CleanBot 3000

John Dizon, Ori Dabach, Sarah Blazic, Elmer Espinoza, Robert Alwin, Ernesto Silva, Inderjeet Gawra, Janine Dabu, Kevin Torres
Advisor: Professor James Flynn
California State University, Northridge
November 15, 2022,

Abstract - The purpose of this project is to limit the spread of interplanetary bacteria contamination within Jet Propulsion Laboratory (JPL) cleanrooms. To achieve this goal, the CleanBot team will continue to develop an autonomous vehicle that can sanitize the JPL cleanroom floors during the night without causing any harm to the flight and instrument hardware. This paper presents the project's research, current progress, and plans. The CleanBot must also comply with JPL requirements and cleanroom norms. The three distinct subteams that comprise the CleanBot research and design team are Mapping Navigation, Marker Localization/Systems, and Power & Mechanical Design. Each subteam built on the work done in earlier semesters and proceeded to do additional research, build an actual prototype, and computer simulations to test algorithms, develop software-based 3D models, and plan out the following semester's objectives.

1. Introduction

The three subteams each tackle a different section of the CleanBot initiative. The Mapping Navigation team studies various path-planning algorithms through research and simulation to find the most optimal path for the CleanBot to traverse while focusing on the JPL requirements for the CleanBot 3000. The paths are judged by how much coverage the UVC LEDs have on the cleanroom's floor. The Marker Localization/Systems team focuses on capturing and processing data received from local inputs such as the Raspberry Pi camera and the LiDAR sensor. This data processing works with the Mapping and Navigation team so that the CleanBot knows where it is spatially. Finally, the task presented to the Power and Mechanical Design team is to create the power system and mechanical components of the CleanBot design.

2. Mapping and Navigation

2.1 Previous Work

During the last semester, the navigation team found a coverage path planning (CPP) algorithm for CleanBot. After researching different types of CPPs, the team narrowed down to three algorithms: The Rapidly Exploring Random Trees (RRT), Particle Swarm Optimization (PSO), and the A-star (A*). The A-star algorithm was chosen as the most suitable CPP algorithm due to its performance, high search efficiency, low computational costs, and rapid search time. At first, the team chose to use MATLAB to simulate the CPPs by using the mobile robotics simulation (MRS) toolbox. The toolbox allows for the simulation of the forward, backward, and angular velocities by creating a differential drive object for the CleanBot. In addition, the LiDAR was simulated using the MRS toolbox by inputting three parameters: the maximum range the lidar can detect objects, the number of points in the observation sweep, and the angle. After running some simulations on MATLAB, the team found the MRS toolbox would lose support soon, and to not waste time and continue progressing, and the team switched from

MATLAB to ROS. Using ROS Visualization (RVIZ) and Gazebo, a virtual environment was built to simulate the A-star algorithm on CleanBot. The team began testing the A-star algorithm to see how efficient the algorithm would be on the CleanBot and was not successful and implementing the A* algorithm.

2.2 Plan of Action

During the Fall 2022 semester, the Mapping and Navigation team utilized Robot Operating System (ROS) to use packages such as a 3d simulator (GAZEBO) and ROS visualization (RVIZ) to simulate the coverage path planning algorithm. After reviewing the previous semesters' progress on the coverage path planning algorithm, A-Star (A*), they deemed the algorithm impractical. The A* algorithm required critical points and an end goal; this made it a viable algorithm for navigating from point A to point B with the quickest and most efficient path. Despite this, CleanBot requires no critical points since the cleanrooms at JPL are constantly changing. Therefore, it cannot be given a navigation goal, and further research was done to find an alternative. We needed an algorithm that covered an environment without any critical points and with the most accurate coverage area. After much research, the navigation team chose the Back-Tracing Spiral Algorithm (BSA) for its coverage path planning algorithm. The reasoning for choosing BSA is as follows, it computes its critical points based on its sensor data. It utilizes a low computational cost. It contains a fine-grain group model. Lastly, it takes note of areas it has yet to cover and returns to them. The navigation team used ROS visualization (RVIZ) to simulate the prototype BSA algorithm.

2.3 Research

The CleanBot 3000 requires a path planning algorithm to accomplish its 1.1 requirements set by JPL: CleanBot 3000 shall navigate the clean room. For the CleanBot 3000 to traverse autonomously, it must be supplied with an algorithm for coverage path planning (CPP). To generate a path planning algorithm, it is necessary to identify the type of CPP it will implement: point-to-point path planning or full-area coverage path planning [2]. A point-to-point path-planning algorithm aims to navigate from one point to another while achieving the most efficient path, requiring critical points. Considering that the goal of the CleanBot is to traverse the entire surface without a navigation objective, the team decided to stay with last semester's decision of using a full-area coverage path planning model that does not require critical points. A technique commonly used in most CPP approaches is decomposing the environment into smaller regions referred to as cells. Cell decomposition can be applied to develop optimal path planning [1]. This will be a crucial technique for CleanBot to cover all portions of its environment. Therefore, the BSA algorithm was suitable for CleanBot to use.

The BSA algorithm can be broken into two sub-algorithms; the first sub-algorithm covers simple regions using a spiral-like path. A *simple region* is an area that can be filled with a spiral path algorithm. The spiral pathway consists of concentric rings that form a continuous pathway from the regional boundaries to the central endpoint. The algorithm uses a set of rules to traverse the spiral. In response to the robot's starting point, it searches for the nearest obstacle and uses it as a reference. The directions can be specified as RLS and OLS. RLS indicates where the obstacle should be referenced during a spiral filling procedure. OLS is the opposite lateral side. These rules are shown in Fig. 1. They describe the logic of how the algorithm interprets the start of the spiral pathway. As

the RS rules are evaluated, cells in the environment are marked as virtual obstacles, and the CleanBot will not travel to them. Commonly, the first external ring of a spiral path is performed near occupied cells [3].

RS1	IF (obstacles_all_around) THEN ending_spiral_point_detected
RS2	IF (NOT obstacle_in_RLS) THEN turn_to(RLS) and move_forward
RS3	IF (obstacle_in_front) THEN turn_to(OLS)
RS4	OTHERWISE move_forward

Fig. 1: Rules of logic for how the robot will reference location for spiral traversal

The second sub-algorithm of BSA includes a back-tracing mechanism to traverse back to unvisited areas where a new spiral procedure can be performed. Back-tracing cells (BC) are detected and stored during a spiral procedure. If BSA discovers more than one unknown cell in its area, it will use those cells as the starting point of a future alternative path. Once the robot has completed a spiral procedure and is at the endpoint, the back-tracking mechanism is invoked. The robot will traverse to the nearest unknown cell block by finding the Euclidean distance between the two points. It reaches the BC by traversing over already visited cells, and once there, it will begin the new spiral procedure. BSA is completed when there are no more BC candidates. Therefore, all free accessible cells have been covered. An example of a BSA algorithm path can be seen below in Fig. 2. The color of the cells denotes the simple regions and shows the spiral path procedure taken. The dotted lines show the back-tracking mechanism.

After extensive research and reviewing the specifications set by JPL, which are as follows:

- CleanBot 3000 shall be able to locate its position in the cleanroom.
- CleanBot 3000 shall be programmable with "no-go" areas that 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 from each other.

The navigation team concluded that the BSA algorithm would be the most suitable for the CleanBot 3000. This is because it computes its critical points based on its sensor data. Therefore, it is an independent algorithm. It also has a low computational cost, which allows for a lower demand on the processing unit. Furthermore, it has a fine-grain model, which means the environment is broken up into fine cells to have finer accuracy for coverage. This allows for coverage between sections of cells. Lastly, the most critical benefit is that the algorithm takes note of the area it still needs to cover by leaving a marker. As a result, it knows to return to it once it completes each simple region it is cleaning.

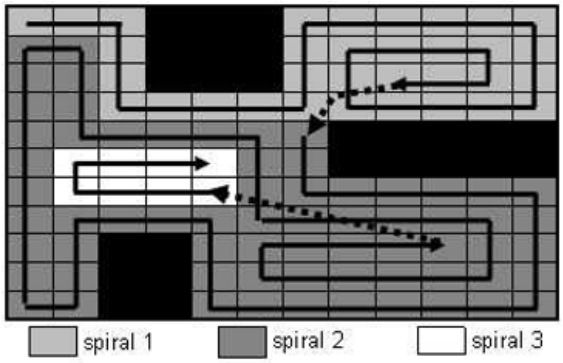

Fig. 2: Example of the BSA algorithm [3]

2.4 Implementation

2.4.1 ROS

This semester, the navigation team continued to use ROS to simulate the CleanBot. ROS is an open-source platform

that integrates all the necessary components into a package that simplifies the build and control of a robotic machine [2]. Furthermore, it also provides the services expected from an operating system, including hardware abstraction, low-level device control, implementation of standard functionality, and more [4]. As a result, ROS simplified the need to develop many tasks and expedited the progress the team could make throughout the semester. This semester the team used the ROS Noetic release on a Linux Ubuntu distro. This is the latest release of ROS and continues to receive long-term support. Additionally, an open-source ROS repository by Tim Clephas and Cesar Lopez was found that provides a CPP implementation of BSA [5]. This repository was used as a stepping-stone to expand on and meet the requirements from JPL.

