UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

DEPARTMENT OF MECHANICAL AND MECHATRONICS ENGINEERING

MTE 482 - Final Report

TableUV - The Ultimate Autonomous Table-Top Sanitization Robot

Proudly Prepared by:

Group 55

Jianxiang (Jack) Xu
Tsugumi Murata
Jerome Villapando
Dong Jae (Alex) Park

10 November 2022

10 November 2022

Prof. Ayman El-Hag
Prof. Baris Fidan
Prof. James Tung
Prof. Andrew Kennings

Department of Mechanical and Mechatronics Engineering
University of Waterloo
Waterloo, Ontario, Canada

Dear MTE 482 Course Instructors,

This report, entitled ’TableUV - The Ultimate Autonomous Table-Top Sanitization Robot’, was prepared to be submitted as the MTE 482 Final Report. It entails a design process and proposal for our final FYDP idea on a palm-size autonomous UV sanitization robot. This report will go over the final design of the robot and its manufacturing, development, verification, timeline, and final expense breakdown. This report was written entirely by us and has not received any previous academic credit at this or any other institution.

Group 55 would like to give special thanks to everyone that has helped with the project throughout the term, including Professor Fidan, Professor El-Hag, Professor Kennings, and Professor Aucoin. If any questions arise about the project or the report, please feel free to contact any of the members of this group for clarification.

Sincerely,
Jianxiang (Jack) Xu, Tsugumi Murata, Jerome Villapando, Dong Jae (Alex) Park

Jianxiang Xu Tsugumi Murata Jerome Villapando Dong Jae Park

Table of Contents

List of Figures

List of Tables

Summary

The following report outlines the motivation and technical details involved in engineering the project from start to finish. This project's main objective is to provide an autonomous tabletop robot that is cost-effective and small to disinfect surfaces in public areas. With current market solutions being bulky and expensive, we believed that our product would be viable given the current global pandemic. Our robot's prime utilization would be in public surfaces such as tables in restaurants and libraries to ensure consistent cleaning and sterilization of the character from common bacteria and viruses.

The mechanical design involved designing multiple iterations of the chassis CAD model to house all the necessary components that make up the robot. Several modifications were made from the initial design that took into account the positioning of sensors and the manufacturing process of the housing such that the assembly process was simple, easily accessible for debugging and modular for future improvements.

The electrical design involved the manufacturing, testing, and modifying of the designed PCBs from MTE 481. The PCBs were ordered from PCBway, manufactured using a reflow oven and hand soldering, and tested using the power supply and oscilloscope. Hardware malfunctions found during the bench-testing and debugging were fixed by methods such as replacing components on the PCB and adding fly wires. After hardware verification was complete, the team developed firmware for the main MCU, ESP32, and the secondary MCU, 8-bit AVR.

The software design process started from a simulation environment where different mapping and localization algorithms were developed and tested for feasibility. After understanding how the software architecture should be designed, time was spent in implementing and integrating the various sensors and algorithms into the onboard ESP32. Much care was taken in maintaining code readability through an organized structure, and coding guidelines and performance were also maintained by leveraging the dual-core hardware available.

Throughout the whole design process, verification in the hardware and software was performed to ensure that we were meeting the chosen constraints and criteria. In terms of the mechanics and hardware of the robot, performance calculations and physical testings for the robot were conducted for each piece of hardware. For software, the quality of code was maintained by doing frequent code reviews and testing the functionality of the robot before making any pull requests to the main branch. Having reached the end of the term, we verified that Constraint I, Constraint II, Constraint III, Constraint IV, Constraint VI were fully satisfied. Additionally Criterion I, Criterion II, Criterion V were satisfied. Some constraints and criteria were not met due to the inability to access labs for testing sensitization quality and, in others, due to lack of time.

1 Introduction

This pandemic has been a pivotal era in modern history, and not just because of all the adverse events, but because this year has brought to light a problem that has been occurring even before the pandemic. In the US, during 2019, more than 2.8 million people were afflicted with an antibiotic-resistant infection, not including viruses such as HIV and influenza [1]. Unfortunately, as a result, these infections lead to the deaths of 35,000. Our Capstone group has a vision to significantly reduce disease transmission by designing an accessible and economical device for the general public.

1.1 Background

Due to the recent global pandemic, disease transmission becomes a significant concern in terms of disease control. Although masks and a frequent hand-wash habit can effectively reduce the risk of infection, public tables can be a secondary source for disease transmission since the bacteria and virus can remain on the surface alive for more than 2 hours and even days depending on the surface condition, and environment condition [2].

1.2 Needs Assessment

Again, masks and a frequent hand-wash habit can effectively reduce the risk of infection, but joint surfaces can be a significant source of disease transmission as well. Because of this, we want to target reducing transmissions through surfaces, specifically targeting objects in public areas that are meant to be shared with numerous people. This device will be targeted towards institutions with common areas shared by the general public, which have a high risk of accelerating the transmission of diseases. Some examples include restaurant tables, community library tables, classrooms, reception desks, and hospitals. The current solution of disinfection is by manual labour, which somewhat defeats the purpose of the action because it not only puts the cleaner at risk but also makes the cleaner a source of transmission. In addition, the effectiveness of the person's method of disinfection is not controlled nor measured, meaning that cleaning effectiveness is not guaranteed. Moreover, some autonomous solutions cost over \$100,000, which are often bought solely by medical institutions that have the rich budget to stomach that cost [3]. Even if the current autonomous solution were to lower its price, its bulkiness is not suitable for tabletops and will require people to evacuate the room to start disinfecting for safety reasons. This Capstone device aims to fill the gap in that market by designing an autonomous and affordable solution to disinfect surfaces.

1.3 Problem Formulation

Conventional disinfection robots are expensive and require people to evacuate before use for safety concerns. There is a need to design and implement an affordable and autonomous mechatronic system capable of successfully disinfecting surfaces in public spaces to reduce disease transmission.

1.4 Constraints and Criteria

1.4.1 Constraints

I The robot must be able to disinfect at least one table of $1.2\,[\mathrm{m}] \times 0.7\,[\mathrm{m}]$ in one charge.
II The robot must be able to operate safely on the table top without knocking off objects.
III The robot must be able to stay on the tabletop without exceeding its boundary.
IV The robot must be able to support a minimum load of $1\,[\mathrm{kg}]$ including the robot weight.
V The cost of production must be cheap enough to be considered as a viable product for restaurants, libraries and other public spaces $<$ \$300CAD.
VI The robot must pose minimum risk to nearby people (such as UV exposure).
VII The robot must be able to sanitize $> 99\%$ of common bacteria and viruses including SARS-CoV-2.

1.4.2 Criteria

I The robot must be small enough such that it does not occupy the tabletop space. (Ideally a palm-size within $1000\,[\mathrm{cm}^3]$).
II The robot must be able to cover a wide surface area of the table.
III The robot must be adaptable for it to be used in various environments.
IV The production cost of the robot must be minimized.
V The robot must clean a standard table ($1.2\,[\mathrm{m}] \times 0.7\,[\mathrm{m}]$) within $30\,[\mathrm{min}]$.

2 Final Design

2.1 Mechanical

Since the final proposal in MTE481, there have been significant changes in terms of mechanical design. In this section, major design details and modifications would be discussed here.

2.1.1 Design Details

The final prototype design is shown as presented in Figure 2-1 below.

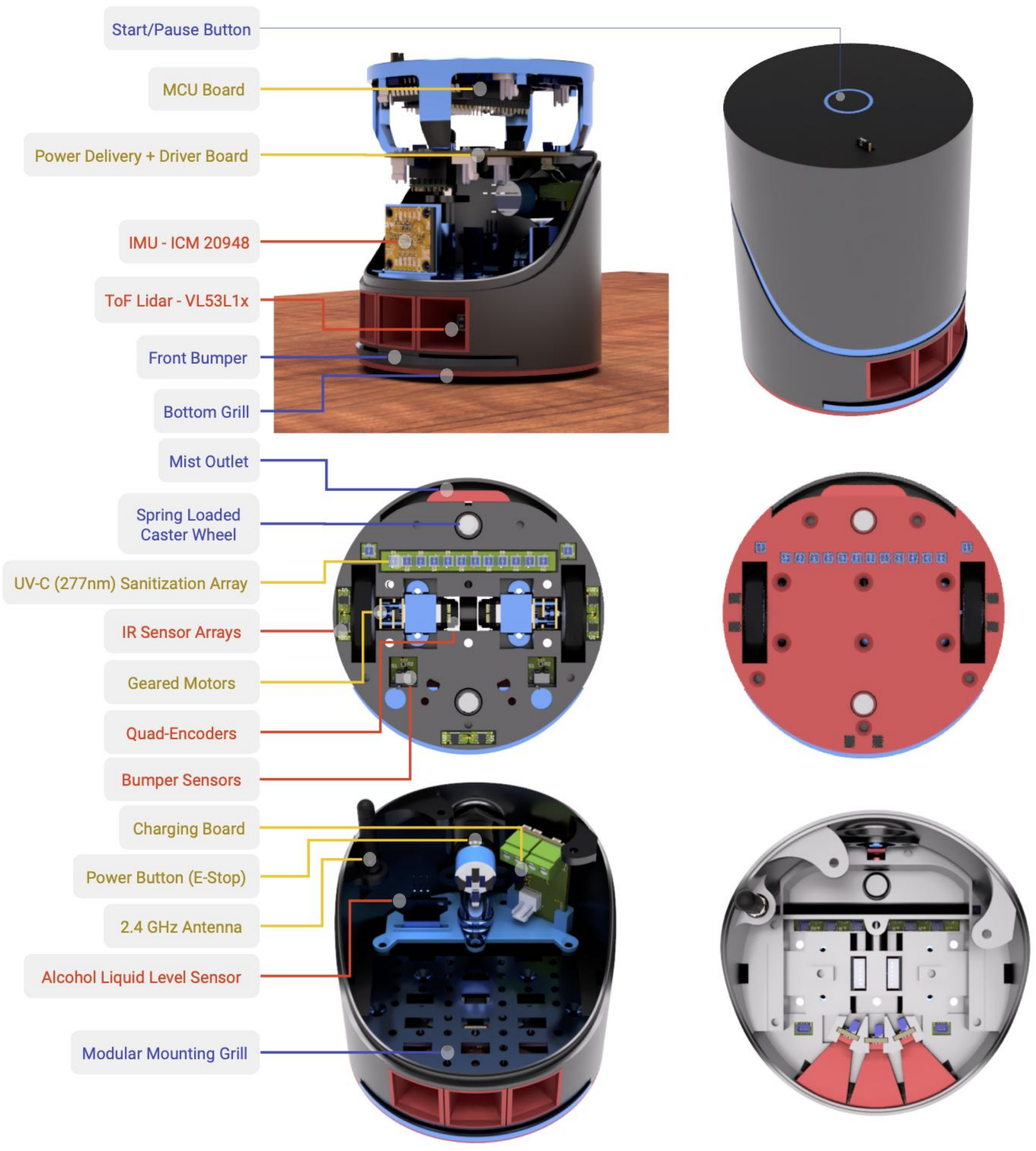


Figure 2-1. Detailed renders with labels

The entire compact chassis can be brake down into two major sets: base chassis and top lid.

2.1.1.1 Base Chassis

From the bottom-up, the base chassis is composed of a base shield, base sensors and actuator, base fixture, liquid sanitization system, ToF Lidar system, modular mounting grill with battery and IMU mounted, charging board, power delivery and driver board, and MCU board mounted on the support bracket.

The base shield is designed to hold the base sensors in place with the base fixtures (in a sandwich fashion with the help of 11 screws). There are significant design changes and detailed design attention to the placement of sensors as discussed in Paragraph 2.1.2.1.

The liquid sanitization system is designed in such a way to minimize the hazard from liquid damage from the leakage or reformation of water droplets to the hardware (as discussed in Paragraph 2.1.2.2 later). The system was tested to ensure the droplets would not form towards the base sensors, and water sealing glue was used to prevent leakage.

The ToF Lidar system is composed of three angled ToF sensors centred about the center of the robot to ensure minimal ($\leq 15\,[\text{mm}]$) dead-zone outside of the robot. A set of three channelling shields was designed to protecting sensors from the debris and cross-channelling effects between the sensors. The sensor shields and ToF sensors were designed with tight-fitting tolerance, so the modules can be removed and attached easily.

The modular mounting grill was designed to allow adaptive placement of batteries and provide a place to hold the IMU in an upright position. Since it is hard to figure the tolerance of fitting with the battery cable and battery altogether inside the fixture, a modular grill with multiple mounting points was designed to fix the robot with ease. The IMU was mounted onto an inverted T bracket, so it can stand upright for the robot pose detection.

Lastly, the MCU board is stacking on top of the power delivery (and driver) board, which sits on top of the base fixture. In addition, the two boards are oriented in a way that the power ports are aligned closely to minimize the length of thick power cables.

The curly cut-out made it super accessible for all critical electronic components, including MCU, AVR secondary drivers, IMU and all sensor data cables.

2.1.1.2 Top Lid

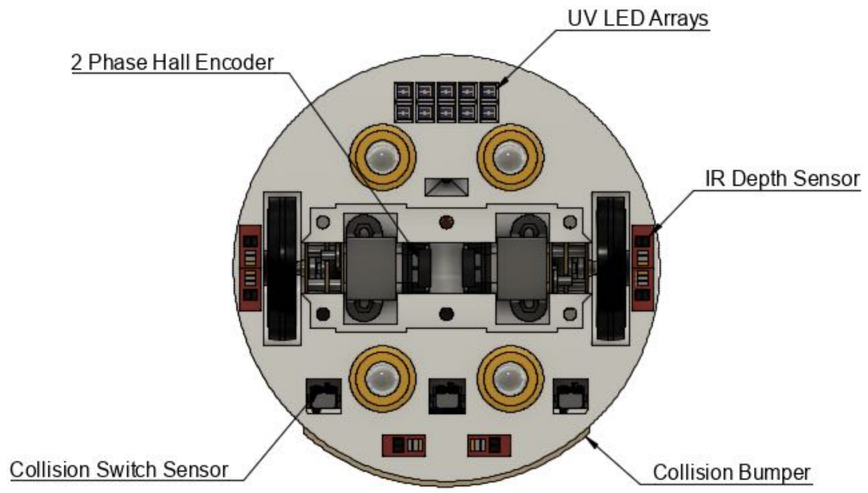
The removable top lid is composed of a power button and a simple elegantly shaped housing lid that would tight-fitted onto the base chassis and fixed with a magnetic ping.

2.1.2 Modifications from Original Design

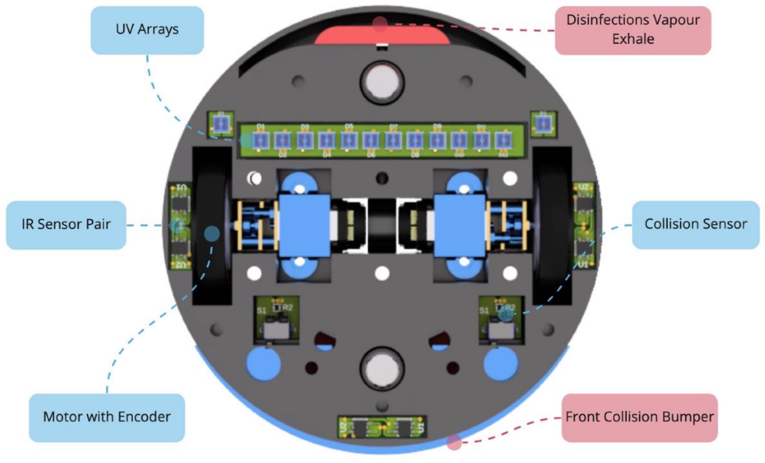
There are significant modifications from the original design. The size of the robot ($\Phi : 100\,[\text{mm}] \times H : 127\,[\text{mm}]$) is doubled from the original robot ($\Phi : 100\,[\text{mm}] \times H : 60\,[\text{mm}]$), but still within the volume criterion (Criterion I). In addition, the new design is more sophisticated and taking into account all aspects of the components and tolerances of the manufacturing processes. The detailed difference can be compared from two viewing perspectives: the base view and the cross-sectional view.

2.1.2.1 Base View Comparison

As shown in Figure 2-2 below, there are major changes on base sensor placements from the base view. Six IR edge detection sensors are aligned in a concentric circle. The UV lights are aligned in a single linear array to ensure the greatest coverage based on the effectiveness calculation. Two collision sensors are placed at the left and right end of the collision bumper to approximate the impact position from the timing difference upon triggering. Moreover, four ball casters are reduced to two sets of ball casters to reduce the contact points to the surface and increase the adaptability to different surface conditions such as uneven surface. Also, one major change visible from the base arrangement is that the vapour outlet is widened to ensure full coverage of wet treatment for sanitization. Lastly, the new design uses the sand-which method (base fixture + sensors + base plate) to keep sensors levelled and fixed at the desired height and position easily without glue.



