane vs. Urethane Wheels.” *RWM Casters*, RWM Caster, 1 Feb. 2022, https://www.rwmcasters.com/products/wheels/polyurethane-wheels/.