2.4.1 Running SLAM

The first step towards traversing a robot's environment and using a coverage path planning algorithm is to create the environment using sensor data. To achieve 2D mapping of the environment, the CleanBot will utilize a lidar-based simultaneous localization and mapping (SLAM) algorithm. A LiDAR-based SLAM system uses a lidar sensor paired with an onboard IMU in one dimension [5]. A LiDAR sensor measures the time it takes for the UV light to transmit, hit a surface, reflect, and get received by the receiver. Using this time measurement, we can calculate the distance between emitting and receiving the signal. Equation 1 below specifies how to calculate the distance.[5]

$$d = c * \frac{t}{2}$$

Eq. 1: Distance formula using t (time) and c (speed of light)

As the CleanBot traverses, with the LiDAR sensor mounted to the top of the chassis, the sensor will spin with a 360-degree rotation while transmitting and receiving points. The SLAM algorithm then stores the points in memory for mapping.
While implementing the SLAM algorithm, we used ROS RVIZ to display our sensor data and an RP-Lidar A1 for our lidar sensor. The team found an open-source repository on Hector SLAM to run our RPLidar A1. Using RVIZ, the Mapping and Navigation team ran SLAM, in which the simulation used its sensor data to construct and update a map of its environment in real time. Multiple tests were performed on the CleanBot traversing, and the sensor data was used to construct and update a map of its environment in real-time. The results can be found in Fig. 3. The SLAM data of the 2d environment will then be utilized with the BSA algorithm to divide the environment into simple regions specified above. The Cleambot will identify any obstacles it may encounter, such as flight hardware, as it traverses the JPL cleanrooms.

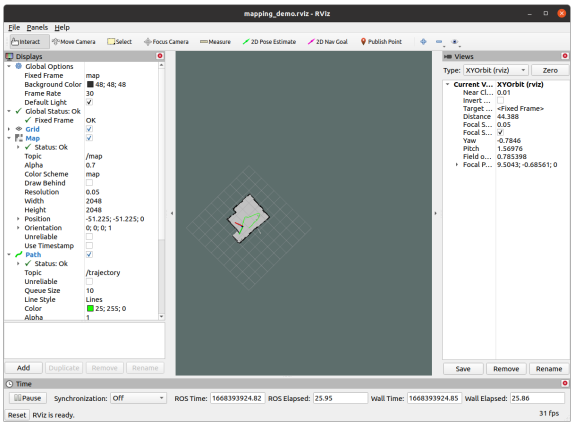

Fig 3. SLAM 2D environment of team member's room
[View enlarged image in Appendix]

2.4.2 BSA Implementation

The navigation team used an open-source code to implicate the BSA algorithm on ROS. First, the BSA needed to be cloned from the GitHub repository and built on a workspace on ROS to simulate

ROS. Afterward, two dependencies needed to be built for the BSA to work appropriately: the tracking_pid and the mobile_robot_simulator [6]. The tracking_pid dependency allows for accurate trajectory following using a variable PID control loop [7]. The mobile_robot_simulator dependency provides two nodes that simulate a laser scanner, in this case, the LiDAR, and a mobile base for simulating the robot's mobility on ROS [8]. For the mobile_robot_simulator dependency, there were no repositories, so the CMake file needed to be constructed, followed by installing all libraries and dependencies for proper functionality. The results of the BSA algorithm can be seen in Fig. 4. The blue lines represent the path the bot has covered, and the green lines represent the path left to be covered by the robot. The bot does not touch the walls due to the cost map parameter inflation radius being set to 1 meter to accomplish the JPL requirement of avoiding flight hardware [2].

3. Marker Localization / Systems

3.1 Previous Work

The Systems team was created to integrate CleanBot's hardware and software. Initially, the focus was on developing the ArUco Markers so that the CleanBot could avoid flight hardware in JPL's cleanrooms. In the Fall of 2021, the team successfully mapped the ArUco markers with a Raspberry Pi camera. In addition, marker recognition was further improved to work with greater distances and in low-level light environments.

In Spring 2022, the Systems team expanded the sensor network by including the Zynq 7000 SoC. The Zynq 7000 SoC was integrated with the Raspberry Pi and serially connected via UART. The connection allowed the Raspberry Pi to send camera data for processing to the Zynq 7000 to plan a path for movement using a marker localization algorithm. Again, there was no specific reason for using these components other than the team's familiarity with their means of function.

The Raspberry Pi and Zynq 7000 SoC would help to monitor the temperature, UV light, and battery levels at all times. While trying to incorporate the temperature data, the team needed a sensor with IP and software drivers compatible with the Zybo Board. Relevant PMODs with existing IPs and software drivers were researched, and two were found to be appealing: Digilent PmodTMP3 and TE Connectivity TSYS01. The Digilent PmodTMP3 was chosen because there was more documentation on using this component on the Zybo board.

The team used Vivado to create a block design and generate a bitstream from it to help implement/configure the temperature sensor using the Zybo Z7-10. As a result, the Zybo Z7-10 successfully received temperature data and displayed it on the UART serial console.

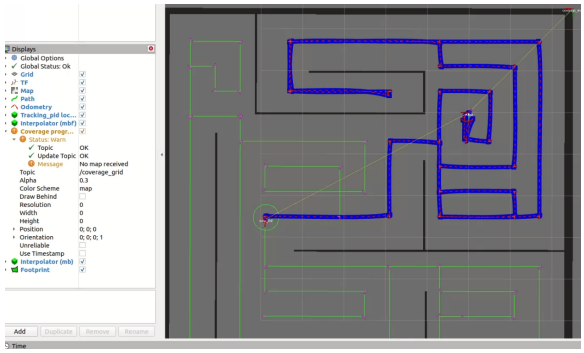

Fig. 4 Results of running the BSA algorithm on ROS
[View enlarged image in Appendix]

3.2 Plan of Action

During the Fall semester of 2022, the Systems team decided to use the Zynq 7000 SoC on the Zybo Z7-10 board without the Raspberry Pi. Last semester, the idea was to use the Zybo Z7-10 in conjunction with the Raspberry Pi, but through further research, the

team has found that the Zybo Z7-10 can perform the Raspberry Pi's data capture as well as the data processing. The data involved in this capture include the image from the Raspberry Pi camera and LiDAR readings. In addition, the simplification will decrease the complexity of the design and will draw less power from the limited CleanBot battery banks.

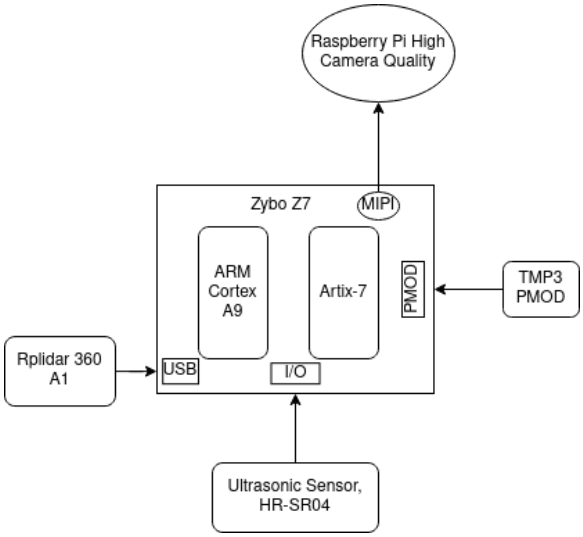

Fig 5: Zybo Z7 System

The Zybo Z7-10 can perform the Raspberry Pi's data capture because it has a MIPI port to use the Raspberry Pi camera and a USB port to connect to the LiDAR sensor. This semester the team focused on using the Raspberry Pi camera using the MIPI port on the Zybo Z7-10. The Systems team currently has a functional block design based on Adam Taylor's "Building a Camera / Imager Test Platform" project on the Diligent site. This project was deemed a good base design due to its scalability. If more visual input was required in the future, this design could account for that.

With the design completed and exported to Vitis, the team plans on adding the Raspberry Pi camera's I2C connections to the C software before testing the camera's picture quality and making corresponding adjustments to the code. The team needed help updating the IP design due to the multiple custom IPs and the switch from Xilinx SDK to Vitis. Since the custom IPs could not be tampered with, the Diligent IP blocks used in the design are outdated but create a functional design. In the future, the team would like to create a more updated design by examining the custom IPs and creating new ones to correspond with the updated Diligent IP blocks.

3.3 Research

This semester's research centered on how FPGA programming works and how the team can implement Raspberry Pi's functionality on the Zybo Z7-10 board. The systems team used the Vivado program from the Xilinx design suite when programming the board hardware. Within the same design suite, Vitis was used to create an application that makes the Zybo board software. The next step was connecting a Raspberry Pi camera and LiDar to the Zybo Board.

The team focused on Adam Taylor's project mentioned in "Building a Camera/Imager Test Platform" as a base to connect the Raspberry Pi camera to the Zybo Z7-10. In the project, the HDMI input and MIPI camera interface are used. While the camera used in the project was a PCam, the systems team plans on changing the I2C connections of the camera in order to use the Raspberry Pi camera. MIPI is an abbreviation for Mobile Industry Processor Interface. The MIPI interface can transfer high data at high bandwidths while using high-speed serial lanes[10]. For example, the CSI-2, Camera Serial Interface issue 2, transfers all the image data in this project. Connecting the MIPI camera to the Zybo board interface requires two MIPI lanes to transfer images. Then, the imager can be configured using the I2C connections. Depending on the camera used, the team will need to configure which pin will be used to make the camera and board

connection. In the project, the PCam only needs a GPIO in pin11 to get a signal to the imager[10]. The FPGA design has two input video connections and one video output path. The HDMI input will be connected to J9, and the MIPI connection will be connected to J2[10]. The HDMI output is connected to J8[10].