(a) Original Design

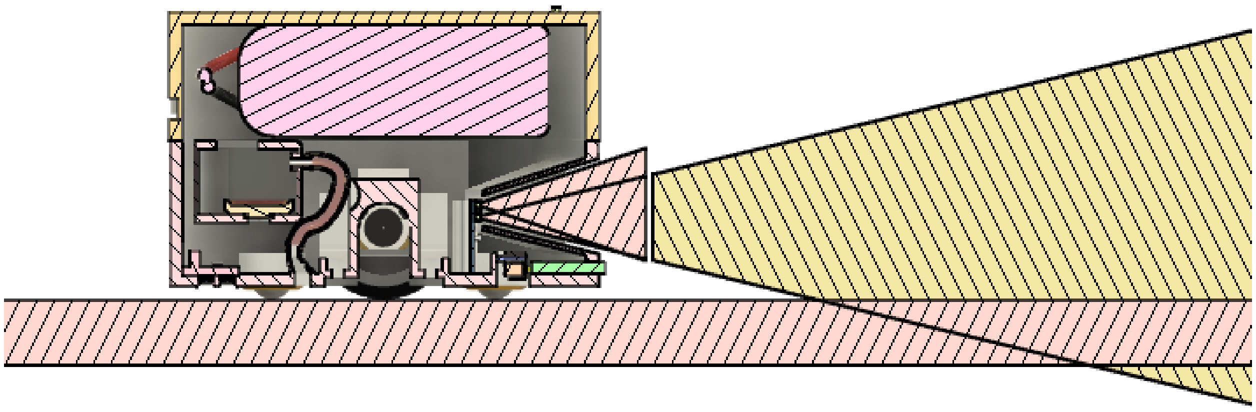


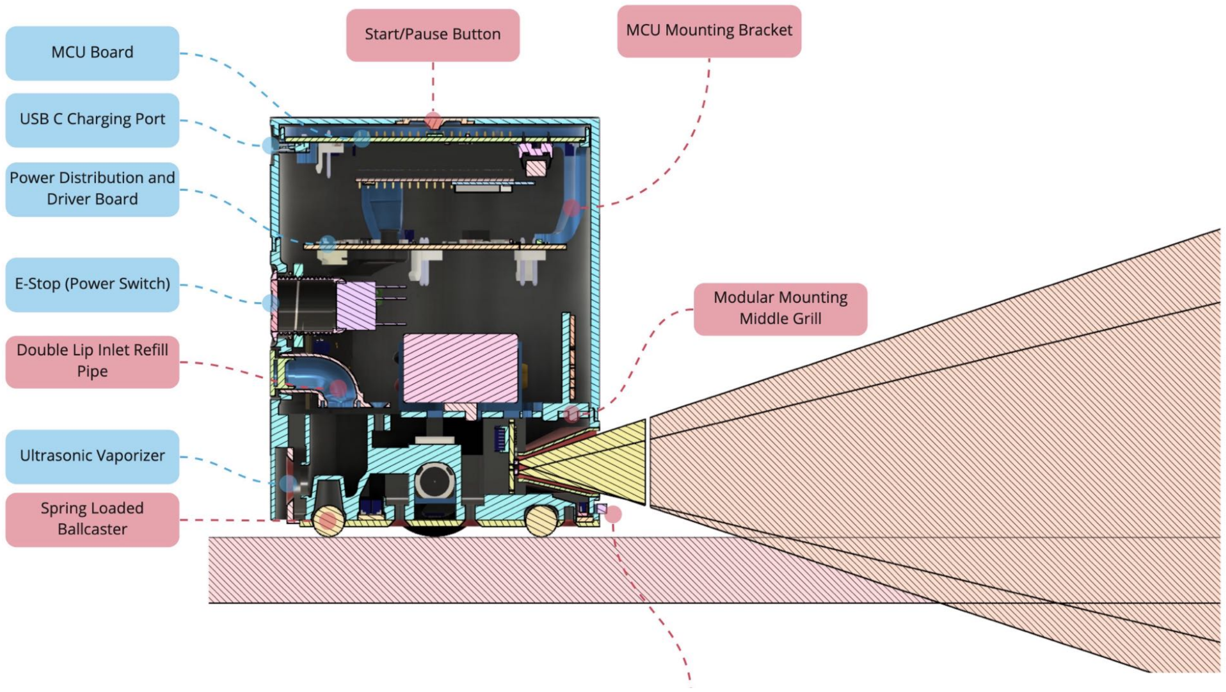
(b) New Design

Figure 2-2. Base Comparison

2.1.2.2 Cross-sectional View Comparison

A side-by-side comparison can be in Figure 2-3 below. One major change is on the vaporizer, and the original design was tested and deemed to be an unfeasible design since the water would drip easily and there are too many leakages. In the new design, the vaporizer system is in a uni-body design and no internal leakage. In addition, the exhale is widen and further away from the sensor and electronics. The refill pipe is designed to be leakage-proof with a tight-fit and double-lip design at the joint connection. Another major change is that the robot is designed much taller with more spaces and tolerances for all hardware and wiring. All hardware would be sitting on the base frame so that it would be much easier to access all the hardware during the software development phase (as demonstrated in Figure 2-6). Also, there are one minor changes on the rear ball-caster wheel, which is loaded with a spring to ensure the robot would tolerate uneven surfaces (with a max deviation of $\sigma = 3\,[\text{mm}]$). The center of gravity is designed so that it is perfectly balanced in all directions, resulting from minimal drifts or wobbles during acceleration. Lastly, a modular mounting grill was designed so that it is possible to adjust the placement of batteries and IMU sensors and fixed with tight-fit tolerance.


(a) Original Design



(b) New Design

Figure 2-3. Cross Section Views Comparison

2.1.3 Manufacturing Process

The main manufacturing process is based on FDM (Fused Deposition Modeling) 3D printer (as shown in Figure 2-4). The main material used is PLA plastic since it is a non-toxic and less shrinking ratio, comparing to other plastic. In addition, it makes the prototyping cost significantly cheaper (about 2 5 [CAD] per robot). It also allows simple manufacturing for uni-body designed chassis so that it requires minimal assembly steps.

Moreover, to make the assembly process simple, easily accessible, and highly modular, the robot was designed purposely with a single type of screws (3 [mm] hex head screws). In addition, most removable components such as sensors and some of the printed components are designed to be tight-fit. As a result, the fitting tolerance and accuracy matter a lot; hence, multiple iterations between manufacturing (> 15 times) and design tweaks (> 100 changes) were needed to perfect the right amount of fitness between all components.

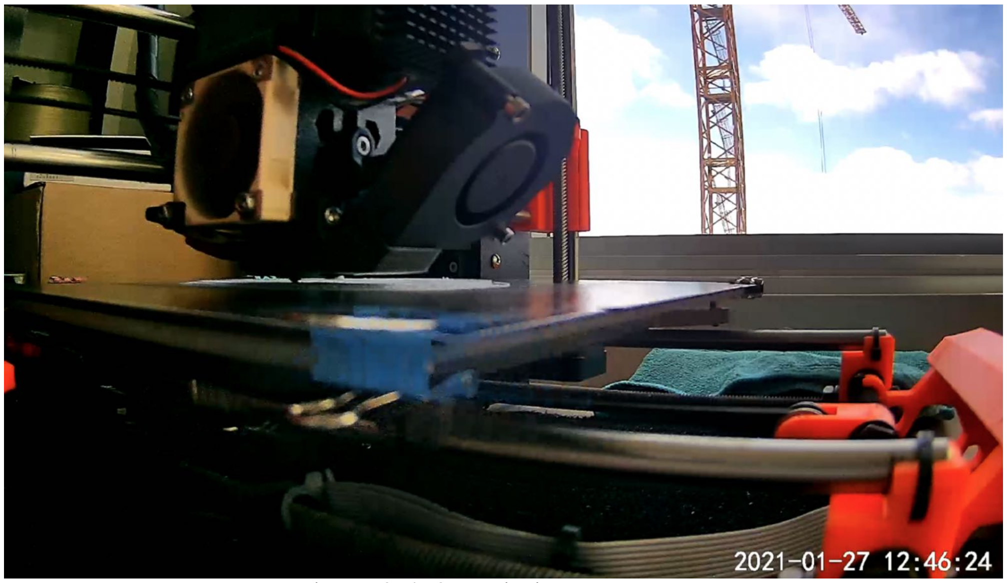


Figure 2-4. 3D Printing In Progress

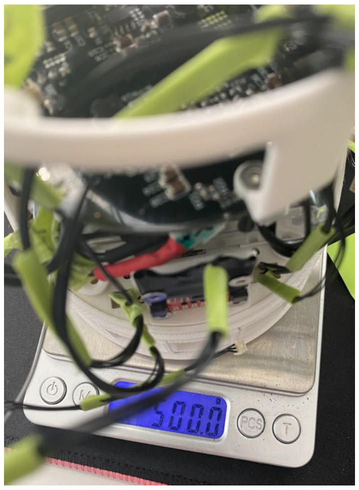
In general, the printing process takes about 48 hours to print all components of the robot in multiple sessions. After printing, the support materials have to be removed manually. The assembly process is quite straightforward from the bottom up:

1. Firstly, assemble the liquid sanitization system (Figure Appendix A-1):
   (a) Insert the ultrasonic vaporizer module, and press-fit with ultrasonic exhale shield
   (b) Mount the liquid level sensor to the reservoir lid and seal with glue
   (c) Tight-fit the reservoir lid to the tank
   (d) Make sure the reservoir is leak-free
2. Attach all base sensors and motors (Figure Appendix A-2):
   (a) Attach two sets of motors with encoders and fix them with screws
   (b) Attach two frontal collision sensors
   (c) Slide in the frontal collision bumper and fix it with screws
   (d) Insert all three IR sensors with cables pre-connected
   (e) Insert all UV-C LED
3. Ensure all sensor cables are properly connected and extended out through the base frame (Figure Appendix A-3)
4. Place down the ball casters and mount the base shield (Figure Appendix A-4)
5. Mount Power Modules (Figure Appendix A-5):
   (a) Mount the modular mounting grill with screws
   (b) Tight-fit, the charger board to the fixture on the reservoir lid
   (c) Mount the power switch
   (d) Fix the battery onto the mounting grill
6. Mount Driver Board (Figure Appendix A-6):
   (a) Mount IMU module to the mounting grill

   (b) Connect power delivery cables to the Driver Board
   (c) Mount Driver Board with three sets of crews and nuts to fix the board in a specific orientation, where removable modules are accessible without disassembling the power board.
   (d) Mount the MCU mounting bracket on top of the Driver Board
7. Flip the robot, and make sure no debris is left inside the robot (Figure Appendix A-7)
8. Connect all data and power cables to MCU board and assemble the MCU board, and do a quick power-on test (Figure Appendix A-8)
9. Close the robot with top cover.

2.1.4 Testing and Performance

As a result, the weight of robot is about 500 [g] (less than the estimated weight of 700 [g] prior the prototype) as shown in Figure 2-5a, and the final build is capable to handle a load of 2 [kg] including the robot weight at full speed (Figure 2-5b), which is more than twice the standard in Constraint IV.


(a) Robot Weight

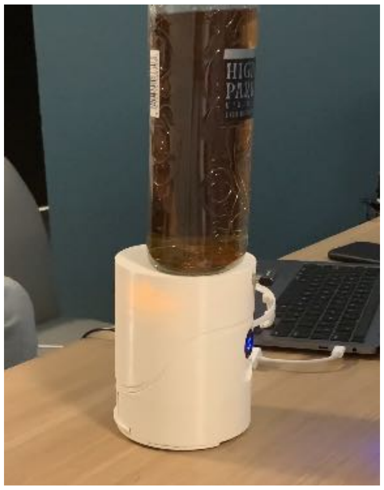
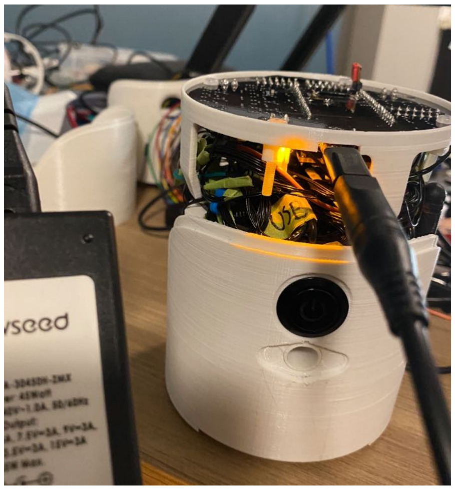

(b) Load Test

Figure 2-5. Robot Load and Weight Test

In addition, the robot as designed is proved to be quite accessible for software developments, all modules are easily accessible without disassembling the entire robot as shown in Figure 2-6.


(a) Easy Access for debugging and charging port

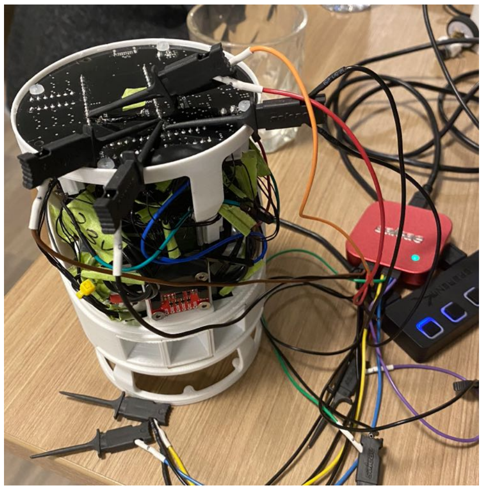

(b) Easy Access for MCU GPIOs to probe for debugging

Figure 2-6. Accessibility Features

2.2 Electrical

2.2.1 Design Details

To satisfy the strict space constraints of the robot, the electrical team designed the majority of the system from scratch. In total, eight custom PCBs were designed, which are the MCU board, Power and Driver board, charger board, ToF sensor board, IR sensor board, collision sensor board, UV LED array, and UV LED single board. To reduce the cost, the PCBs were designed with two layers, and to reduce the manufacturing complexity, the majority of the components were kept on a single side of the board. The final PCB design render is shown in Appendix B. To simplify the harnessing, an off-the-shelf cable is used for low current signals connected with JST connectors, and for high current path such as battery input and LED supply, 16 AWG cable with XH connector is used. The electrical system architecture was modified from the one in 481 reports during the circuit design to simplify harnessing and optimizing board space. The full final electrical system architecture is shown below in Figure 2-7.

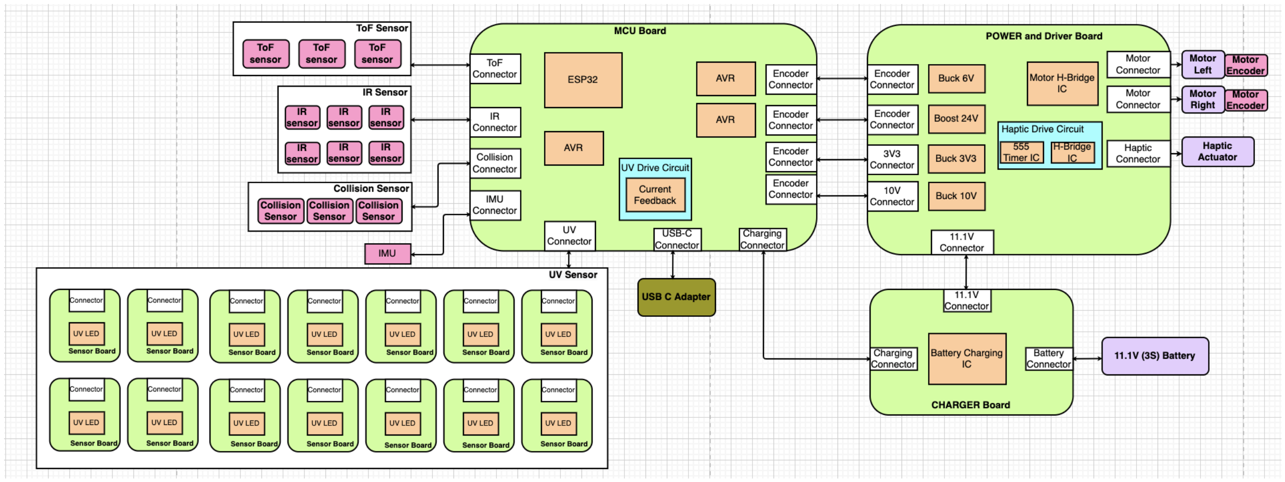


Figure 2-7. Electrical Architecture Diagram

2.2.2 Manufacturing Process

After designs were completed and checked, Gerber files were generated and sent to the manufacture. All PCBs were ordered through PCBWay alongside their respective stencils. Stencils are sheets of metal that are cut out to easily smear solder paste over the proper areas of the PCB in manufacturing. The electrical components to be soldered on the PCBs were ordered separately from Digikey. Three or more batches of each component were ordered just in case there were any mistakes in manufacturing or testing of the PCBs. Manufacturing of the PCBs started once the PCBs, stencils, and components arrived. To efficiently solder the components onto the PCBs, a process called reflow soldering was used. Since there were no available reflow ovens nearby, a hotplate was used instead. The group borrowed a Volterra V-one from one of the member's former companies to use as a hotplate. The machine is able to control the temperature and define a solder reflow profile which was perfect for improving the manufacturing process. The general procedure for soldering each PCB starts off with using the stencil and aligning it with the features of the PCB. Using a squeegee, solder paste was smeared on the stencil on top of the PCB. After removing the stencil, components from the Digikey bag were removed and placed onto the designated areas of the PCB. Lastly, the PCBs with the components and wet solder paste was placed onto the Voltera hotplate on which a pre-define heating profile is performed. The solder paste then hardens, completing the soldering of the components onto the PCB. Most of the PCBs had a few components that needed to be soldered on the backside of the PCB, and this was done by manually soldering with a soldering iron.

2.2.3 Testing and Modifications from Original Design

After all, PCBs are manufactured, a bench test is conducted using the power supply and oscilloscope. The goal of the bench test is to confirm that hardware operates as intended and there are no design errors or flaws. Below summaries of the major issues found and design modifications made to solve them.

2.2.3.1 10V lead compensator issue

The first bench test conducted is to make sure the power regulations work with no load. Since the robot runs on a Li-Po battery, all circuit needs to function properly before being connected to the battery. In the power and driver board, there are five power rails, the main 12V battery, bucked 10V for the UV LED, bucked 6V for motor, bucked 3V3 for MCU and sensors, and boosted 24V for the haptic drive. For the test, 12V is supplied with the power supply, and each rail is probed with an oscilloscope to confirm the output voltage signal. With this test, an issue was found on the 10V rail as the output voltage was only 9.4V with peculiar 34khz oscillation riding on it. Additionally, the current drawn from the power supply was significantly higher when the 10V rail is turned on and the inductor heated up as well. After debugging, the capacitor of the lead-compensator added to this switching regulator was short-circuiting and causing malfunction to the circuit. Thus the design was modified to remove this lead-compensating capacitor.

2.2.3.2 Haptic time constant

The haptic drive circuit on the power and driver board and consists of a 555 timer IC that generates the 113kHz resonant frequency for the haptic actuator and H-bridge IC that drives this accordingly. During design and simulation, this resonant frequency was set using the calculation formula provided on the internet and confirmed with simulation. However, during the test, the output frequency was found to be 80kHz and significantly smaller than the desired frequency. By measuring the capacitor charge/ discharge with an oscilloscope, it turned out that the provided formula was wrong. After hand-calculating the capacitor charge and discharge time with the resistor and capacitor value used on the circuit, the output frequency was calculated as the observed output value. Thus now back-solving with resistor and capacitor value with the desired frequency, the required resistor value was computed. Since the required value of resistor was not available, two resistors were used in parallel to create the equivalent desired resistance, and this was done so on the hardware by stacking one 0603 SMT resistor on top of another. The final output of the haptic drive signal is shown below in Figure 2-9, and the frequency is indeed 113kHz.

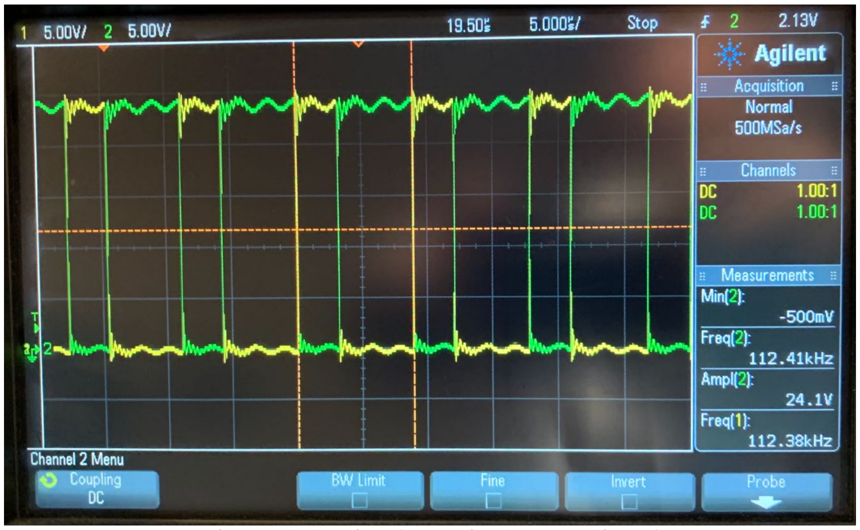


Figure 2-8. Fixed Haptic Output Signal

2.2.3.3 ToF XSHUT

ToF sensor board designed which uses the Tof sensor, VL53L1X from ST Microelectronics, consists of 6 signals which are power, ground, SDA (I2C), SCL (I2C), interrupt, and shutdown signal. In the design, there are three ToF sensor boards, and to decrease the number of control signals from the MCU board, the shutdown signal for each sensor was tied together. However, during firmware testing, it was discovered that three sensors require individual shutdown signals. This allows the I2C address for each sensor to be modified so that each sensor can be run in parallel. This issue was figured out once the PCBs were designed, and the main MCU, ESP32, had all of the GPIOs used up. Thus, the three unused pins from the secondary 8-bit AVR MCU were used for this shutdown signal by extending fly wires from the unused pin to the sensor cable, as shown in the image below. This is not ideal as the control signals are split between the main MCU and secondary MCU, but as this shutdown signal is only required once when booting up the sensors, it did not cause a problem during implementation.

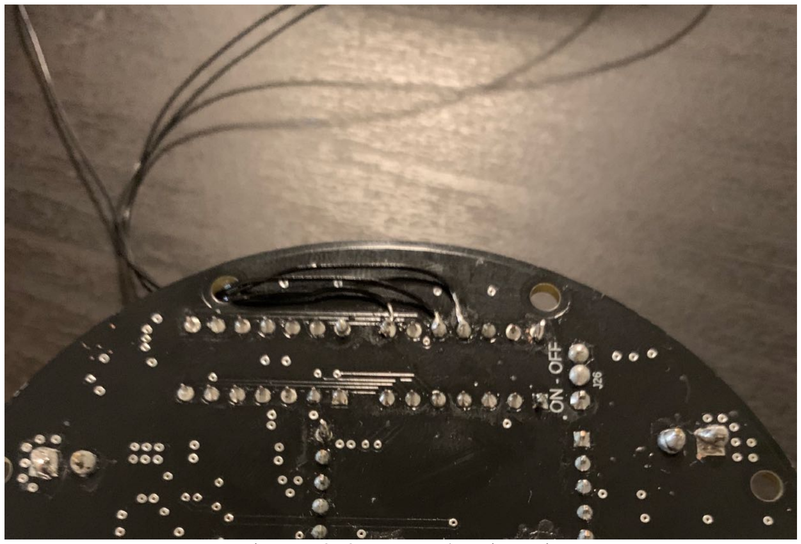


Figure 2-9. ToF Flywire Fix

2.2.3.4 Charger IC Testing

Testing the charger board required extra care as improperly charging a LiPo battery can be very dangerous. Therefore, portions of the circuit were tested and verified in order of least dangerous to most dangerous actions. The first step was to power the charger IC with the 13.5V input using a benchtop power supply. When there is no battery connected, the charger IC is designed to pass the input to the output of the board. This was verified with a multimeter to see if the input matched the output. After that, known voltage levels on various points of the circuit were probed to verify that the circuit was in a non-charging state. Next were tests involving the insertion of the battery into the charger board. First, tests were run with the battery initially at full charge to determine if the charger IC reacted accordingly. Then, after depleting the battery, the charging portion of the circuit was thoroughly tested. Through the power supply readings, one could confirm that the charging current was regulated as intended and slowly depleted as the battery reached its completed charge state. Overvoltage and under-voltage input states were also tested by varying the power supply, which had succeeded.

2.2.3.5 Charger board thermistor

During the design process, the charger board was designed to connect to a thermistor that was to be placed near the battery. The battery charger IC that was chosen has a feature to take this thermistor reading and monitor the battery's temperature accordingly. However, in evaluating how the thermistor would be placed in the mechanical apparatus, it was decided that it would be too much of a hassle at this moment to account for the thermistor wire assembly. However, since the circuit was designed to have a thermistor connected, the logic implemented in the circuit would not operate if the thermistor was missing. The voltage divider in place would make it seem like the temperature is way out of the bounds of the intended operating temperature. This was an issue because this impeded the charging of the battery. Upon replacing the thermistor placeholder with a 25kohm resistor to imitate constant ambient temperature, the charger IC was able to charge as intended.

2.2.3.6 Battery Inline PTC Fuse

After all, PCBs were tested and confirmed that they operate without any flaws, the battery is used to supply the loads. However, at this point, the team realized that inline fuse was not added to the design and thus, if load shorts large current can be drawn from the battery. Therefore, a PTC fuse was purchased to protect against fast blow and slow blow fuse. PTC was selected instead of mechanical fuse as physical replacements are not required, and as it can be integrated into the existing system with ease by soldering it with series to the battery path.

2.2.4 Performance

After hardware testing is completed and all malfunctions are removed, the performance of the electrical system is measured. This is done to verify the designed electrical system works on the targeted environment and setup.

2.2.4.1 UV LED Effectiveness

The performance of the UV LED sensor against viruses and bacteria was initially planned to be measured with help from professor Aucoin in his lab at the University of Waterloo. However, due to the pandemic and lockdown, the lab experiment was not conducted, and thus measurements were not taken. From LED's power dissipation, theoretical effectiveness was computed, but this couldn't be confirmed.

2.2.4.2 Motor and Encoder

The performance of the motor is measured by setting the same PWM frequency and measuring the motor speed for both motors. The initial experiment conducted by setting both motors at the same PWM frequency resulted in drifting, and thus motors had to be tuned for open-loop control. The speed of the motor is obtained by using the encoder on the motor, and the motor is provided with a PWM value that sweeps from 0 to 100% duty. This PWM duty to encoder value is compared for both motors to tune the motor output. Additionally, the encoder sensor tick to distance conversion is obtained by measuring the encoder ticks against a specific distance. This can then be used to measure the distance moved by the robot using the encoder data.

2.2.4.3 Battery Charging Current

The main performance criteria to be measured with the charger board was its ability to charge the battery. During testing of the charge current for the charger board, it seems that the recorded current being drawn was around 1.027 A. The intended charging current was supposed to be 1.3A, meaning it would take about 95 minutes to fully charge the 1300mAh battery to completion. The intended charge time was 1 hour, so this is definitely a performance drop from the original design.

2.3 Software

2.3.1 Design Details

2.3.1.1 Overall Architecture

In this section, we describe the software architecture of TableUV. In essence, it is divided into two branches, namely the App Level and the Dev Level, which takes advantage of the dual-core hardware capabilities of the ESP32. All of the sensor updates, including those for the AVR and the supervision over the current state of the robot, are performed in Dev Level on Core 0, whereas the App Level takes those sensor data to perform more hardware-intensive tasks such as SLAM.

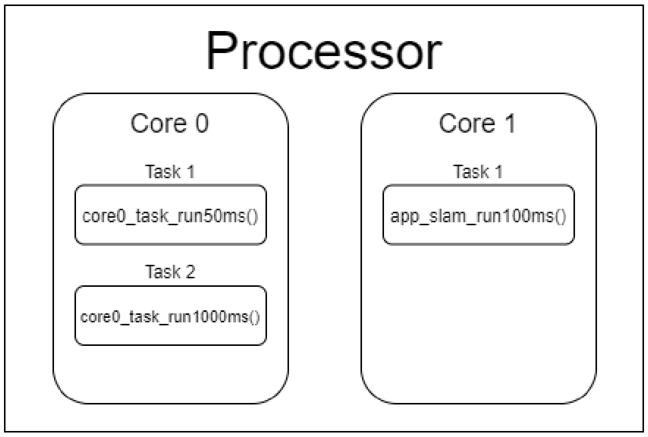


Figure 2-10. Diagram of the different tasks that are assigned to each of the cores in ESP 32.

2.3.1.2 AVR: Secondary (Slave) Level Software

In the electrical design, three 8-bit AVR MCUs were used to offload some processing power from the main ESP32 MCU. Two of the AVRs are designed to interface the two motors and their encoders [driver AVR], while the last AVR was meant to interface the six IR sensors and two collision sensors [sensor AVR]. Using secondary MCU means that a communication protocol needs to be defined between the AVRs and the ESP32. Therefore, the two motor AVRs were designed to communicate using I2C, while the sensor AVR would use UART. The low-level AVR firmware architecture is shown below.

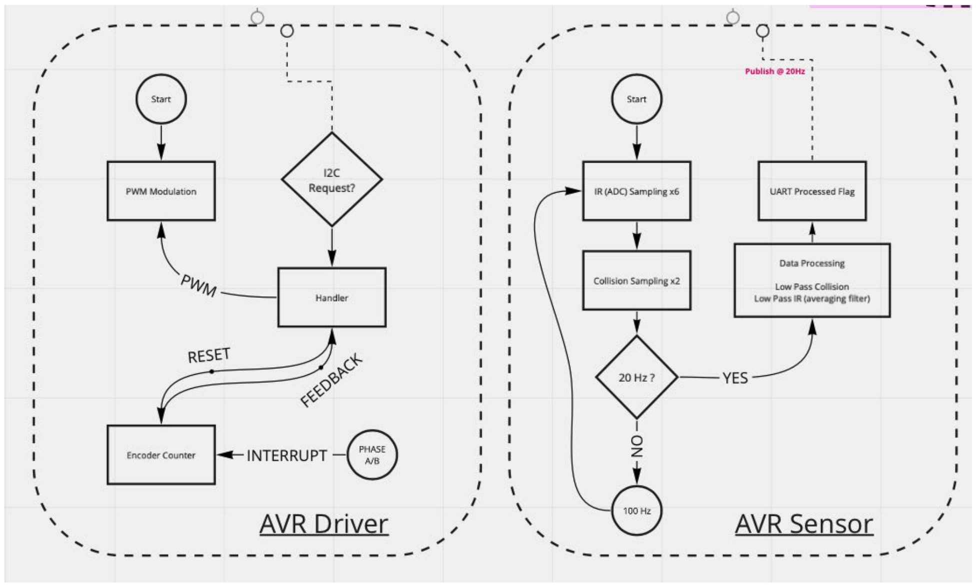


Figure 2-11. Diagram of the AVR Firmware Structure.

To write firmware and flash code onto these AVRs, an environment was set up using VSCode and platform to allow seamless access to the AVRs. To interface VSCode and the AVRs, an Arduino UNO was wired to a breadboard in a certain format so that an AVR can be placed and flashed accordingly.

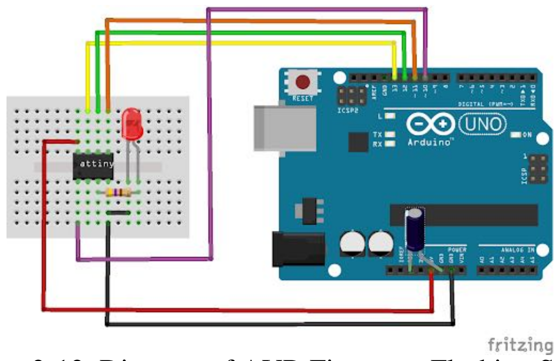


Figure 2-12. Diagram of AVR Firmware Flashing Setup.

The sensor AVR used 8 pins for the 8 different sensors, and 2 pins were reserved for UART communication. For the IR sensors, a simple ADC read is performed continuously at 200Hz and checks if the value is above a certain threshold. This threshold was determined by testing the IR sensors when faced in front of a table versus off the edge. The collision sensor is continuously read in a similar manner but instead of using a predefined threshold, the firmware relies on the AVR reading a digital HIGH versus LOW at the pins. UART is often used with two pins, one for transmitting and one for receiving. Since there was no need to transmit data from the ESP32 to the sensor AVR, only UART transmission from the AVR to the ESP32 was established. It was decided that a baud rate of 115200 would be chosen for the UART communication. The structure of the message at each timestep would be a single byte, each bit indicating whether a sensor has been triggered, with 0 being the normal state. However, since transmitting bytes at 200Hz intervals may overload the ESP32 which has to handle the receiving of said message, a low pass filter was implemented to reduce the effective frequency. For each sensor, every 20Hz, the firmware would check if 8 of the last 10 sensor readings were a trigger reading, and update the UART byte accordingly. Therefore, the sensor AVR is now successfully sending data at 20Hz rather than 200Hz.

Each driver AVR has common components which are the motor, encoder, and I2C interface. There is a GPIO pin reserved for mode selection on hardware which indicates the side of the motor the AVR is controlling. This allows for a common code to be used for both motors, and also allows different I2C address to be set. In addition to the common components, each driver AVR controls unique peripheral devices. Left driver AVR controls the ToF initialization signals that were introduced from the hardware testing and modification, and the right AVR controls haptic driver enable signal and the water level sensor input. The driver AVRs are communicating with the ESP32 continuously at 20Hz where the ESP32 (master) sends motor drive and peripheral commands to the driver AVRs, and the driver AVRs responding back continuously to master with the encoder sensor values. The encoder signals are connected to the interrupt pins on the AVR, and decodes the two phases into ticks using quad-encoder scheme. Motor is controlled with PWM frequency running at 2kHz, and I2C is running at 100kHz.

2.3.1.3 ESP: Device Level Software

2.3.1.3.1 Device Configuration

The ESP32 microcontroller has an RTOS implemented called FreeRTOS. This meant that the ESP32's code execution is now divided into tasks that repeat at a predetermined frequency. In this section, the device level task structure will be explained. On the device level there are three tasks, one running at 1Hz, 20Hz, and then 50Hz. There are also some device level libraries that were created but only called directly in the application level, mainly by the supervisor task running at 20Hz.

2.3.1.3.2 Device AVR Driver

"`dev_avr_driver.h`" is a library designed to handle the I2C message protocol and commands to interface between ESP32 and driver AVRs. This library utilizes the ”Wire.h” library available on ESP32 to handle the I2C communication. Command messages are stored in the data struct created inside the library, and the public functions created can access and modify the struct data. The message update, transmit, and receive is being ran in the 50kHz task. Additionally, the encoder data received is converted into distance and angle which is used by the global map.

2.3.1.3.3 Device AVR Sensor

"`dev_avr_sensor.h`" is a library designed to handle the UART messages coming from the sensor AVR and structure the incoming data accordingly. The library uses ESP32's built-in hardware Serial library to establish the 115200 baud rate connection. A function was created to check the serial stream for incoming data from the sensor AVR and that function is being ran in the 50Hz task. If there is new data on the stream, it is stored locally. The app level can then retrieve that local data using a get function.