This design requires software to configure the image processing in the FPGA and the chosen camera, PCam. Within the software, the I2C is used to configure the PCam connection. A register will be read, and if the correct data is not inside it, then the PCam is not connected. In this project register, 0x3100 is read with a value of 0x78 (I2C address), and when that value is not received, it can be said there is no connection[10]. Information will be read and sent to the PCam, then switched between the inputs using the AXIS switch.

3.4 Raspberry Pi Cam Implementation

To get an efficient system, the Systems team wanted to connect the Raspberry Pi camera straight to the Zybo Z7-10. A Zybo Z7-10 board was chosen to perform this task as it is a system-on-chip board. The Zybo Z7-10 has a dual-core ARM Cortex-A9 processor and a Xilinx 7-series FPGA logic. With the ARM processor, the team can connect multiple peripherals and control them as one would with a Raspberry Pi. In addition, the FPGA logic and the processor allow for fast data processing through parallel processing.

In other designs, such as the TMP3 senor connection, Pmods connect a peripheral to the Zybo Z7-10. They are typically used in designs featuring the Zybo because Pmods have detailed documentation from Digilent, thus making them easy to implement. In addition, peripherals such as an ultrasonic sensor can be implemented much faster if a Pmod is found that can fit the requirements needed. Unfortunately, the camera the team had access to was the Raspberry Pi Camera, which will be harder to implement since it utilizes a MIPI connection through a port on the Zybo. Using a MIPI connection requires mapping the I2C connection within the software to communicate properly with the board, thus making Pmods easier to use.

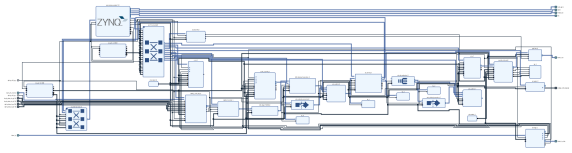

Fig. 6: Finished Block Design
[View enlarged image in Appendix]

To connect the camera to a monitor through Zybo, the Systems team used multiple IP blocks provided through a project published by Digilent. These IPs include processor system resets, AXI interconnects, a ZYNQ7 processing system, video timing controllers, a dynamic clock generator, an AXI Bayer to RGB, AXI Gamma Correction, AXI4-stream subset converters, AXI4-stream switches, an AXI Video Direct Memory Access, DMA, an AXI4 stream broadcaster, AXI4-stream to video out, AXI GPIO, RGB to DVI video encoder, and DVI to RGB video decoder. This design, made in Vivado, contains an HDMI and MIPI input connection. Through custom IPs created by the given project, the MIPI input was made into data that can be gamma corrected and streamed into the HDMI output. The custom IPs used in the design are the MIPI CSI receiver and the MIPI D physical receiver. These IPs in the block design can be used but are blocked from changing and viewing the design details. The final design allows a video stream from the Raspberry Pi Cam to the Zybo Z7-10 through MIPI, processed and outputted to the Zybo Z7-10's HDMI TX.

The Systems team made a Vitis application controlled through C files to control the hardware created. The outdated

SDK project provided by Diligent was utilized to recreate the application as a Vitis project, the most recent software platform used to control the hardware created on Vivado. This new project was based on the XSA file created by generating the bitstream from the block design pictured above. The files transferred from the original SDK project were sorted to determine which files were necessary for implementing the application in Vitis. The C files that were imported to the new Vitis application were the files that control the clock, I2C connections, video resolution, and camera detection.

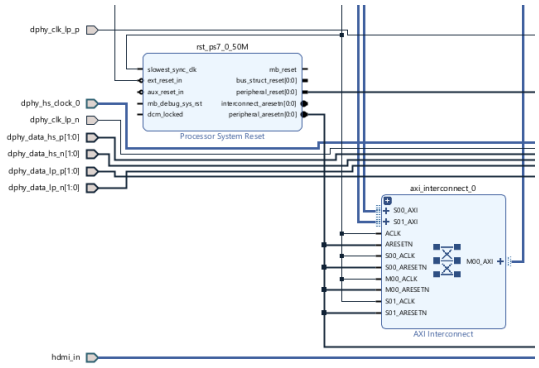

Fig. 7: Inputs of the Design
[View enlarged image in Appendix]

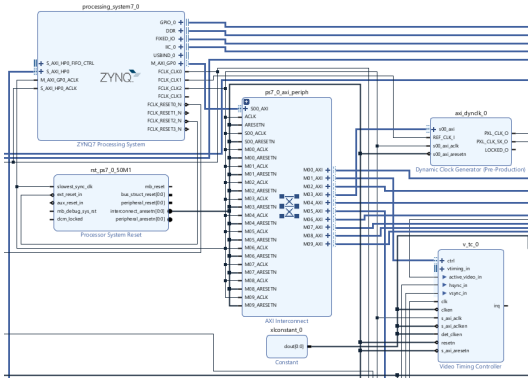

Fig. 8: Main Processing Section of the Design
[View enlarged image in Appendix]

The team built the project on Vitis by editing six different Makefiles that regarded the two custom IPs used in the project to receive the MIPI connection. These makefiles were located in the hardware drivers directory, cortex library sources folder, and the cortex library sources located in the Zynq platform folder. These changes guarantee that the build will work regardless of the user's operating system.

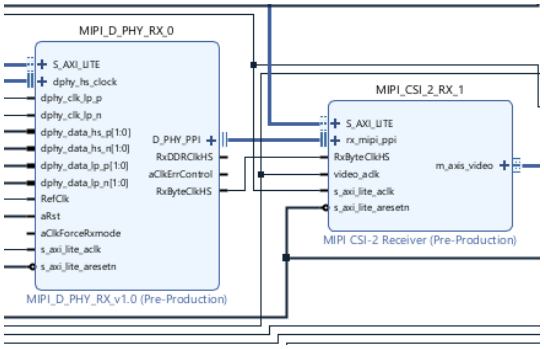

Fig. 9: Custom IPs Used

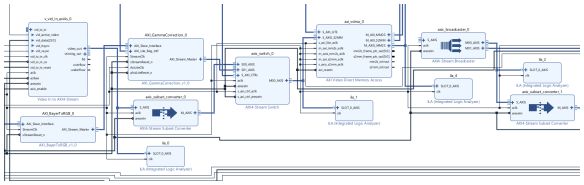

Fig. 10: Video Processing and Logic Checkers
[View enlarged image in Appendix]

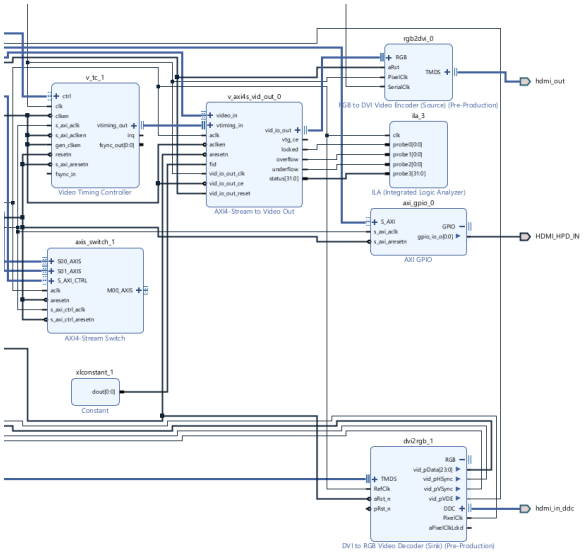

Fig. 11: Streaming Output Data
[View enlarged image in Appendix]

4. Power & Mechanical Design

4.1 Previous Work

In Spring 2022, our team continued researching essential components required to complete our CleanBot. Unfortunately, the team again received a message from JPL that funding was unavailable. However, the team remained positive and began focusing on creating an alternative solution to turning our CleanBot research into a reality. The Sanitation team received the dosimetry test from JPL but encountered an issue where only one of the nine Klaran Light Engine UV-C LED modules worked, resulting in a travel time of ten meters per hour. Despite the minor setback, the team continued to do research based on the light engine modules which can be implemented for CleanBot's applications in the future. The Solidworks team decided to design a model of a three-wheel design to improve maneuverability and reviewed multiple components implemented from the previous semester. The mechanical team looked at brushless DC motors that offer the best performance and high levels of energy efficiency. Moreover, the team concluded to utilize Polyurethane wheels in our design because it allows us to maintain minimal amounts of bacteria, dust, and dirt build-up throughout CleanBot's operation times.

4.2 Research Plan of Action

For the Fall 2022 semester, the Power and Mechanical team focused on extending our research and transitioning into a prototyping phase to finalize CleanBot's components. All sub-teams maintained the growth mindset while continuing our progress to find quality and efficient components to implement in our CleanBot. The team hopes to receive funding to complete our build-into-production in future semesters.

At the beginning of the semester, the team continued their research in 3D printed design to better visualize the placements of the components and a realistic depiction of the final build of CleanBot. However, once the team was notified that they were meeting the JPL team on November 1st, the Power and Mechanical team decided to eliminate the 3D printing idea because of the potential difficulties it can pose when the 3D printed material interacts at specific temperatures. One example is if the CleanBot is left in the car during transportation during hot weather, then it could deform and warp.