2.3.1.3.4 Device Battery

"`dev_battery.h`" is a library designed to handle the charger board's IC status messages. It is also where the battery voltage monitoring functions are designed. The LiPo charger IC on the charger board has three states: charging complete/idle, charging, and fault. This is shown through LOW, HIGH, or a toggling state on the status pin respectively. Handling the first two states is very simple; the ESP32 simply must read the status pin and check for its digital state. To handle the fault state, there is no built in function to check whether the pin is toggling at certain frequency. There is a way to setup an interrupt when a pin's digital reading changes. The interrupt service routine can simply increment a variable and now there just needs to be a function that checks that variable to see if it increments at a certain rate.

2.3.1.3.5 Device IMU

"`dev_imu.h`" is a library used to interface with the ICM-20948 IMU sensor. It initializes the configuration of the IMU to its default parameters for the sampling modes, full scale and also enabling the use of the magnetometer. The `dev_imu_get_values()` is called at a certain frequency during which the sensor is checked for new readings. If there are new values, the mutex guarding the shared data is locked and the values are updated; after which the mutex is locked again. Some initial pre-processing of the sensor reading is also handled within this library such as converting accelerometer values from g to m/s^2.

2.3.1.3.6 Device Status LEDs

"`dev_led.h`" is a simple library that handles the LEDs and button of the robot. There are several functions designed to change the output of the green and red LED at the top of the robot. There is also a function to check if there was a button press. Button bouncing is a phenomenon where the mechanical position of the button can fluctuate from on and off when pushed before settling to a certain state. Since no hardware debouncing was designed for the button, this needed to be handled in firmware. This was done by apply a simple filter in code to reassure the state of the button.

2.3.1.3.7 Device ToF Lidar

"`dev_tof.h`" is a library that handles asynchronous firing sequence and data acquisitions of three sets of ToF Lidar sensors. The library uses I2C to communicate with the ToF modules. Initially, the library would power on each ToF sensors in sequence through "`dev_avr_driver.h`" (a work around since the ESP has ran out of GPIOs), and reconfigure each of them with different I2C address, so that all three modules are capable to fire and capture data in parallel. The library would acquire data from each sesnors and configure the next firing window and settings for each sensor to allow continuous and high-speed capturing of 15 data points per 100 [ms]. This allow a sensor resolution of 4° per obstacles, which is equivalent to a resolution of 6 [mm] at the depth of 100 [mm]. To note, we turn off all low angled sensing regions to avoid inaccurate measurements from the table surfaces. The library also does the signal validation from sensor conditioning registers, so it is capable to know the conditions of the signal, and only keep those that are good conditioning signals and assign low probabilities to warning signals. In the end, these data would be memory copied by the `"app_slam.h"`, and processed in parallel with Gaussian mixture model.

2.3.1.3.8 Device UV

`"dev_uv.h"` is a library that handles the UV LED drive circuit. The UV LED drive consists of reference signal generation via DAC output, and duty generation from pwm output. The function takes in the DAC and duty value as an input, and outputs the corresponding drive signals. Additionally, setting the firmware shutdown signal HIGH will turn off the UV drive circuit for safety reasons.

2.3.1.4 ESP: App Level Software - SLAM

In order to meet the constraints such as being able to safely maneuver around the table without colliding with objects and to be able to detect objects, a clear understanding of the robots surrounding was necessary. To achieve such tasks the App Level Software as mentioned before is run on a separate core from the sensor updates where it primarily uses the updated sensor values to perform simultaneous localization and mapping to an extent.

One of the main objectives of the App Level was to achieve SLAM. This was one of the most important tasks to be handled as it would allow the robot to have a good belief of the surrounding map along with the locations of the various obstacles that are within the robots proximity. The overall flow diagram for SLAM is seen in the figure below.

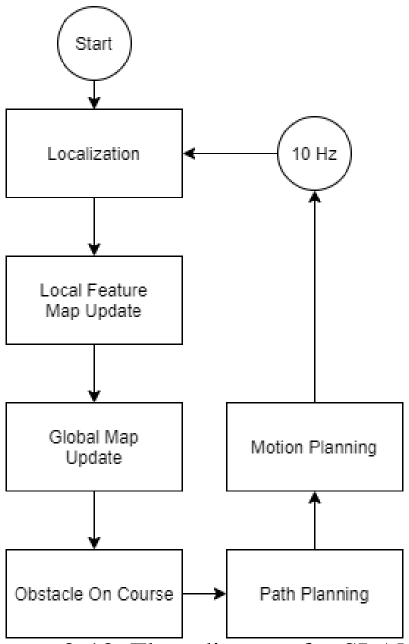


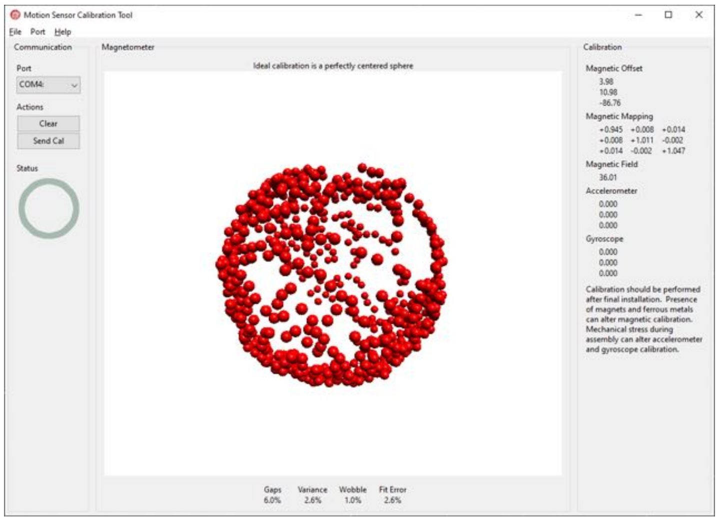
Figure 2-13. Flow diagram for SLAM.

2.3.1.4.1 Localization

Localization is mainly handled using the values from the encoders. Using the geometry of the robot wheel along with the tick counts, the displacement of the robot in terms of position and orientation from its initial pose is calculated. For obvious reasons, relying solely on the encoders for positioning is prone to drift. We try to minimize this using a 9-DOF IMU (ICM - 20948) which allows to capture 9 distinct types of motion; namely acceleration, angular velocity and magnetic orientation in the X, Y and Z axis.

In order to incorporate the IMU into the localization, several steps needed to be taken. First of all, the IMU was calibrated to ensure that proper values were being published from the sensors. Special care needed to be taken for the magnetometer as it measures small magnetic field. Therefore through calibration any hard iron magnetic offsets are determined and removed so that the magnetometer values can be used to determine the magnetic north with accuracy.

After calibrating the IMU, the values from the IMU need to be fused together in order to estimate the orientation. Initially, a Kalman Filter approach was considered which is a popular orientation filter algorithm amongst many commercial applications. This is due to their accuracy and effectiveness but comes at a price of requiring a high sampling rate of around 512 Hz up to 30 kHz. Additionally, large state matrices and their corresponding matrix calculations lead to requiring large and often inefficient complexities.



Port: COM4:

Ideal calibration is a perfectly centered sphere

Magnetic Offset
3.98
10.98
-86.76

Magnetic Mapping
+0.945 +0.008 +0.014
+0.008 +1.011 -0.002
+0.014 -0.002 +1.047

Magnetic Field
36.01

Accelerometer
0.000
0.000
0.000

Gyroscope
0.000
0.000
0.000

Calibration should be performed after final installation. Presence of magnets and ferrous metals can alter magnetic calibration. Mechanical stress during assembly can alter accelerometer and gyroscope calibration.

Gaps	Variance	Wobble	Fit Error
6.0%	2.6%	1.0%	2.6%

Figure 2-14. Flow diagram for SLAM.

Due to our limited hardware specifications, rather than using a Kalman Filter, a Madgiwck Filter is used which uses the accelerometer, gyroscope and magnetometer values to correct itself for orientation. This provided fairly steady values for orientation but was quickly realized that the use of a magnetometer in an indoor setting such as our robot would be infeasible as there would be many sources of magnetic fields that cannot be accounted for during the calibration.

2.3.1.4.2 Local Feature Map Update

The next step is to obtain the local feature map from the device level. It would memory copy the entire feature points captured from ToF Lidar since last fetch, and the current status of collision sensors and IR edge sensors.

2.3.1.4.3 Global Map Update

After the update of local feature map in terms of feature clusters, it is possible to update the global map. The global map would self update based on the pose and global coordinates of the robot from the localization. Then, it would projects all features clusters based on look up tables and converting polar coordinates to cartesian coordinates. The global map uses exponentially weighted averages to retain the history of the map while fusing the latest information.

$$v_t = \alpha v_{t-1} + (1-\alpha) g_t \tag{1}$$

where:

v_t	= current value of the grid cell to be updated
v_{t-1}	= previous value of the grid cell
g_t	= new values to be fused
α	= rate of memory loss

There are some extensive and extreme algorithmic optimization to allow the complicated SLAM algorithm running less than 20 [ms], permitting more algorithms to be implemented in the future. Specifically, the gobal map utilizes the tile updating strategy to create an infinity map in a finite static memory allocation. The edge sensors are projecting with a pre-populated configuration that does not require additional heavy floating point of trigonometry, while projecting sensors properly in different pose of the robot. In short, a lot of integer optimization and quantization were formulated to reduce the run-time computation complexity. The resultant global map is quite neat as shown in Figure 2-15.

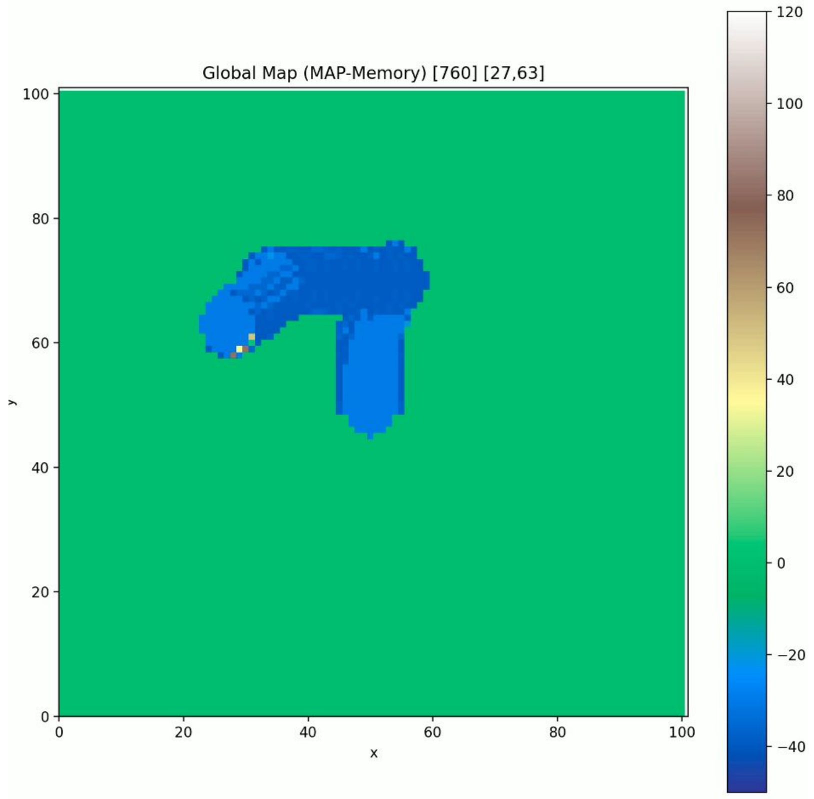


Figure 2-15. Live Global Map Serial Output from ESP32 (Live monitoring in Python)

2.3.1.4.4 Object On Course

From the global map generated cumulatively, the objects that are in the way of the robot motion can be computed. This data can be used by supervisor node to terminate the motion profiles for emergency stop.

2.3.1.4.5 Path Planning and Motion Planning

Currently, these two sections are still working in progress, and the motion profiles are handled by the supervisor node. The plan is to utilize the global map to generate an ideal path with algorithm like A* so it can ensure full coverage of the cleaning instead of random walk as right now. Then the motion planning would convert the trajectory path to a motor profiles so that the supervisor node can command the motor controller at a higher rate (20 [Hz]), via state space control strategy.

2.3.1.5 ESP: App Level Software - Supervisor Node

One key app level software is the supervisor node. Originally, it was planned to live at the same core with SLAM, but it came to a realization that the supervisor node shall be updated at the same rate possible with the fastest devices. Hence, context switching between dual CPUs can really slow down the system by semaphore blocking. Hence, the supervisor node is brought down to device level, and it does not need semaphore blocking with devices and all it needs was the critical object on course semaphore signalling from the app level SLAM. The main purpose of the supervisor node is to provide the supervision on the entire status of the robot, and it would have the right to update the command output to the driver devices. This will ensure the safety of the robot and its users. The supervisor would turn off the UV light and motor when the robot is lifted or detecting an edge of the table. Lastly, the supervisor node is currently performing pre-programmed choreography to avoid obstacles upon the detection of the edge and obstacles at distance from all sensors.

2.3.2 Modifications from Original Design

Due to the constraints in time, some of the tasks that the team initially set out to complete were not achieved and therefore were either loosened or dropped in order to advance the project. Here we outline some of these details.

2.3.2.1 Localization through sensor fusion of IMU and Encoders

Although we were successful in gathering encoder data from the encoders, the IMU proved to be very inconsistent to be used indoors. Even with accurate calibration of the magnetometer to mitigate any hard iron offsets, the initial values of the IMU were different in each iteration of testing. Therefore, the localization of the robots position and orientation was determined only through the encoder tick values which were prone to drift.

2.3.2.2 Coverage Path Planning

With localization not having been fully implemented, it was difficult to implement coverage path planning algorithms such as the spiral motion or the zig-zag motion. These motions requires a good belief of the map along with the locations of the various obstacles that are present on the table. These coverage path planning motions were preferred as they consume less time and do not revisit a portion of the table that had been previously sanitized. We decided to use a random coverage algorithm where the robot would follow a a straight line in a random direction until it either hits an obstacle or detects the edge of the table. Once such event occurs, the robot would turn right, repeatedly checking that the obstacle/edge is no longer detected; at which point the robot continues to move forward. This method was rather simple to implement as it didn't require the knowledge of a table's map nor the location of the robot but it did mean that we couldn't verify a 100% coverage of the table and also that there would be repeated sensitization over the same area.

2.3.2.3 Migrating from ESP-IDF to Arduino

During the initial development process of the development, we were relying on ESP-IDF (Espressif IoT Development Framework) as our main basis for building and flashing firmware onto the ESP32. Initially, it was thought that the ESP-IDF would be better suited for our application as it is native to the ESP32 meaning that the API would allow us more control over the ESP32 compared to Arduino's API. Although this was very much the case, we soon realized that Arduino has a wider variety of open-source libraries to use compared to ESP-IDF which was ideal due to the tight deadline that we were up against. Additionally, getting to a comfortable level in using ESP-IDF required significant amount of studying which we did not have. These were the reasons why we quickly made the switch to Arduino.

2.3.3 Development Process

Initially, we started off with a basic simulation of the robot in a tabletop environment where it was possible to test the basic ideas that we had for the project. Using Webots and Matlab, we developed a fairly accurate representation of the robot with all of its actuators and sensors with which simple localization and path planning algorithms were implemented.

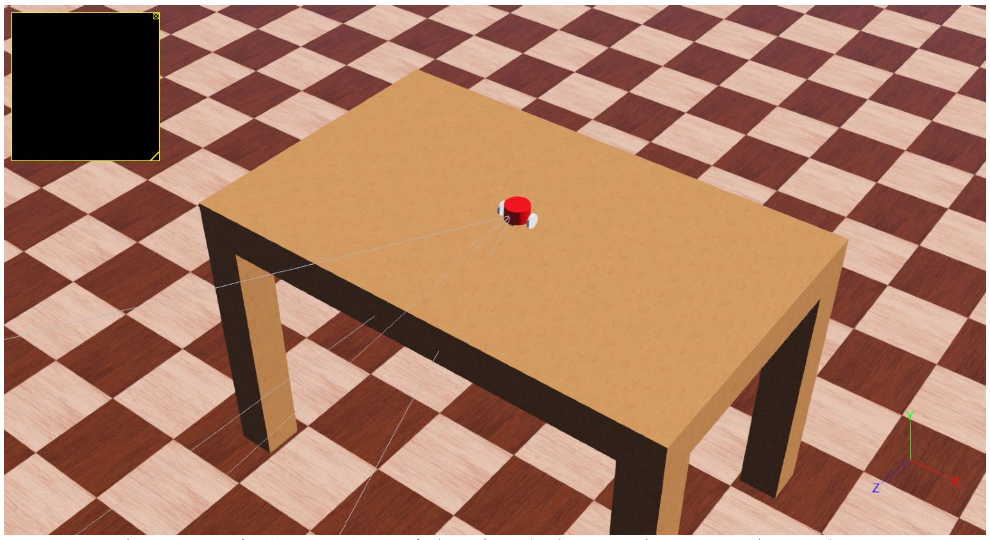


Figure 2-16. Screenshot of the simulation environment in Webots.

Afterwards, the development for the actual software to be flashed onto the robot was developed in C/C++ using Github as our main version control and source code management system. We started off with developing the firmware interface for the various sensors that are housed on the robot. Each sensor was developed separately on their respective branches and tested on the actual hardware to verify that the sensors are working correctly. After integrating the low-level code, we moved on to developing the higher level software pipeline such as developing the algorithms for SLAM, collision avoidance and a token state machine. These tasks were created using Github Issues and were assigned to one of the available members. The pull request was then reviewed by each of the members before being merged to the main branch. This process allowed different sections of the code to be worked on in parallel as well as distribute the responsibility throughout the members.

2.3.4 Testing and Performance

Different functionalities within the code were separated using macro feature flags. These feature macros defined which application level or development level software should be included in the complication therefore allowing us to test each of the features separately and also to track down any bugs in the code.

Each feature flag was tested separately by visually verifying the functionalities were being executed successfully. Additionally, the different test cases were manually simulated to verify the robots functionality in different scenarios as well. This process was especially important for the high level supervisor code that handled the different states that the robot was in. It was critical for the robot to stop the UV LEDs when the robot was either lifted from the table or e-stopped for whatever reason. All of these safety critical scenarios were heavily tested to ensure the safe use of the robot in commercial settings.

3 Schedule and Budgeting

3.1 Schedule

The table in Appendix -1 shows the different tasks that were briefly organized throughout the term to determine the ordering, priority and dependency of different tasks. We had meetings throughout the beginning of the term to discuss the plans on how to execute the project and came up with flexible milestones that we wanted to achieve as seen in the table.

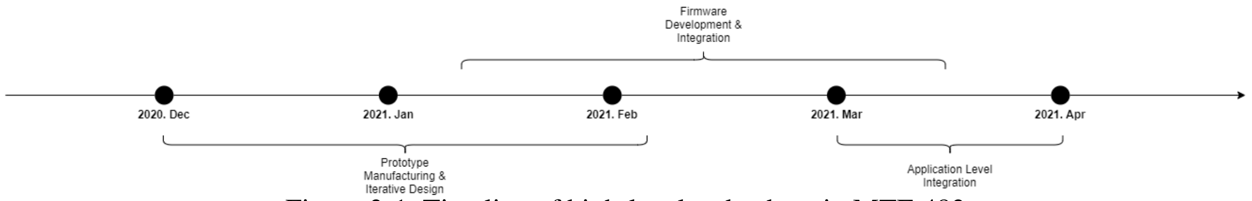
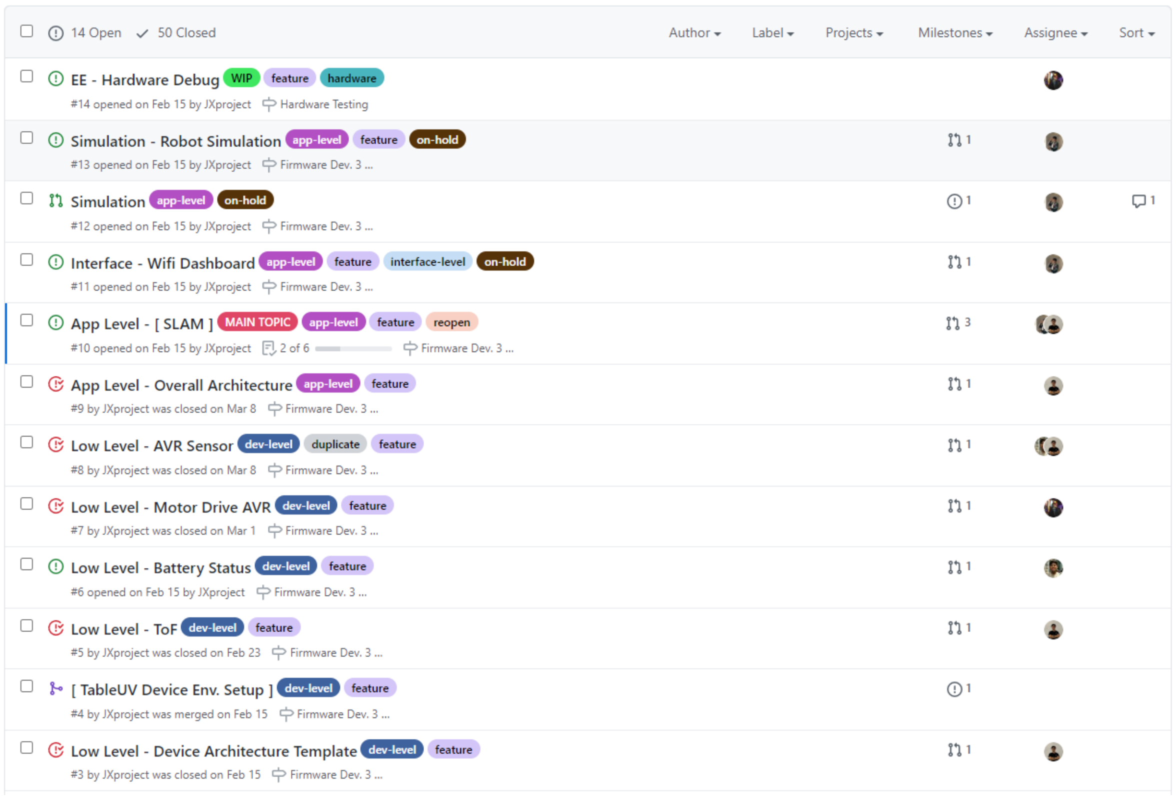

Firmware Development & Integration
2020. Dec
2021. Jan
2021. Feb
2021. Mar
2021. Apr
Prototype Manufacturing & Iterative Design
Application Level Integration

Figure 3-1. Timeline of high-level tasks done in MTE 482.

In addition to this, we heavily relied on Github issues to track the different tasks that we wanted to work on. This facilitated assigning tasks to different members of the group and also allowed for multi-tasking between different versions of the code using branches.



14 Open ✓ 50 Closed — Author ▾ Label ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

EE - Hardware Debug `WIP` `feature` `hardware`
#14 opened on Feb 15 by JXproject — Hardware Testing

Simulation - Robot Simulation `app-level` `feature` `on-hold` — 1
#13 opened on Feb 15 by JXproject — Firmware Dev. 3 ...

Simulation `app-level` `on-hold` — 1 — 1
#12 opened on Feb 15 by JXproject — Firmware Dev. 3 ...

Interface - Wifi Dashboard `app-level` `feature` `interface-level` `on-hold` — 1
#11 opened on Feb 15 by JXproject — Firmware Dev. 3 ...

App Level - [SLAM] `MAIN TOPIC` `app-level` `feature` `reopen` — 3
#10 opened on Feb 15 by JXproject — 2 of 6 — Firmware Dev. 3 ...

App Level - Overall Architecture `app-level` `feature` — 1
#9 by JXproject was closed on Mar 8 — Firmware Dev. 3 ...

Low Level - AVR Sensor `dev-level` `duplicate` `feature` — 1
#8 by JXproject was closed on Mar 8 — Firmware Dev. 3 ...

Low Level - Motor Drive AVR `dev-level` `feature` — 1
#7 by JXproject was closed on Mar 1 — Firmware Dev. 3 ...

Low Level - Battery Status `dev-level` `feature` — 1
#6 opened on Feb 15 by JXproject — Firmware Dev. 3 ...

Low Level - ToF `dev-level` `feature` — 1
#5 by JXproject was closed on Feb 23 — Firmware Dev. 3 ...

[TableUV Device Env. Setup] `dev-level` `feature` — 1
#4 by JXproject was merged on Feb 15 — Firmware Dev. 3 ...

Low Level - Device Architecture Template `dev-level` `feature` — 1
#3 by JXproject was closed on Feb 15 — Firmware Dev. 3 ...

Figure 3-2. Screenshot of the simulation environment in Webots.

3.2 Budget Evaluation

From the design constraint and criteria, the targeted project cost was to keep it under 300 CAD. The project costs are governed by the amount spent on the hardware, and this includes the mechanical components, 3D printing material cost, electrical components cost, and the PCB costs. From 481 report, the expected cost was calculated as 393.63 CAD. The expected cost was above the targeted cost due to the high cost of the sensors, specifically the UV LED. As UV LED effectiveness against virus and bacteria could not be verified due to pandemic, the UV LED with highest power dissipation was selected, and thus the most expensive one.

Table 3-1. Expected Cost Summary per robot [reference]

Domain	**Total Cost (CAD)**
Mech Total	68.1
Elec- Sensor Boards	218.38
Elec- MCU Boards	55.76
Elec- Driver Board	14.575
Elec- Power Board	36.78
TOTAL	393.63

The actual cost can easily be calculated by the cost spent on hardware. The full table is shown in the Appendix D, and the summarized table is shown below.

Table 3-2. Actual Cost Summary per robot [reference]

Domain	**Total Cost (CAD)**
Mech Total	68.1
Elec- Sensor Boards	245.46
Elec- MCU Boards	133.34
Elec- Driver and Power Board	68.495
Elec- Charger Board	60.1
TOTAL	575.495

From the table above the actual cost is much higher than the expected cost. This is because the expected cost calculated was before the design was completed, and thus the cost was calculated just with the main IC. In the actual design, the board contains the PCB cost, connector, cable, passive components such as resistor,capacitor, and inductor. Each items are low cost but summed up together will have significant effect on the total cost. Again to reemphasize, the sensor cost is the most expensive component in the design. During the winter term, pandemic did not settle and the UV LED effectiveness experiment could not be conducted. Hence the highest power dissipated UV LED is still kept which brings up the total cost.

4 Conclusions and Recommendations

4.1 Conclusion

The design thus far is able to meet most of the constraints and criteria that has been set at the beginning of the FYDP. Starting off with constraints, the robot is expected to disinfect a standard table in one charge with a combination of the UV LEDs, the piezo disinfectant sprayer and the motors; all of which were able to consume less capacity than a full battery. The robot is able to detect obstacles and maneuver around them accordingly by utilizing the ToF sensor, bumper sensor, appropriate mapping, and localization. The robot is able to stay within the tabletop boundaries with the help of IR sensors to detect the edge. The robot is able to support the appropriate load as proven in motor load calculations and physical testing. Even if budget has costed a lot more than expected, there are still evident improvements to be made that can ensure the cost of production is cheap enough to be considered viable on the market. Lastly, although it was not properly tested, the robot should be able to sanitize bacteria at 99% with support from research paper claims and the assistance of Professor Aucoin.

Overall, this report clearly outlines all the work that was done this term, including the modifications made to the design, the manufacturing, the commissioning, and the verification processes. It also summarizes the group's timeline, financial expenses, and all the teamwork required to get this complex system running. With this, although there is still room for improvement, all the major development for this project has come to a close.

4.2 Future Recommendation

Although the designed robot was able to meet most of the criteria and constraint set, further improvements can be made to enhance the performance and to optimize the design.

4.2.1 UV LED Effectiveness Validation

Validating the UV LED effectiveness against bacteria and virus has been brought up from the beginning of MTE 481. However due to the pandemic, lab verification was infeasible which led to the use of the highest power dissipating UV LED and thus the most expensive one. Therefore it is recommended to conduct the lab verification to determine the required UV LED power which can possible reduce the cost of production.

4.2.2 Redesign hardware architecture

One challenge the team encountered during the firmware debugging was with the use of 8 bit AVRs. This secondary MCU needs to be flashed using the setup shown in Paragraph 2.3.1.2, and can't directly interface with computers unlike the main MCU, ESP32. This made debugging extremely difficult, and due to the limited number of GPIO on the main MCU, 3 AVRs were used in the design. Thus it is recommend to revise the hardware architecture to use 2 ESP32 instead of 1 ESP32 with 3 AVR MCU. This will make debugging much simpler, and architecture wise much simpler as primary and secondary MCU uses the same processor. In addition, the additional ESP32 can helps to offload all device level tasks in a separate core, allowing an extra core on the high level side for wireless communication (such as remote streaming).

4.2.3 Advanced path planning and SLAM

For the software, it is recommended to improve the path planning and object avoidance algorithm by implementing SLAM. Currently due to the time and computation constraint, the global map created is not used to generate path and to localize the robot position. With SLAM implemented, the robot will be able to sanitize table more efficiently which can reduce cleaning time and save battery consumption.

4.2.4 A Better Manufacturing Process

After the redesign, the water tank still has small leakage at the exhale due to the uneven surface finishes when removing the support materials. Hence, a better manufacturing process is needed to provide a leakage free contact between the ultrasonic actuator and water reservoir.

5 Teamwork Effort

Overall, the team worked very well together as most of the members could perform work to their strengths while also applying their skills to help each other out. Besides, the team managed to split tasks and work in parallel with maximal efforts. Fortunately, the group could come together and work in the same room for most of the second term allowing for efficient and lengthy work sessions. All of the members used their reading week and spare time to meet together and work each day for almost 10 hours a day. These sessions were a mix of moments of collaborative discussion, learning, and focused development. With this, we were able to completely disregard the negatives of collaborating over online calls and maximizing our efforts in person.

5.1 Summary of Group member efforts

Jianxiang (Jack) Xu:

- Mechanical development and manufacturing
- Website development
- Firmware development
- Software development
- Presentation / Video / Final Report

Tsugumi Murata:

- Hardware development
- Hardware bring-up and bench testing
- Hardware validation and debugging
- Firmware development
- Presentation / Video / Final Report

Jerome Villapando:

- Hardware development
- PCB manufacturing
- Hardware bring-up and bench testing
- Hardware validation and debugging
- Firmware development
- Presentation / Video / Final Report

Dong Jae (Alex) Park:

- Sensor Testing
- Website development
- Software development
- Firmware development
- Presentation / Video / Final Report

Glossary

CAD Computer-Aided Design.
COVID Coronavirus.

DAC Digital-to-Analog Converter.

ESP32 ESP32.

FYDP Fourth Year Design Project.

IMU Inertial measurement unit.
IR Infrared.

LED Light Emitting Diode.
Lidar Lidar, which stands for Light Detection and Ranging.

MCU Micro-controller Unit.

OTA Over the air.

PCB Printed Circuit Board.

ROI Region of Interest.
Roomba Roomba Robot.
RPM Rate per minute.

SLAM Simultaneous Localization and Mapping.
SPICE Simulation Program with Integrated Circuit Emphasis.

ToF Time of Flight.

UV Ultra-violet.

References

[1] *Antibiotic resistance threats in the united states*. [Online]. Available: https://stacks.cdc.gov/view/cdc/82532.

[2] G. o. C. News, *Coronavirus disease (covid-19): Summary of assumptions*, 2020. [Online]. Available: https://www.canada.ca/en/public-health/services/diseases/2019-novel-coronavirus-infection/health-professionals/assumptions.html.

[3] L. A. T. News, *$160,000 to buy germ zapping robot for langley hospital*, 2020. [Online]. Available: https://www.langleyadvancetimes.com/community/160000-to-buy-germ-zapping-robot-for-langley-hospital/.

Appendix A   Mechanical Assembly

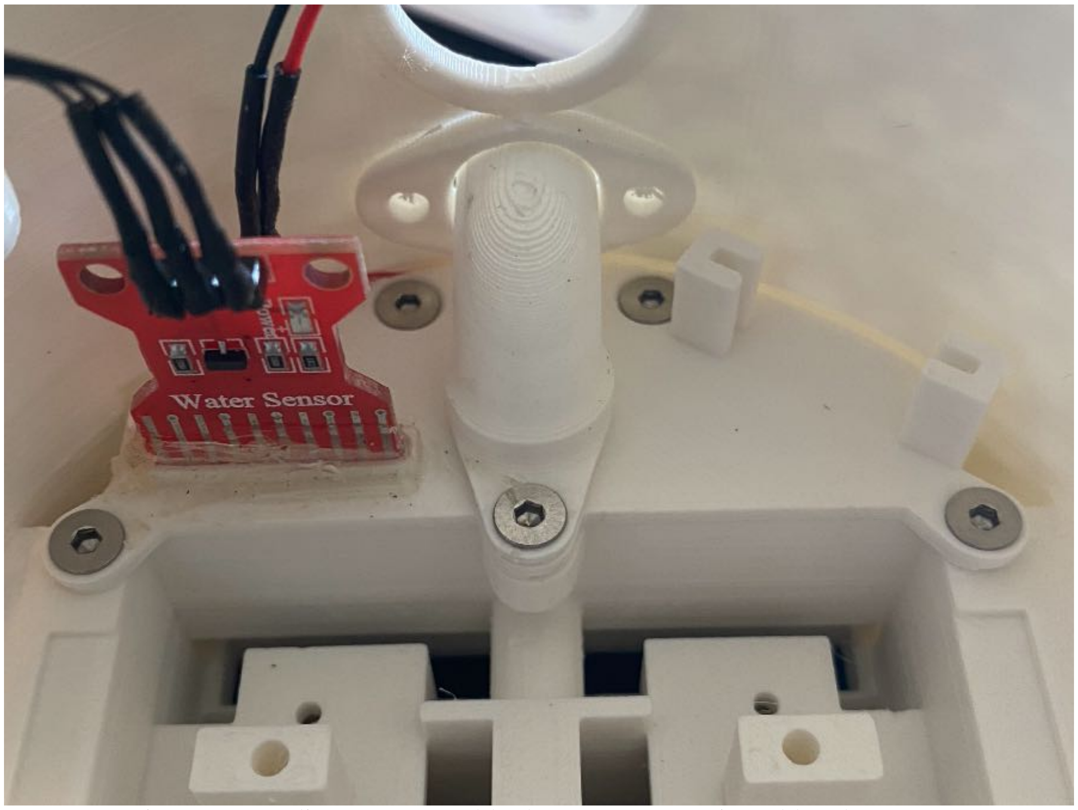


Figure Appendix A-1. Assemble the water reservoir and level sensors.