The Mechanical team continued redesigning the chassis and implementing a three-wheel system, and maintained the idea of incorporating polyurethane wheels to the design and a caster ball wheel.

4.2.2 Prototype Plan of Action

During the mid-semester, we were informed that the JPL in-person tour was approved. Once the team was informed, they began purchasing items out of pocket to create a working prototype within six days. Unfortunately, due to the six-day restriction, we had to rely on Amazon, all-electronics, and any stores that would allow the team to complete our prototype.

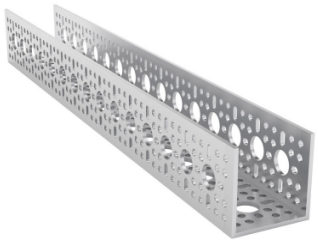

Fig. 12: goBilda Aluminum 1120 U-Channel

The team found a modular chassis made out of Aluminum on ServoCity and built it on top of having our CleanBot remain as modular as possible for future semesters to add and modify any components without setting back progress, as shown in Fig. 12 and 13.

During our prototyping phase, the team encountered some challenges not encountered

during the research phase. The Power team began utilizing FuseBox to protect from destroying our electrical components since it was cheaper to replace fuses than the components themselves. The team also utilized Buck and Boost converters to provide the correct voltage levels to specific components and motors for extra safety measures. With the prototype complete, the team can transition into a modular system that could be used for the final build.

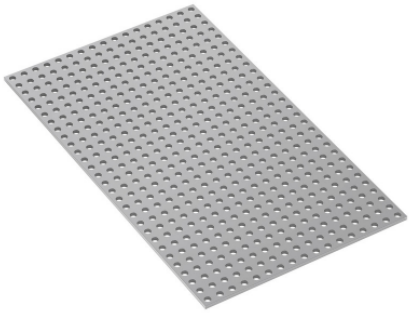

Fig. 13: goBilda Aluminum 1116 Grid Plate

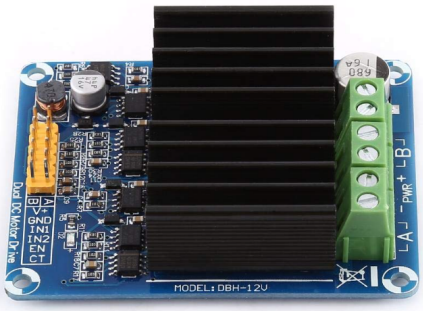

Fig. 14: DBH-12 Motor

4.3 Power Systems Design

4.3.1 Batteries/Charging

This semester, the batteries and charging team focused on simplifying the design from last semester. The previous semester's design used eight NH3054QE34 batteries from Inspired Energy. Each battery provided 6.8Ah and 14.4V and was to be connected in parallel for use in the CleanBot, which would have provided a total of 783.36Wh. To simplify connections, it was recommended that the team use only one battery rather than eight batteries in parallel. The Varta EasyBlade 24 battery was selected as a potential solution and is shown below in Fig. 15.

Fig. 15: Varta Easy Blade 24 Battery

This 25.9V, 58Ah battery provides 1502Wh, allowing the robot a longer run time than the previous battery configuration. The battery also has short circuit protection, over-discharge current protection, and overcharge current protection, as well as its IEC62133-2:2017 and UN28.3 certifications [9]. The battery also has active cooling and does not require external battery management. The battery is currently pending approval by JPL. However, the team believes it is a good substitute for the Inspired Energy battery configuration. The charger used with our selected battery is the Easy Charger 24, shown in Fig. 16, which includes active cooling and a sealed aluminum die-cast enclosure that will protect against shock, dirt, vibration, and fluids.

The charger's algorithms will charge the battery precisely and balance the charge time and life. In addition, the charger has its UL 1564, EN 60335-2-29, and AZ/NZS 60335 (RCM) certifications to ensure the safe operation of the charger in its use [9]. Varta is a well-established company based out of Germany and has been making batteries since 1904. The battery model is also relatively

newer, meaning that support for the battery shall continue well into the future.

Fig. 16: Varta Easy Blade 24 Charger

4.3.2 Prototype

This semester, the first physical prototype of the CleanBot was constructed. The prototype used a 12V 10Ah LIFEPO4 battery, an Arduino MEGA 2560, a 5-12V dual channel H-bridge controller board module from Oumefar, and a 6-way 12V fuse box to power and control the 775 DC motors. A buck converter from Songhe was used to step the 12V from the battery down to 9V to power the Arduino. Sonar was also connected and incorporated into the code of the Arduino so that the CleanBot would stop if an object were detected within 45cm of the Sonar. Finally, the components were wired according to the circuit diagram shown below in Fig. 17.

The motor controller used a PWM to control the speed of the motors, and the Arduino was programmed to accept a signal through the ia6b receiver so that the robot could be controlled remotely using the Flysky FS i6x remote control. The robot used variable speed control to steer and change direction, and the Sonar would send a signal to the Arduino to set the motor speed to zero if there was an object within the set detection distance at any point during the operation of the CleanBot. The 12V DC motors were drawing a combined 25A on start-up, so the H-bridge needed to handle this inrush of current. A 30A fuse was connected to the motor controller to protect the device, while a 3A fuse was used for the buck converter. UV-A LED light strips were attached to the bottom of the prototype to simulate the UV-C light modules used for sanitation in the final design.

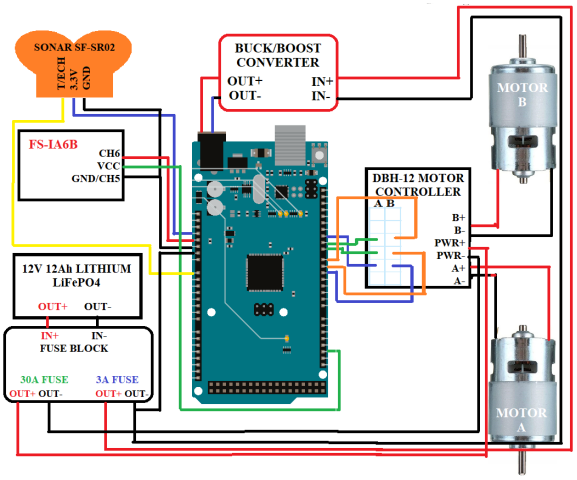

Fig. 17: CleanBot 3000 Prototype Wiring Diagram
[View enlarged image in Appendix]

4.4 Mechanical Design

4.4.1 Solidworks

Following the steps from the previous semester's design, this time, the team decided to give minor adjustments to the design since various components changed, like motors and batteries, the Zybo board, LiDAR, and Sonar, had to be moved just a little to accommodate the change. A visualization of the changes can be seen in Fig. 18.

Fig. 18: SolidWorks Model: CleanBot 3000(Isometric View)

The reasoning behind this design is all about the modularity, functionality, and maneuverability of the CleanBot inside the JPL Cleanroom. Starting with the three-wheel design, with two motorized wheels on the rear, the robot can make sharper turns, clean in more spaces, and traverse the room more efficiently. To better visualize this concept, imagine a car on a two-lane road; if the driver decided to make a U-turn, it would then have to perform at least three movements to make the sharp U-turn. One option to make the CleanBot more maneuverable is to add rear steering to the car, but it would require more moving parts and possibly space for error. Now a three-wheel design uses two powered rear wheels and one more wheel at the front for support reasons; if one of the motorized wheels turns to the front while the other turns backward, this will allow the robot to make 360º turns in any direction on its axis. With this in mind, going back to the car example, if the robot needed to make a U-turn, it would do it in just one movement.

The distribution of CleanBot's components focused on weight distribution, user-accessible components, and usability of each component. Electric motors, as well as the battery, account for the most weight, so they are placed on the first floor of the CleanBot, so the center of gravity is lower, preventing the robot from flipping on its side while making turns; fans are also placed on this level, for better cooling of the robot since the battery and motors are the ones that produce the most heat, adding channels to the floor of the second level will allow the air to flow from the side grills located on the second floor, to be pulled in and flow through the Zybo board and cool its components flowing down into the first floor where again mixes with cool air from side channels and then flowing out of the CleanBot.

Lastly, the cover of the CleanBot has a hole on the top for the LiDAR, which needs to be on the top to get a clean scan of the room and its obstacles, combined with the Sonar window located at the front of the second floor will allow the CleanBot to not run into walls, people and most importantly into equipment.

As in previous semesters, the material used for the model is still 80/20 Aluminum, which is the required material from JPL because of its durability, and thermal and electrostatic properties.

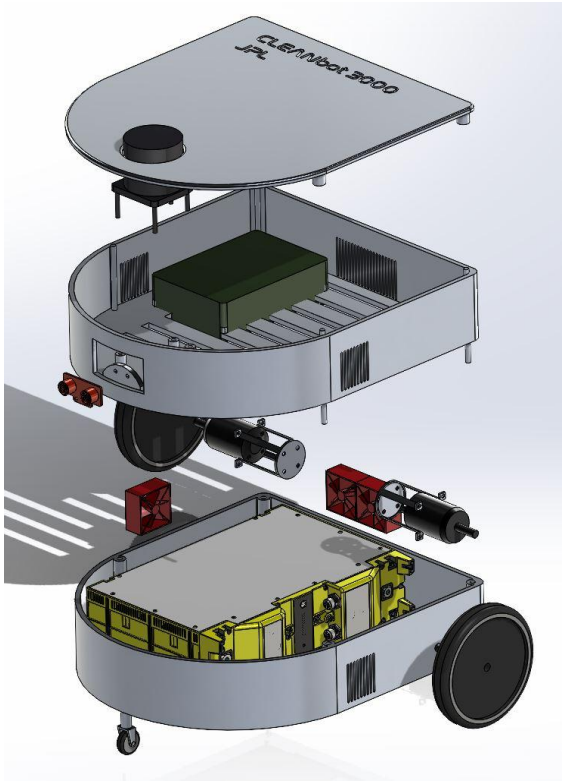

Fig. 19: SolidWorks Model: CleanBot 3000 (Exploded View)

4.4.2 Prototype

For this semester, the team decided to build a prototype of CleanBot. The idea was to construct a model with, although not the same model, approximate dimensions as the SolidWorks model. It started with the elemental design foundations of the 3D model, a 3-wheel design where the back wheels are powered by DC motors, with the addition of a Sonar so the CleanBot can stop if within a specific set distance of an obstacle, as well as a working Lidar scanner to map the environment in which the CleanBot transverses.

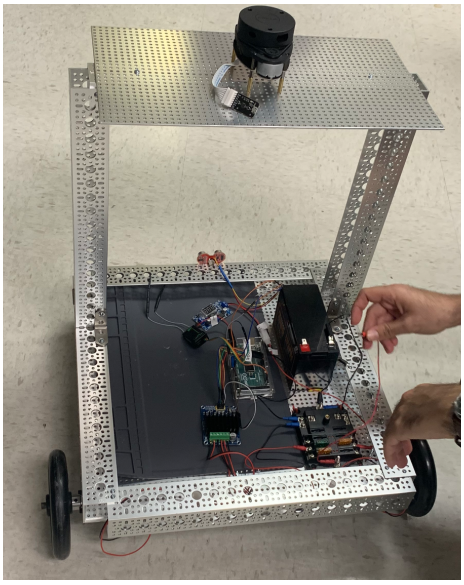

Fig. 20: Fall 22' Prototype Model of CleanBot 3000

As seen in Fig. 20, the prototype was constructed entirely in an aluminum body, with two 6-inch rubber wheels and a caster ball wheel at the front. The addition of a silicon mat prevented the circuit boards from contacting the aluminum and causing a short circuit. The addition of the lidar scanner was at the very top of the CleanBot, so it can easily map the environment without any obstacles that may obstruct a full 360-degree view of the room it traverses. The prototype was completed successfully in a week and was able to be showcased at the JPL tour.

4.4.3 Motors

Fig. 21: 775 DC Motor [12]

For the current prototype, the power team decided to use the 775 DC 12V - 24V Motors because of their torque output. Their controllable speed and power using an Arduino, and more importantly because of the documentation, which is the keystone to any successful project, in this case, the prototype of CleanBot.

The team also encountered minor errors while testing the motors in the prototype, where it seemed that the motors did not have the power to move the robot. The problem was later identified; stall current should be the same as a motor controller's continuous current specification. The future of motors will be focused on the synchronous drive control for the robot, which is the most ideal drive mode; this drive control consists of several wheels, In our case, 3 wheels, that are not only equally spaced but also each wheel is capable of being driven and steered. [13] All the wheels point in the same direction and rotation speed, meaning that the robot only has two degrees of freedom. Nonetheless, its high maneuverability allows the robot to control its orientation directly, as it behaves like a holonomic robot.

4.4.4 Wheels

For this semester's prototype, a pair of 6-inch rubber wheels were used [14], primarily because of the easy access to this wheel type since it is used in walkers and some types of wheelchairs that are widely accessible. In future iterations of the prototype and definitely in the actual CleanBot, it is required to use polyurethane as the material for the wheels because, in the JPL Cleanroom, it is desired not to leave marks on the floor by the rubber from the wheels, the idea is to clean the space, not to leave marks or pollutants. Polyurethane wheels are available to purchase as caster wheels. In addition, it is planned to detach the tire from the wheel hub and create/obtain a custom wheel hub to reduce dust collection and make the robot lighter. [15][16]

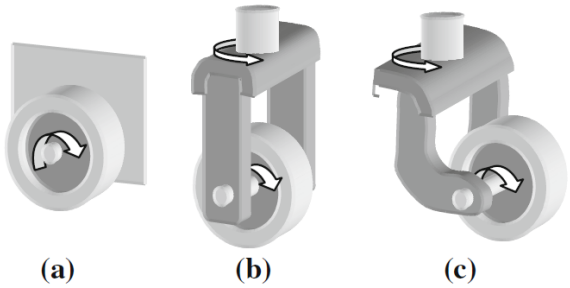

Fig. 22: Conventional wheels: a) Fixed wheel. b) Centered orientable wheel. c) Off-centered orientable wheel. [17]

Another critical decision to make about the wheels is their placement on the CleanBot, as seen in Fig. 22; so far, in the prototype, the team decided to go with fixed wheels. The fixed wheels have two degrees of freedom and can move either backward or forward. These wheels are commonly used as drive wheels on rear-wheel drive robots. [17] These are the most widely used type of wheels in robots since they can be used for steering purposes, just like in CleanBot; by having two independently controlled motors, the robot can 360 degrees of movement. Now, centered orientable wheels have a vertical axle that is directly connected to the wheel, this type of wheel is helpful for the third wheel that goes in the front; it helps in the maneuverability and stability of the CleanBot, but in future prototypes of the robot, this type of wheel placement can be taken into consideration for the back wheels also. Lastly, off-centered wheels have full freedom swivel joints that allow the wheel to rotate freely when the robot is in motion; however, this type of wheel can suffer from wheel flutter, which could lead the robot to instability issues and control problems. [17] Considering this information and the ability to clean the robot from dust or other particles about twice a week by hand, it is very important to consider tight spaces of the wheels and other parts of the CleanBot that will tend to get dirtier with usage.

5. Conclusion

As Fall 2022 is ending, the CleanBot team continues to progress even when the JPL budget is null.

The Mapping and Navigation team continued with last semester's progress and deemed the A* algorithm obsolete. The team then searched for an alternative coverage path planning algorithm correlating with JPL requirements for the CleanBot 3000. The BSA algorithm was then found and was able to be built and simulated successfully. Now that the team has a working CPP algorithm, it can be imported into our prototype model and run in a real-world environment.

This semester the systems team moved toward a more streamlined design using the Zybo Z7-10 board. To begin the implementation of this new layout, the team focused on connecting the Raspberry Pi camera to the FPGA for processing and gathering data. This data will be used with the Navigation team's algorithm to help CleanBot navigate the cleanroom. The team plans on testing the Raspberry Pi camera in the following semesters before incorporating the LiDAR and UltraSonic sensors into the design.

The Power and Mechanical team progressed their research and transitioned into the prototyping phase and the potential final design's detailed digital model on SolidWorks. The battery and charging station had improved from eight batteries in parallel to a singular battery that offers more fail-safe features.