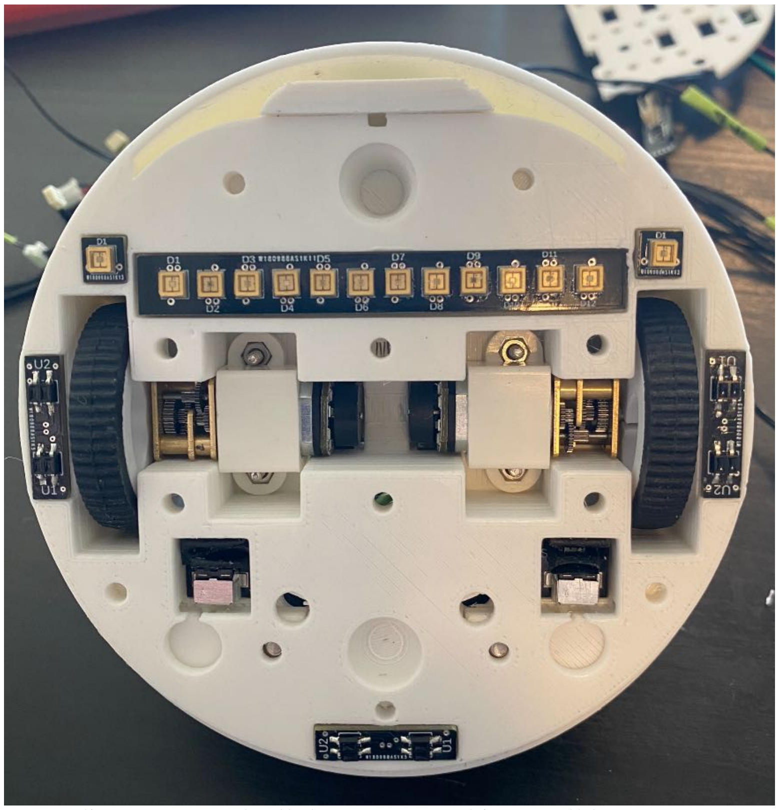


Figure Appendix A-2. Attach all the motors and insert all base sensors to the base

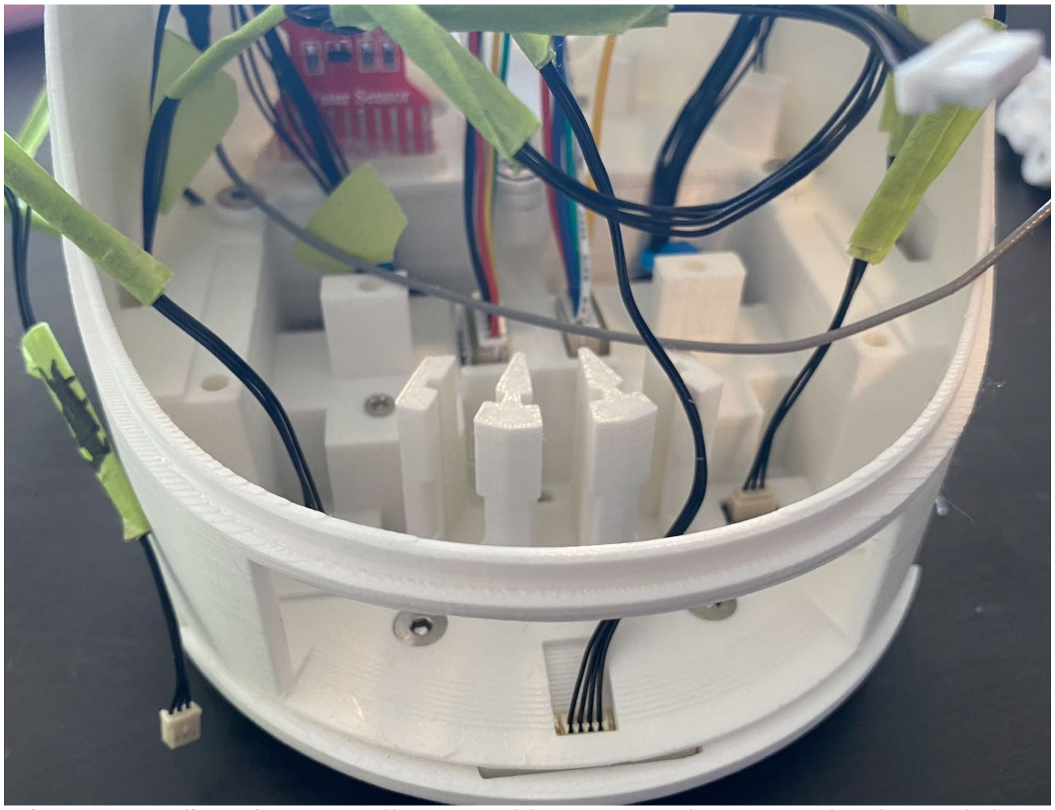


Figure Appendix A-3. Ensure all sensor cables are properly connected and extended out

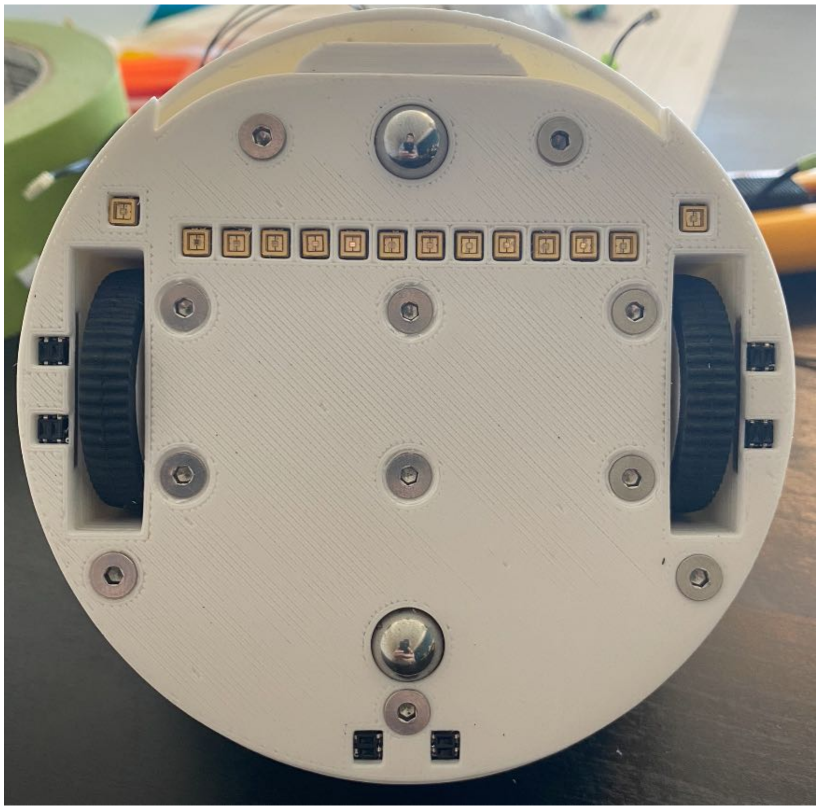


Figure Appendix A-4. Place down the ball casters and close the base sensor shield

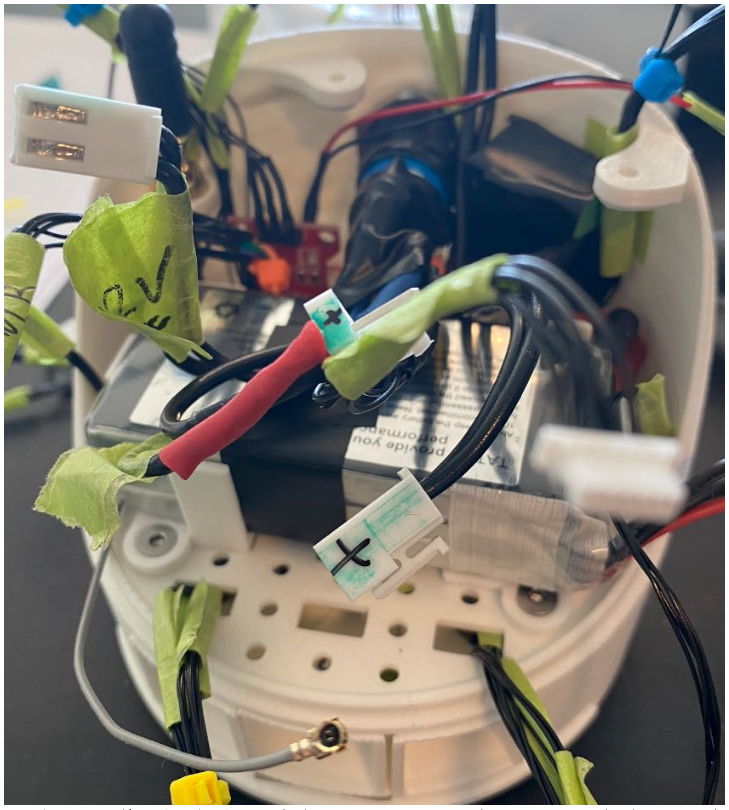


Figure Appendix A-5. Attach battery, power button, and charger board

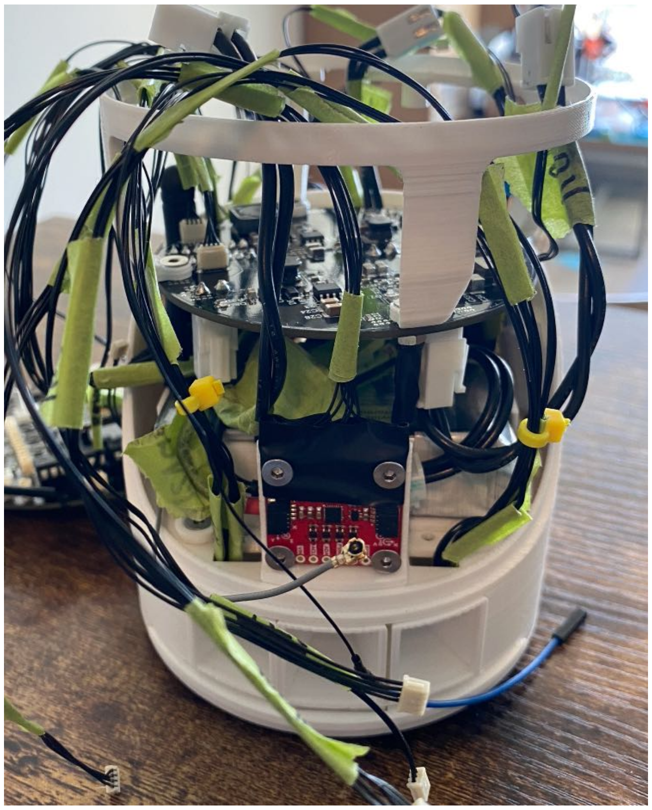


Figure Appendix A-6. Mount the power distribution board, IMU module, and MCU bracket

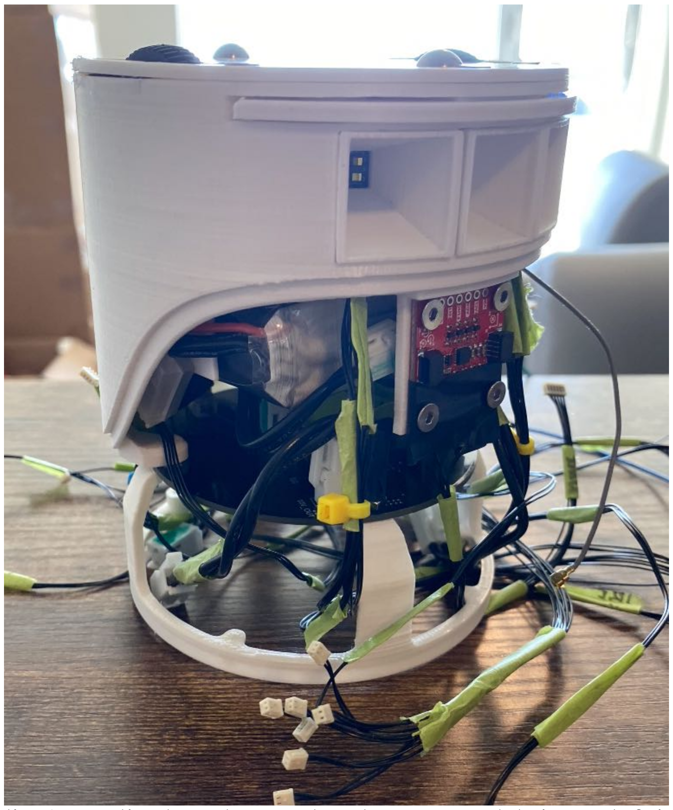


Figure Appendix A-7. Flip the robot, and make sure no debris are left inside the robot

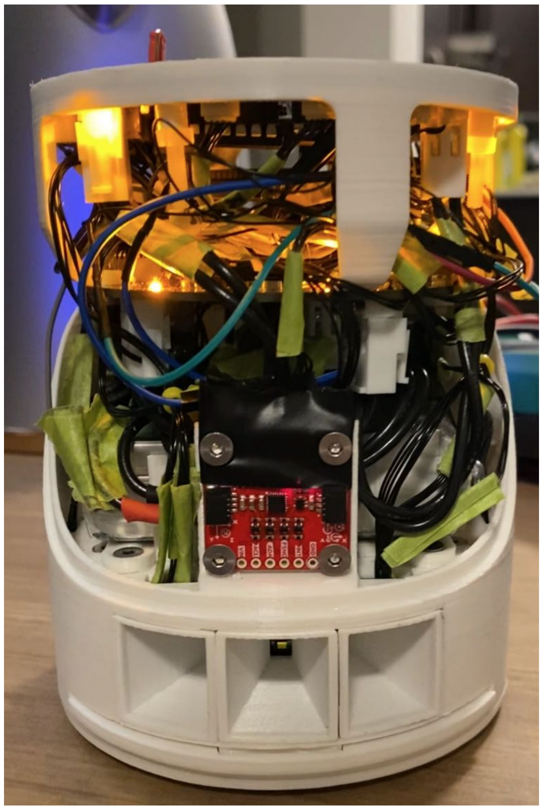


Figure Appendix A-8. Connect all sensor cables to the MCU, and mount the MCU with plastic screws, and do a quick power-on test