6. Appendix

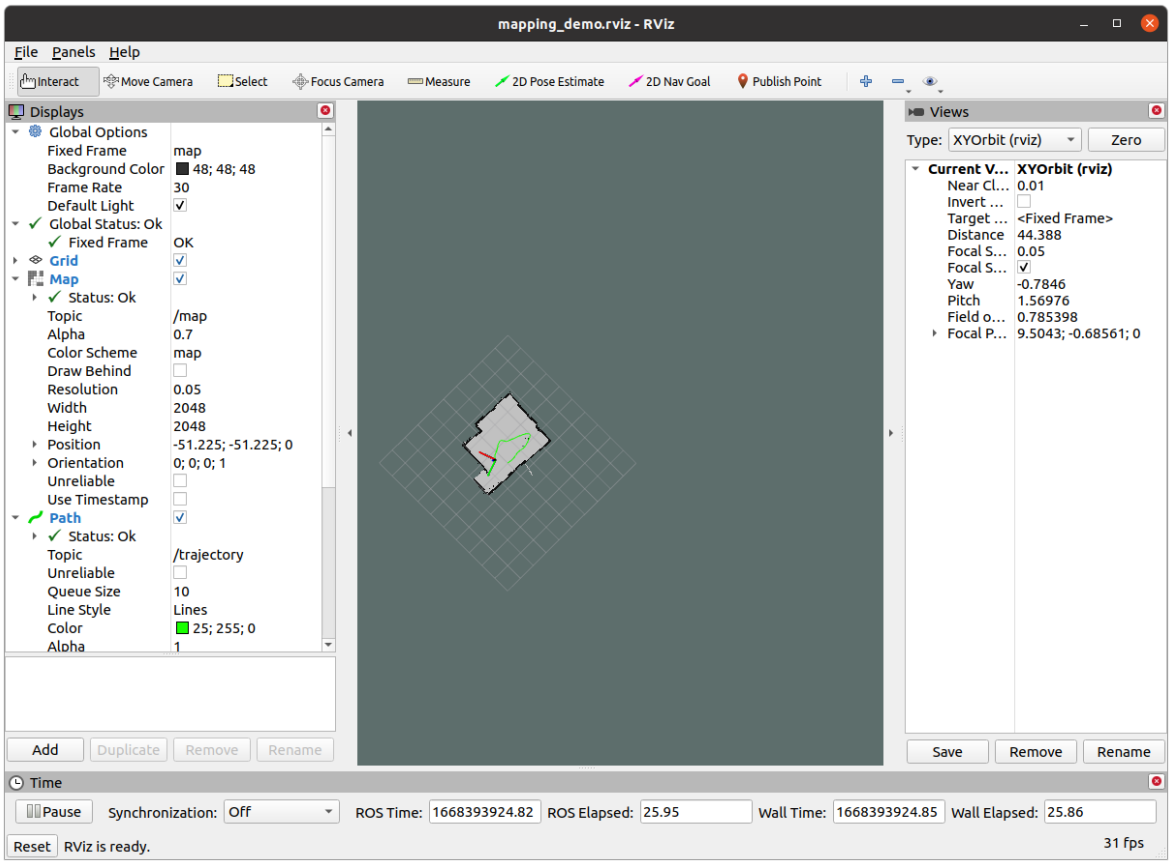

Fig 3. SLAM 2D environment of team member's room

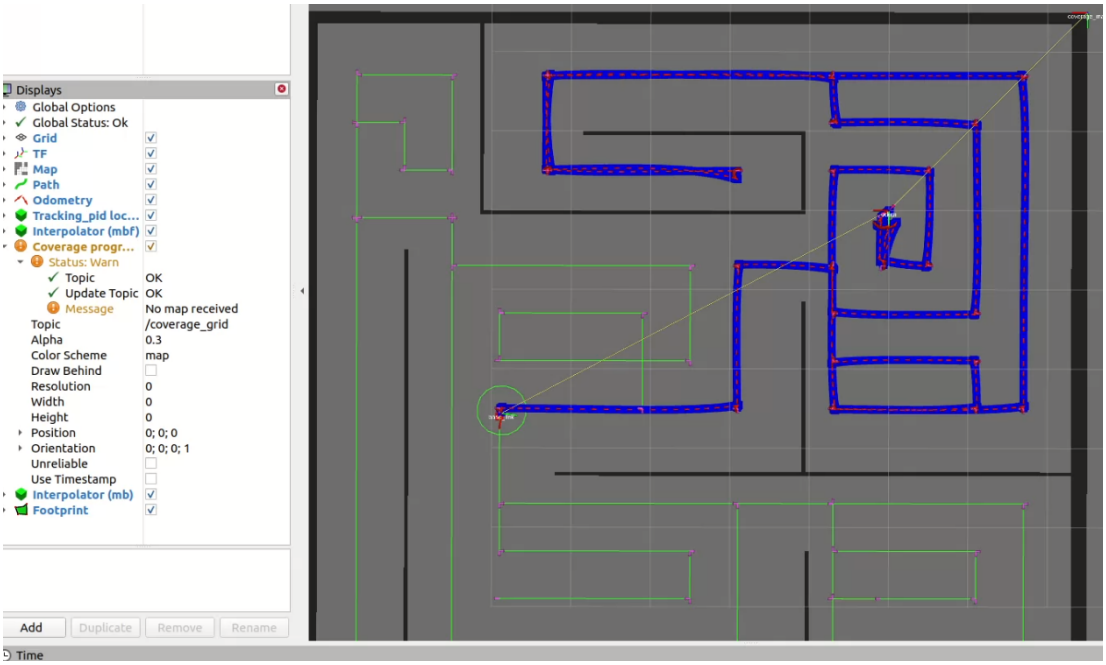

Fig. 4 Results of running the BSA algorithm on ROS

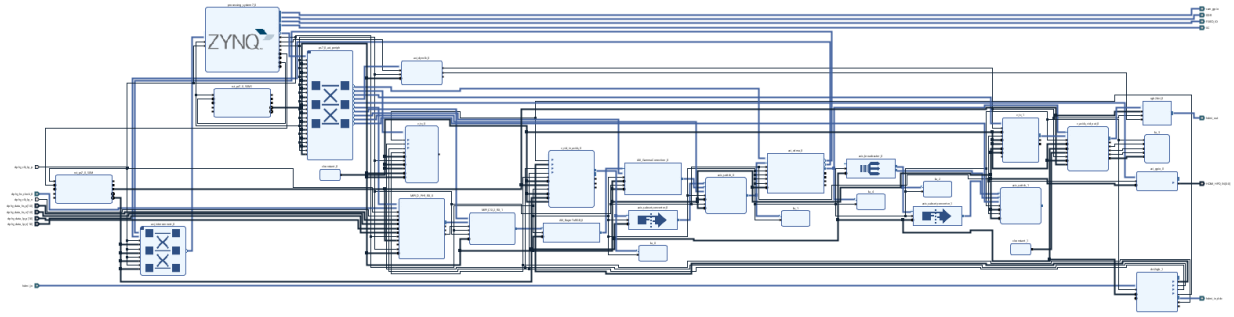

Fig. 6: Finished Block Design

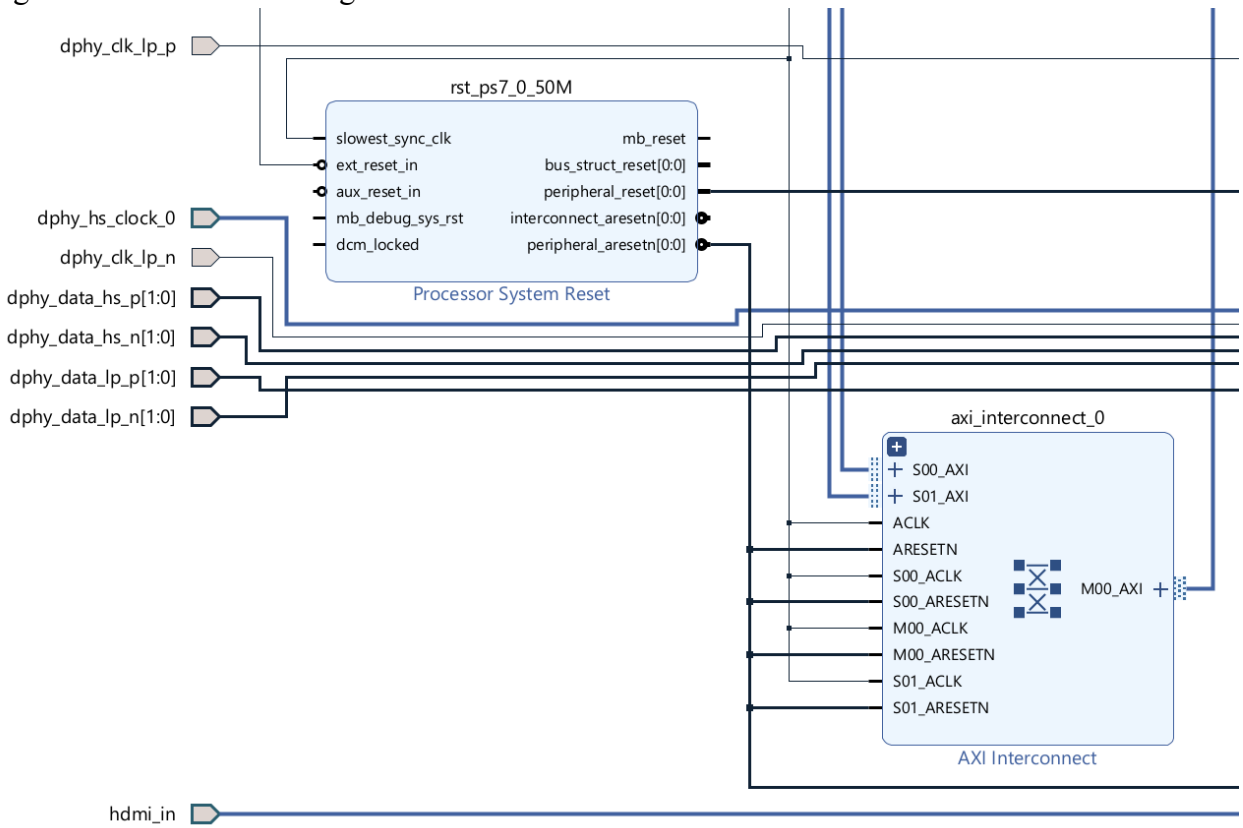

Fig. 7: Inputs of the Design

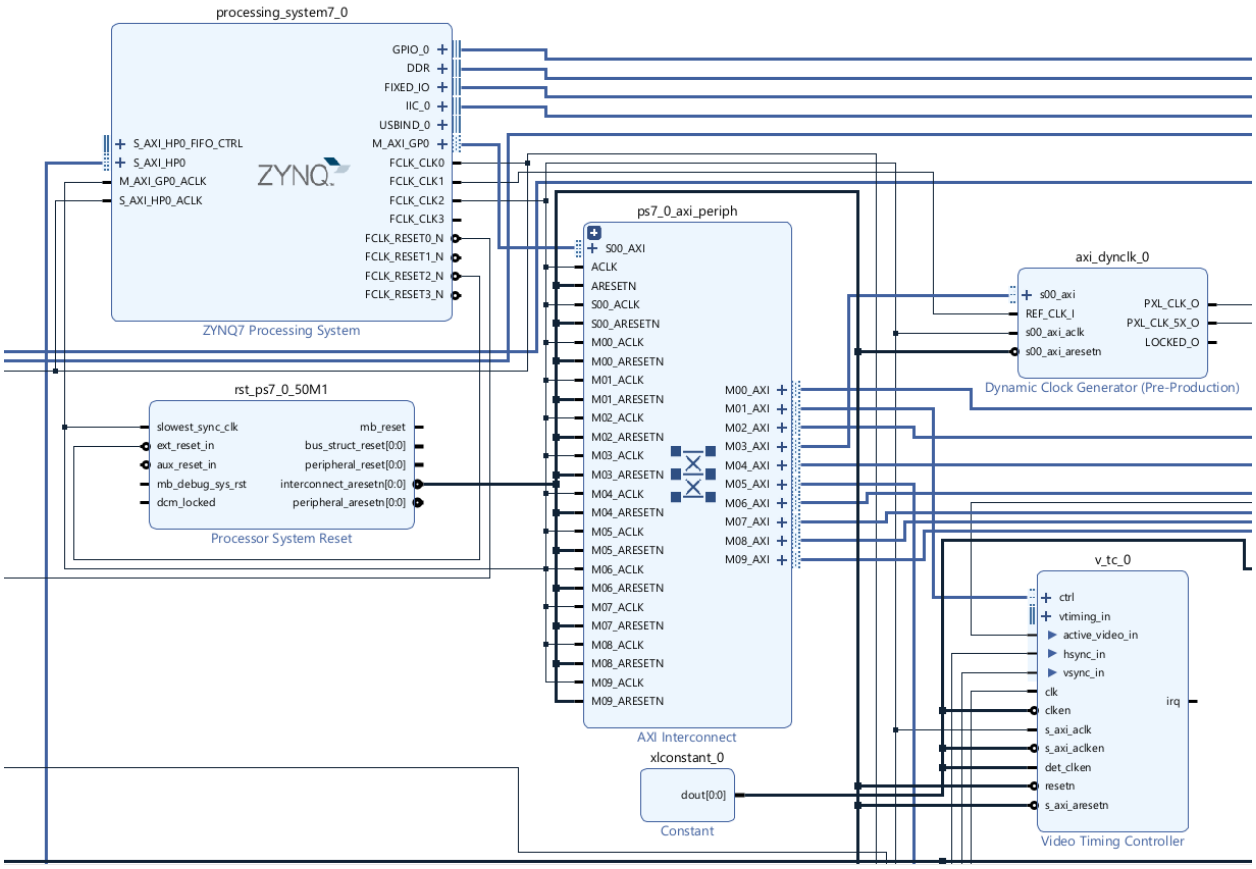

Fig. 8: Main Processing Section of the Design

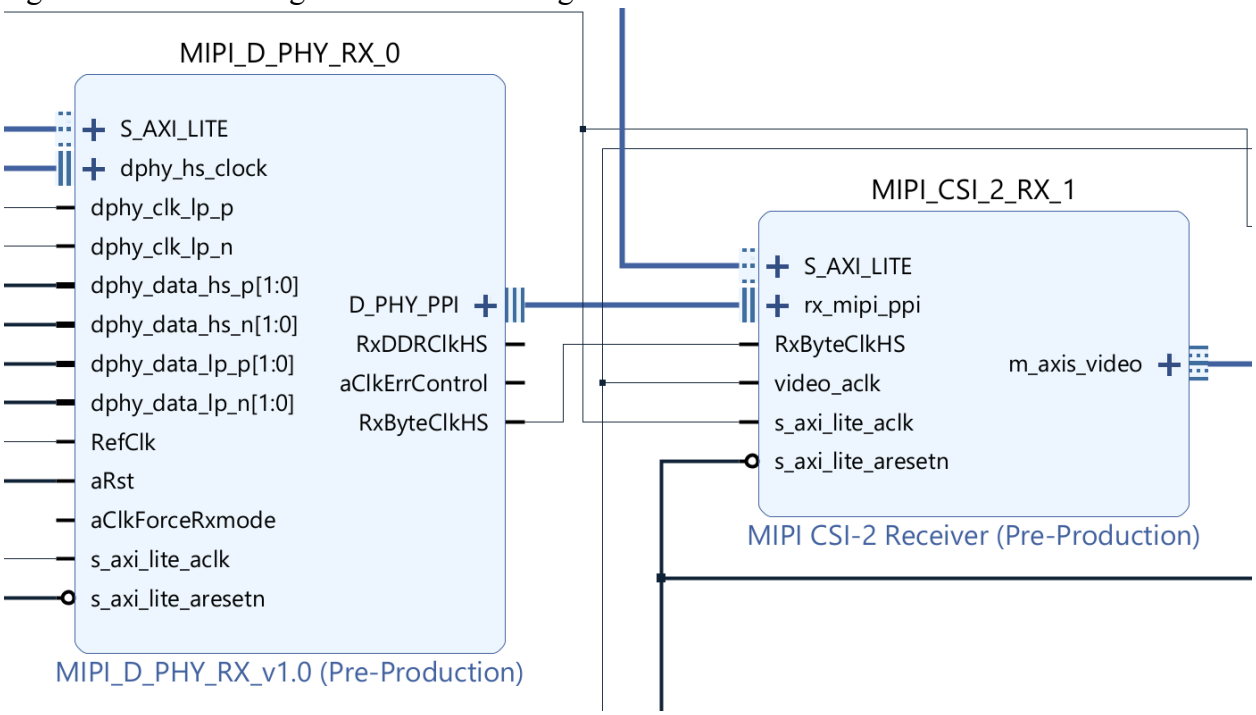

Fig. 9: Custom IPs Used

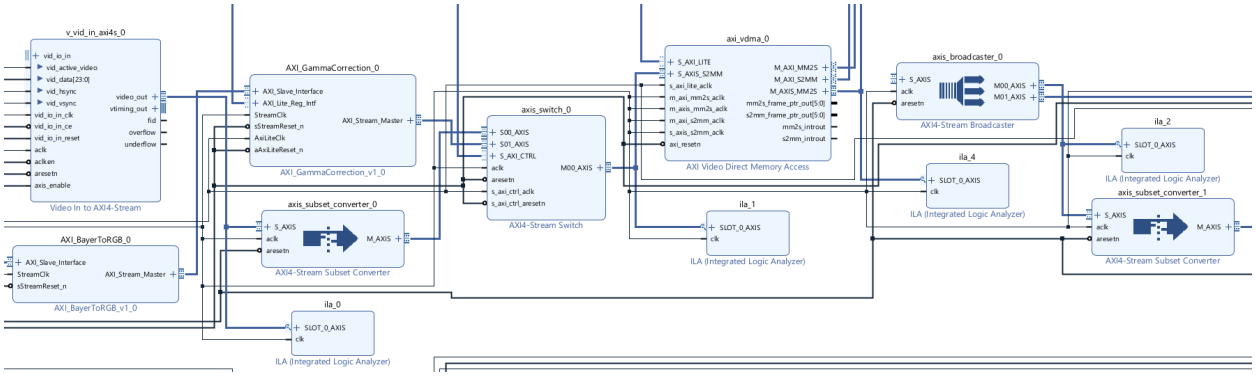

Fig. 10: Video Processing and Logic Checkers

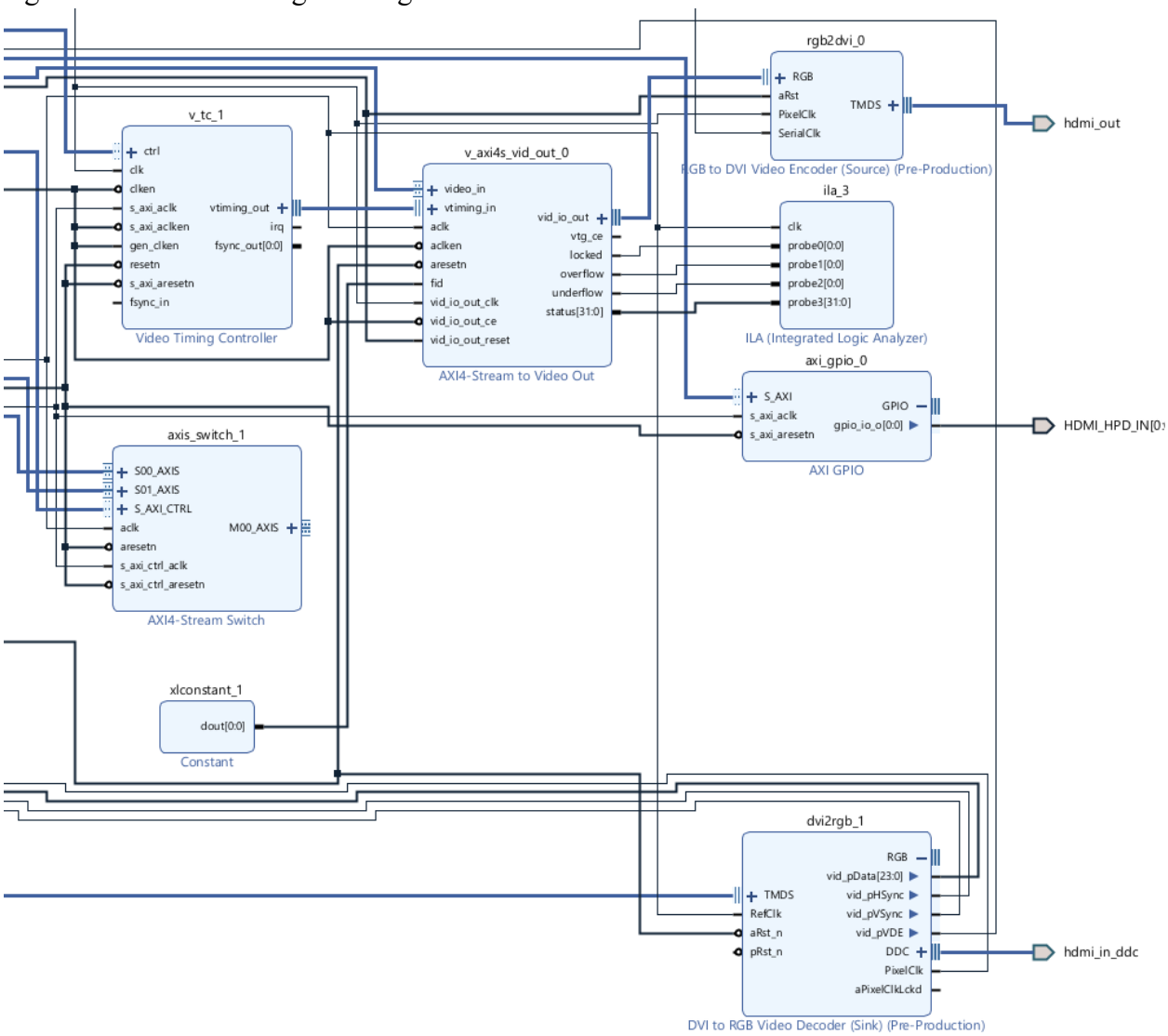

Fig. 11: Streaming Output Data

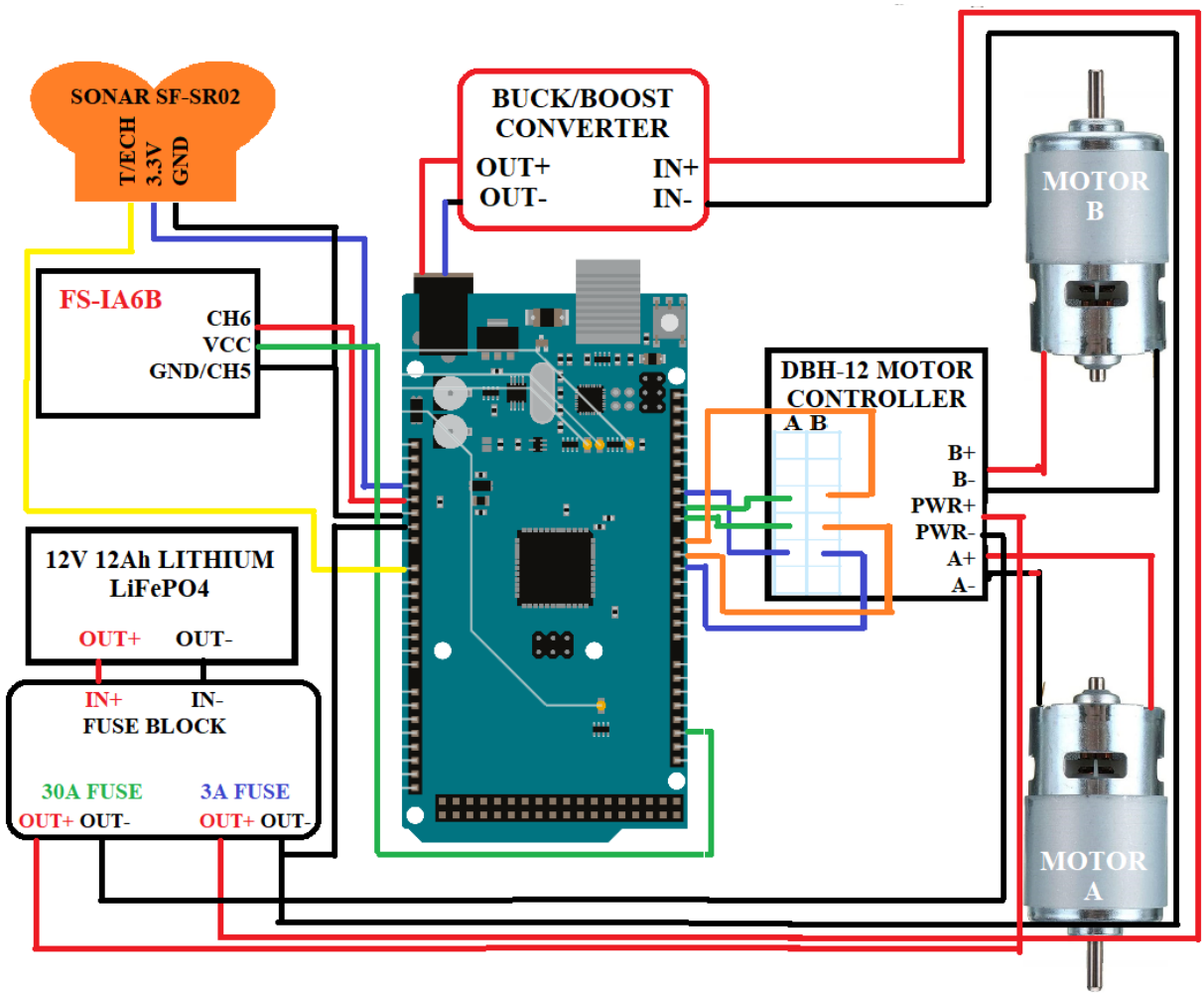

Fig. 17: CleanBot 3000 Prototype Wiring Diagram

References

[1] B. Nasirian, M. Mehrandezh, and F. Janabi-Sharifi, "Efficient coverage path planning for mobile disinfecting robots using graph-based representation of environment," *Frontiers*, 01-Jan-1AD. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.624333/full. [Accessed: 13-Nov-2022].

[2] O. Dabach, M. Valdez, J. Reyes, R. Aanan, and J. Flores, "CleanBot 3000" California State University Northridge, Northridge, CA, tech., 2022.