Appendix B PCB Design

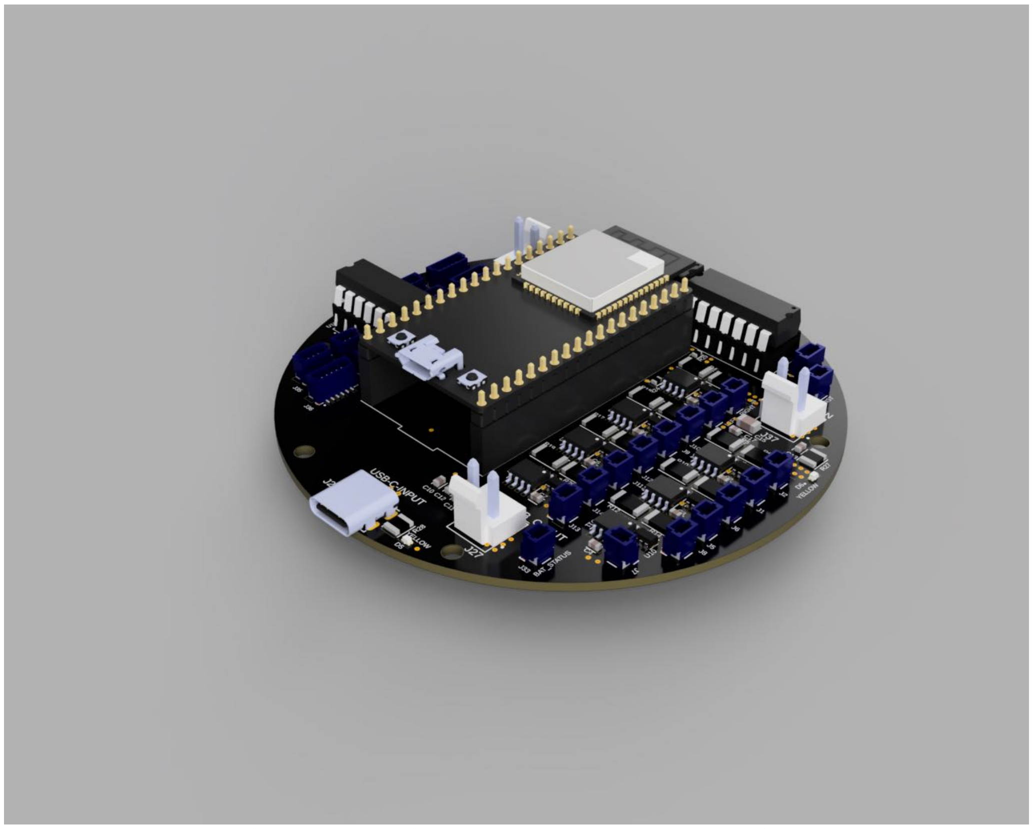


Figure Appendix B-1. MCU Board PCB Design

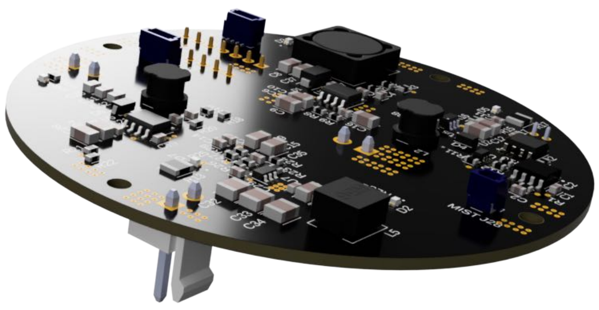


Figure Appendix B-2. Power and Driver Board PCB Design

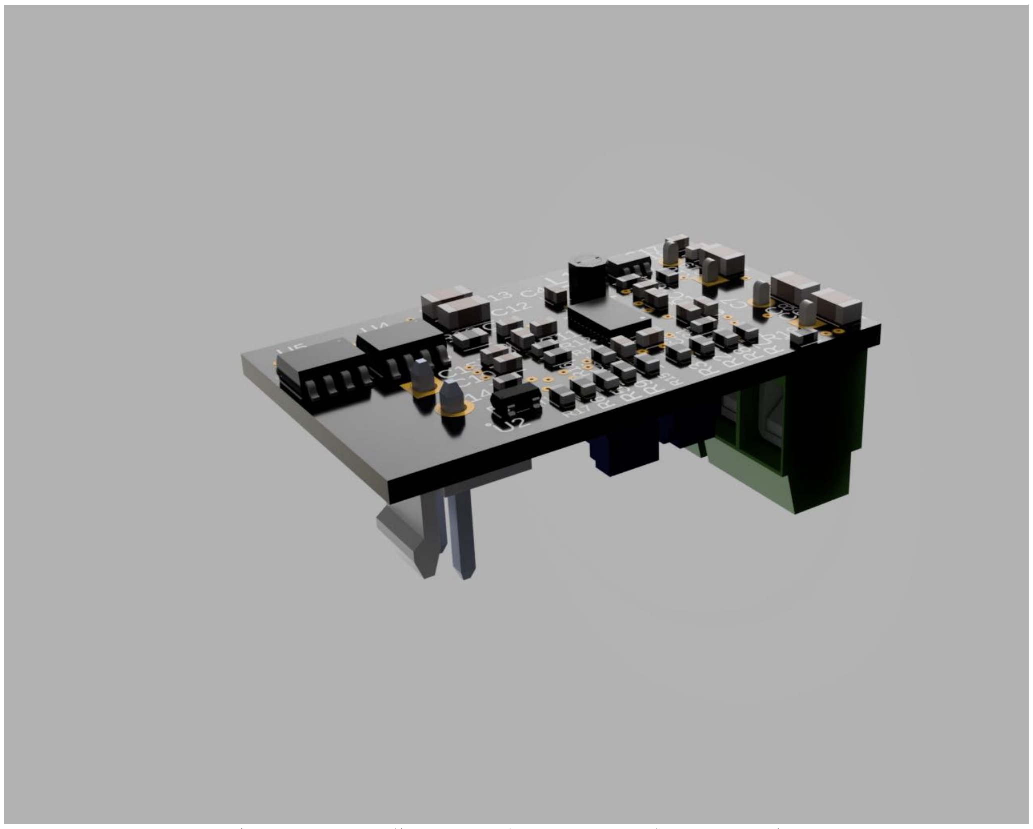


Figure Appendix B-3. Charger Board PCB Design

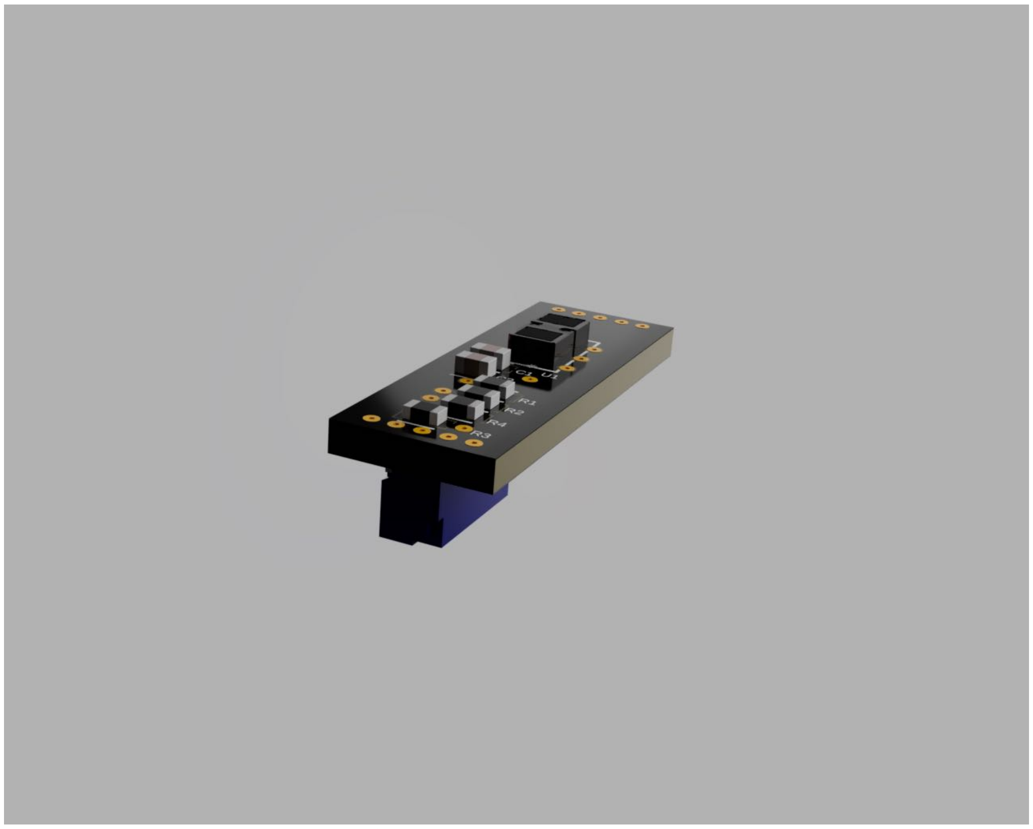


Figure Appendix B-4. TOF Sensor Board PCB Design

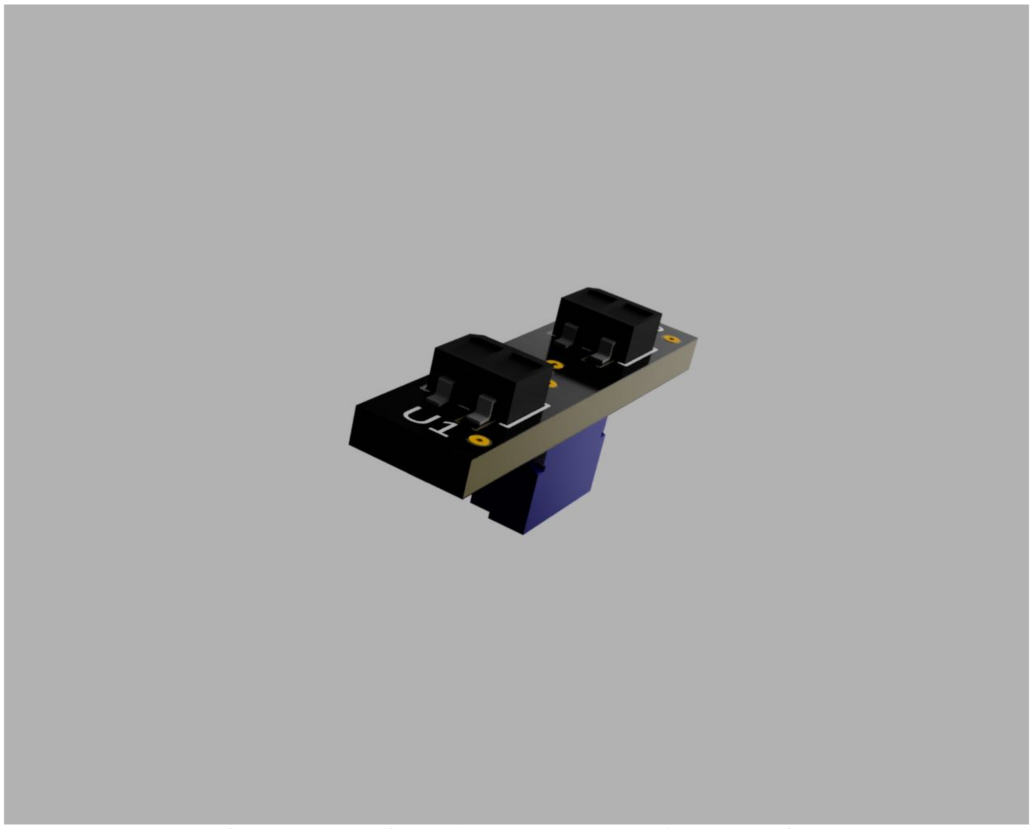


Figure Appendix B-5. IR Sensor Board PCB Design

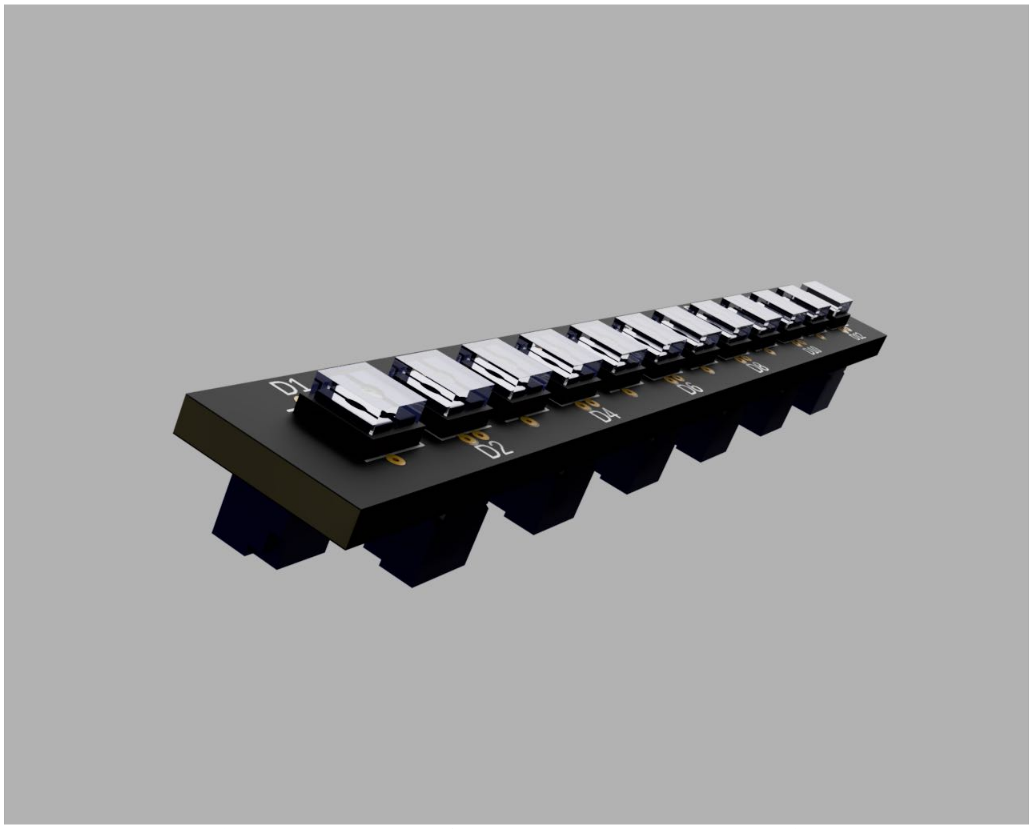


Figure Appendix B-6. UV Sensor Board PCB Design

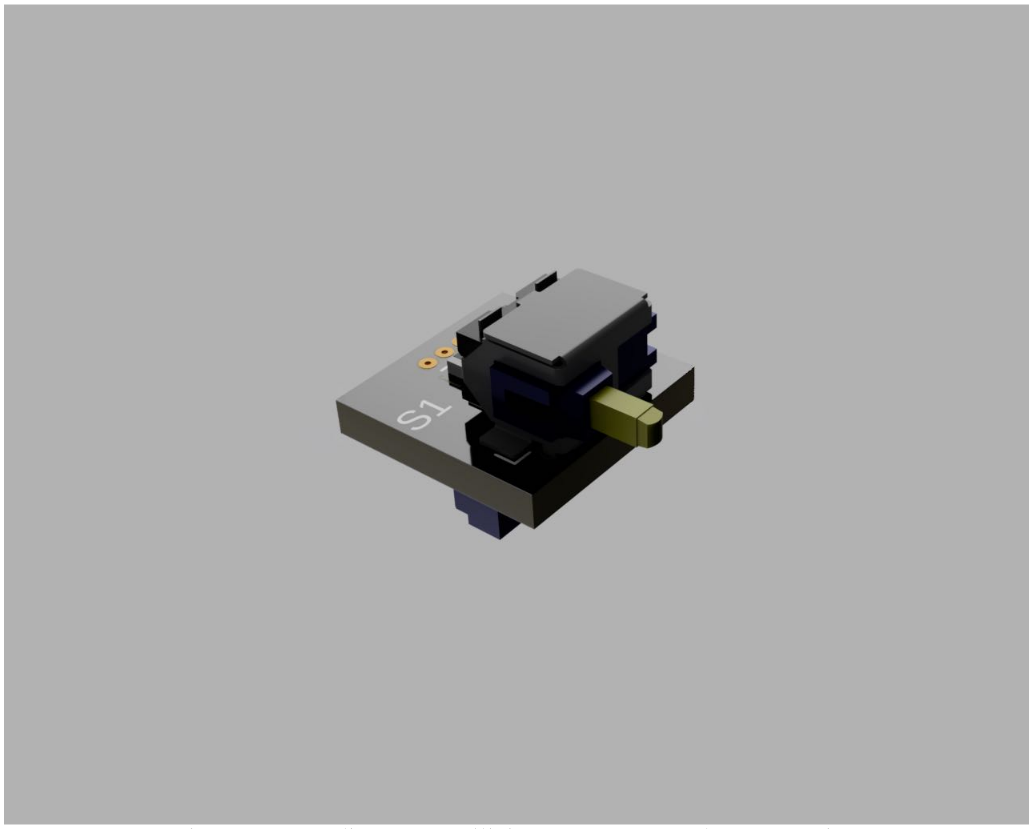
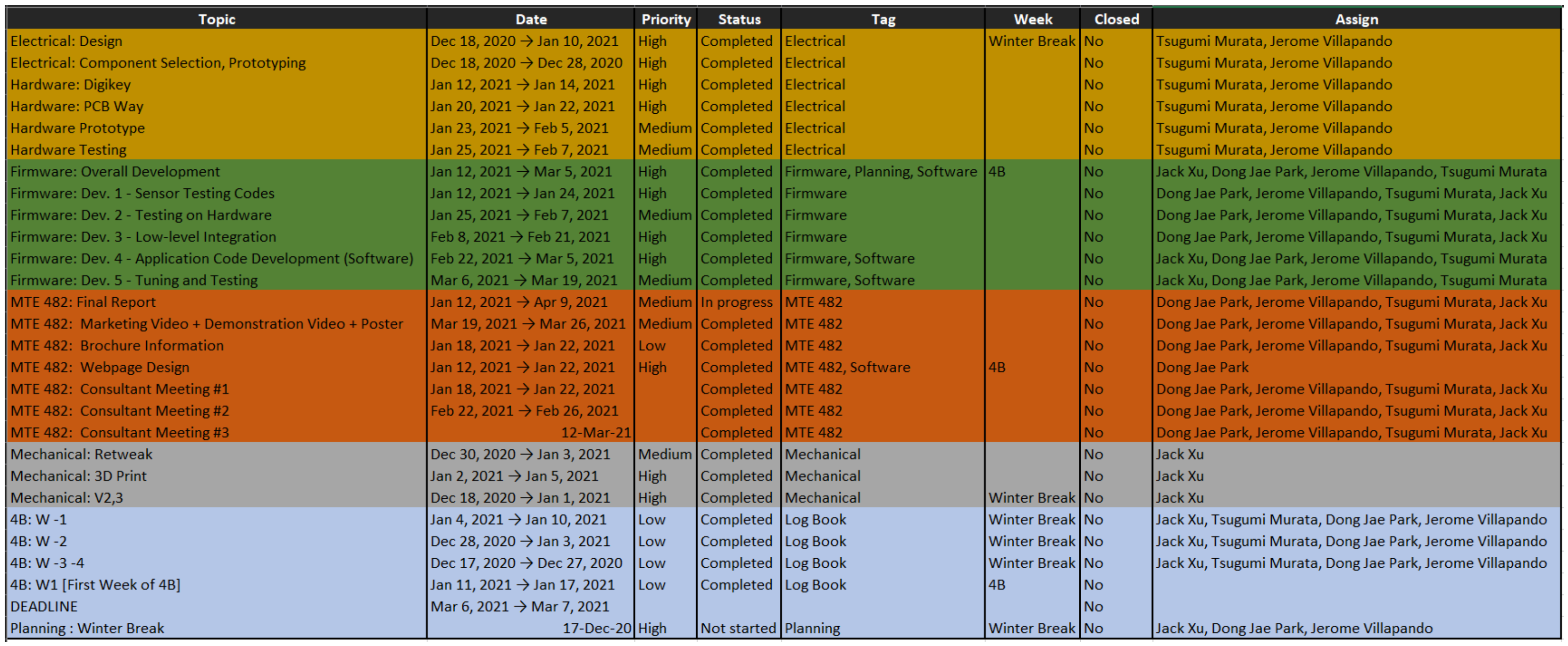