[3] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "BSA: A Complete Coverage Algorithm," *Shibboleth authentication request*, Apr-2005. [Online]. Available: https://ieeexplore-ieee-org.libproxy.csun.edu/stamp/stamp.jsp?tp=&arnumber=1570413. [Accessed: 13-Nov-2022].

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

[5] C. Pao, "VSLAM and LIDAR in Robotic Navigation," *CEVA's Experts blog*, 10-Nov-2022. [Online]. Available: https://www.ceva-dsp.com/ourblog/how-are-visual-slam-and-lidar-used-in-robotic-navigation/. [Accessed: 13-Nov-2022].

[6] T. Clephas and C. López, Eds., "nobleo / full_coverage_path_planner," *GitHub.com*. https://github.com/nobleo/full_coverage_path_planner

[7]T. Clephas and C. López, Eds., "nobleo / tracking_pid," *GitHub.com*. https://github.com/nobleo/tracking_pid/

[8]M. Hansen, Ed., "mrath / mobile_robot_simulator," *GitHub.com*. https://github.com/mrath/mobile_robot_simulator

[9] "Easy blade - can-bus enabled modular battery with over 1500WH per module," VARTA. [Online]. Available: https://www.varta-ag.com/en/industry/product-solutions/lithium-ion-battery-packs/asb/easy-blade. [Accessed: 13-Nov-2022].

[10] Taylor, Adam. "Building a Camera / Imager Test Platform." *Digilent Projects*, 3 Dec. 2018, https://projects.digilentinc.com/adam-taylor/building-a-camera-imager-test-platform-9d9A5d.

[11]"Zybo Z7 Reference Manual - Digilent Reference," *digilent.com*. https://digilent.com/reference/programmable-logic/zybo-z7/reference-manual?redirect=1

[12] "12V DC RS-775 Brushed DC Motor | NFPshop.com." https://nfpshop.com/product/775-dc-motor-voltage-12v-dc-motor-high-speed-big-torque-from-source-factory

[13] Nava RodríguezN. E., *Advanced Mechanics in Robotic Systems*. London: Springer London, 2011.

[14] "Amazon.com: Stander Replacement 6-inch Walker Wheels, Compatible with the EZ Fold-N-Go Walker and the Able Life Space Saver Walker, Black, Set of 2 : Health & Household," *www.amazon.com*. https://www.amazon.com/Stander-Walker-Replacement-Wheels-Space/dp/B00SXP8X9O/ref=sr_1_7?crid=2FM030VPG46S7&keywords=WALKER+WHEELS&qid=1667973144&sprefix=walker+wheels%2Caps%2C145&sr=8-7 (accessed Nov. 09, 2022).

[15] “4” Orange Acrylic Disc Wheel - 2 Pack,” ServoCity. https://www.servocity.com/orange-4-00-precision-disk-wheels/ (accessed Nov. 09, 2022).

[16] “3607 Series Disc Wheel (14mm Bore, 72mm Diameter, Black) - 2 Pack,” ServoCity. https://www.servocity.com/3607-series-disc-wheel-14mm-bore-72mm-diameter-black-2-pack/ (accessed Nov. 09, 2022).

[17] Robert, “Types of Wheels Used in Robotics,” *Robotics Shop*, Jun. 09, 2020. https://roboticsshop.net/types-of-wheels-used-in-robotics/#:~:text=Types%20of%20Robot%20Wheels%201%20Standard%20Fixed%20Wheel (accessed Nov. 09, 2022).