Figure Appendix B-7. Collision Sensor Board PCB Design

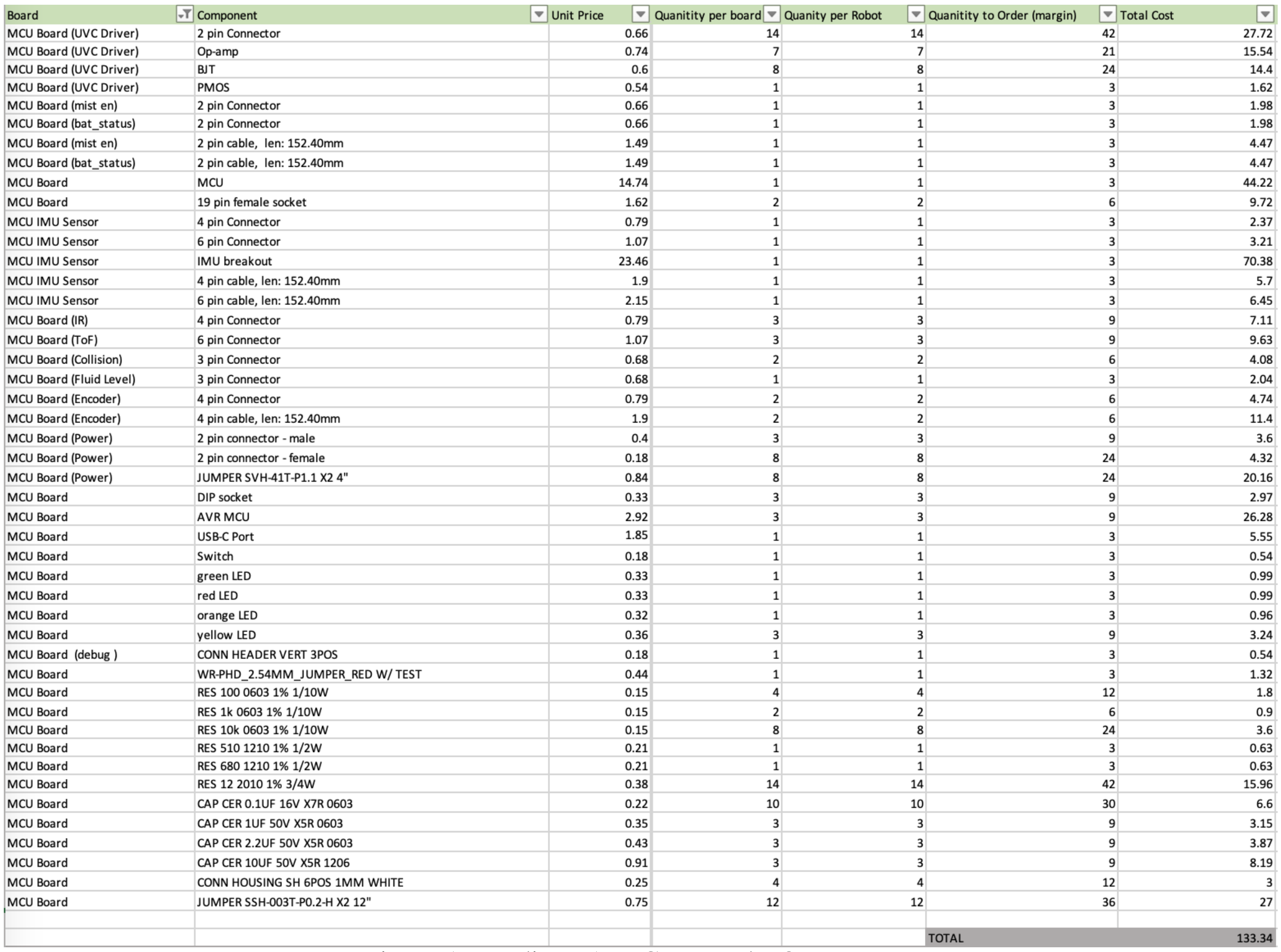
Appendix C Schedule



Topic	Date	Priority	Status	Tag	Week	Closed	Assign
Electrical: Design	Dec 18, 2020 → Jan 10, 2021	High	Completed	Electrical	Winter Break	No	Tsugumi Murata, Jerome Villapando
Electrical: Component Selection, Prototyping	Dec 18, 2020 → Dec 28, 2020	High	Completed	Electrical		No	Tsugumi Murata, Jerome Villapando
Hardware: Digikey	Jan 12, 2021 → Jan 14, 2021	High	Completed	Electrical		No	Tsugumi Murata, Jerome Villapando
Hardware: PCB Way	Jan 20, 2021 → Jan 22, 2021	High	Completed	Electrical		No	Tsugumi Murata, Jerome Villapando
Hardware Prototype	Jan 23, 2021 → Feb 5, 2021	Medium	Completed	Electrical		No	Tsugumi Murata, Jerome Villapando
Hardware Testing	Jan 25, 2021 → Feb 7, 2021	Medium	Completed	Electrical		No	Tsugumi Murata, Jerome Villapando
Firmware: Overall Development	Jan 12, 2021 → Mar 5, 2021	High	Completed	Firmware, Planning, Software	4B	No	Jack Xu, Dong Jae Park, Jerome Villapando, Tsugumi Murata
Firmware: Dev. 1 - Sensor Testing Codes	Jan 12, 2021 → Jan 24, 2021	High	Completed	Firmware		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
Firmware: Dev. 2 - Testing on Hardware	Jan 25, 2021 → Feb 7, 2021	Medium	Completed	Firmware		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
Firmware: Dev. 3 - Low-level Integration	Feb 8, 2021 → Feb 21, 2021	High	Completed	Firmware		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
Firmware: Dev. 4 - Application Code Development (Software)	Feb 22, 2021 → Mar 5, 2021	High	Completed	Firmware, Software		No	Jack Xu, Dong Jae Park, Jerome Villapando, Tsugumi Murata
Firmware: Dev. 5 - Tuning and Testing	Mar 6, 2021 → Mar 19, 2021	Medium	Completed	Firmware, Software		No	Jack Xu, Dong Jae Park, Jerome Villapando, Tsugumi Murata
MTE 482: Final Report	Jan 12, 2021 → Apr 9, 2021	Medium	In progress	MTE 482		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
MTE 482: Marketing Video + Demonstration Video + Poster	Mar 19, 2021 → Mar 26, 2021	Medium	Completed	MTE 482		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
MTE 482: Brochure Information	Jan 18, 2021 → Jan 22, 2021	Low	Completed	MTE 482		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
MTE 482: Webpage Design	Jan 12, 2021 → Jan 22, 2021	High	Completed	MTE 482, Software	4B	No	Dong Jae Park
MTE 482: Consultant Meeting #1	Jan 18, 2021 → Jan 22, 2021		Completed	MTE 482		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
MTE 482: Consultant Meeting #2	Feb 22, 2021 → Feb 26, 2021		Completed	MTE 482		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
MTE 482: Consultant Meeting #3	12-Mar-21		Completed	MTE 482		No	Dong Jae Park, Jerome Villapando, Tsugumi Murata, Jack Xu
Mechanical: Retweak	Dec 30, 2020 → Jan 3, 2021	Medium	Completed	Mechanical		No	Jack Xu
Mechanical: 3D Print	Jan 2, 2021 → Jan 5, 2021	High	Completed	Mechanical		No	Jack Xu
Mechanical: V2,3	Dec 18, 2020 → Jan 1, 2021	High	Completed	Mechanical	Winter Break	No	Jack Xu
4B: W -1	Jan 4, 2021 → Jan 10, 2021	Low	Completed	Log Book	Winter Break	No	Jack Xu, Tsugumi Murata, Dong Jae Park, Jerome Villapando
4B: W -2	Dec 28, 2020 → Jan 3, 2021	Low	Completed	Log Book	Winter Break	No	Jack Xu, Tsugumi Murata, Dong Jae Park, Jerome Villapando
4B: W -3 -4	Dec 17, 2020 → Dec 27, 2020	Low	Completed	Log Book	Winter Break	No	Jack Xu, Tsugumi Murata, Dong Jae Park, Jerome Villapando
4B: W1 [First Week of 4B]	Jan 11, 2021 → Jan 17, 2021	Low	Completed	Log Book	4B	No	
DEADLINE	Mar 6, 2021 → Mar 7, 2021					No	
Planning : Winter Break	17-Dec-20	High	Not started	Planning	Winter Break	No	Jack Xu, Dong Jae Park, Jerome Villapando

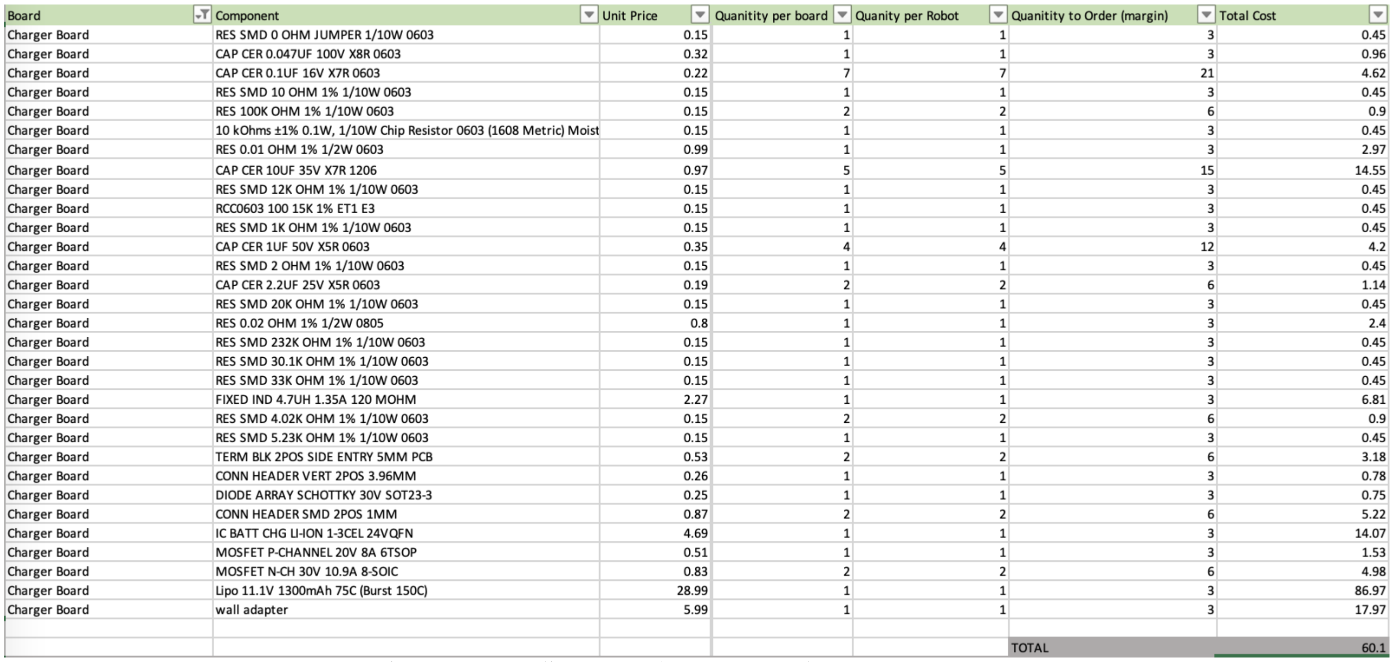
Figure Appendix C-1. Table displaying the initial scheduling of different tasks.

Appendix D   BOM



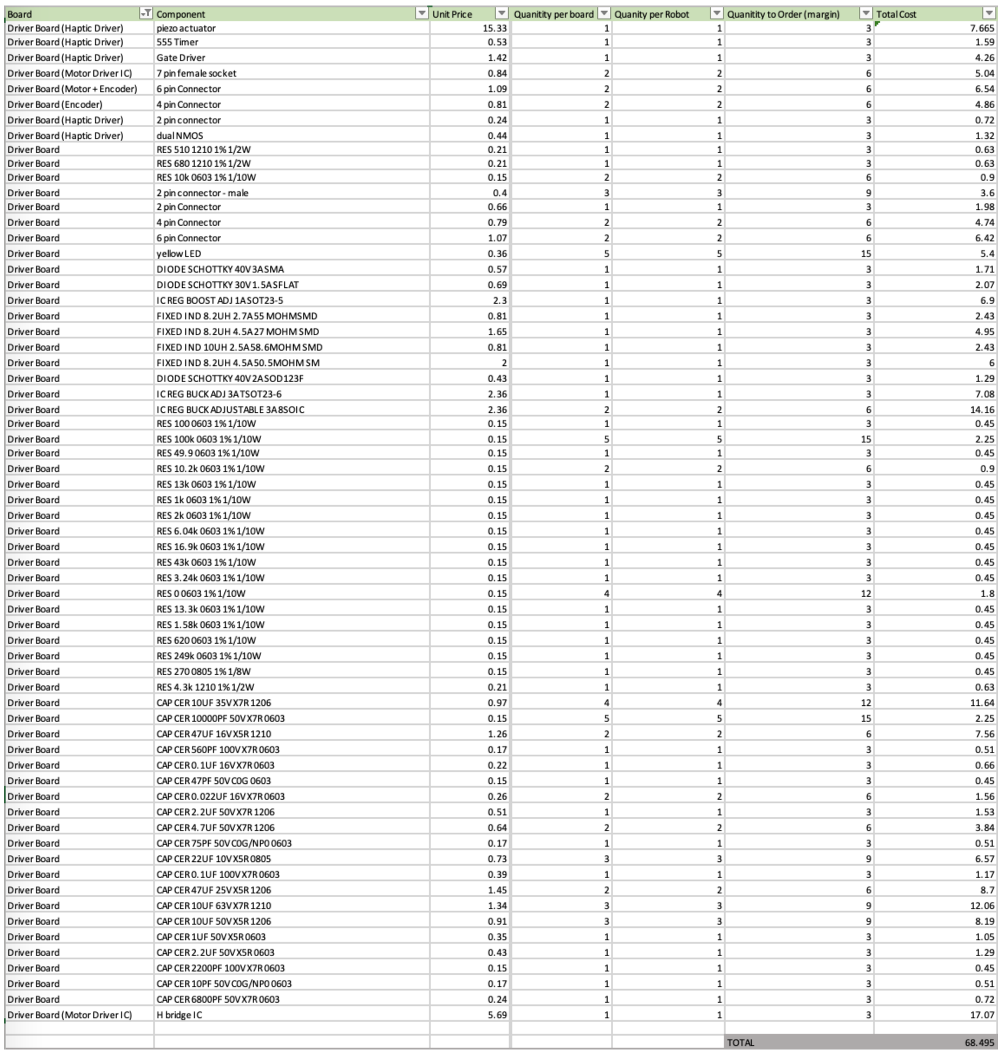
Board	Component	Unit Price	Quanitity per board	Quanity per Robot	Quanitity to Order (margin)	Total Cost
MCU Board (UVC Driver)	2 pin Connector	0.66	14	14	42	27.72
MCU Board (UVC Driver)	Op-amp	0.74	7	7	21	15.54
MCU Board (UVC Driver)	BJT	0.6	8	8	24	14.4
MCU Board (UVC Driver)	PMOS	0.54	1	1	3	1.62
MCU Board (mist en)	2 pin Connector	0.66	1	1	3	1.98
MCU Board (bat_status)	2 pin Connector	0.66	1	1	3	1.98
MCU Board (mist en)	2 pin cable, len: 152.40mm	1.49	1	1	3	4.47
MCU Board (bat_status)	2 pin cable, len: 152.40mm	1.49	1	1	3	4.47
MCU Board	MCU	14.74	1	1	3	44.22
MCU Board	19 pin female socket	1.62	2	2	6	9.72
MCU IMU Sensor	4 pin Connector	0.79	1	1	3	2.37
MCU IMU Sensor	6 pin Connector	1.07	1	1	3	3.21
MCU IMU Sensor	IMU breakout	23.46	1	1	3	70.38
MCU IMU Sensor	4 pin cable, len: 152.40mm	1.9	1	1	3	5.7
MCU IMU Sensor	6 pin cable, len: 152.40mm	2.15	1	1	3	6.45
MCU Board (IR)	4 pin Connector	0.79	3	3	9	7.11
MCU Board (ToF)	6 pin Connector	1.07	3	3	9	9.63
MCU Board (Collision)	3 pin Connector	0.68	2	2	6	4.08
MCU Board (Fluid Level)	3 pin Connector	0.68	1	1	3	2.04
MCU Board (Encoder)	4 pin Connector	0.79	2	2	6	4.74
MCU Board (Encoder)	4 pin cable, len: 152.40mm	1.9	2	2	6	11.4
MCU Board (Power)	2 pin connector - male	0.4	3	3	9	3.6
MCU Board (Power)	2 pin connector - female	0.18	8	8	24	4.32
MCU Board (Power)	JUMPER SVH-41T-P1.1 X2 4"	0.84	8	8	24	20.16
MCU Board	DIP socket	0.33	3	3	9	2.97
MCU Board	AVR MCU	2.92	3	3	9	26.28
MCU Board	USB-C Port	1.85	1	1	3	5.55
MCU Board	Switch	0.18	1	1	3	0.54
MCU Board	green LED	0.33	1	1	3	0.99
MCU Board	red LED	0.33	1	1	3	0.99
MCU Board	orange LED	0.32	1	1	3	0.96
MCU Board	yellow LED	0.36	3	3	9	3.24
MCU Board (debug)	CONN HEADER VERT 3POS	0.18	1	1	3	0.54
MCU Board	WR-PHD_2.54MM_JUMPER_RED W/ TEST	0.44	1	1	3	1.32
MCU Board	RES 100 0603 1% 1/10W	0.15	4	4	12	1.8
MCU Board	RES 1k 0603 1% 1/10W	0.15	2	2	6	0.9
MCU Board	RES 10k 0603 1% 1/10W	0.15	8	8	24	3.6
MCU Board	RES 510 1210 1% 1/2W	0.21	1	1	3	0.63
MCU Board	RES 680 1210 1% 1/2W	0.21	1	1	3	0.63
MCU Board	RES 12 2010 1% 3/4W	0.38	14	14	42	15.96
MCU Board	CAP CER 0.1UF 16V X7R 0603	0.22	10	10	30	6.6
MCU Board	CAP CER 1UF 50V X5R 0603	0.35	3	3	9	3.15
MCU Board	CAP CER 2.2UF 50V X5R 0603	0.43	3	3	9	3.87
MCU Board	CAP CER 10UF 50V X5R 1206	0.91	3	3	9	8.19
MCU Board	CONN HOUSING SH 6POS 1MM WHITE	0.25	4	4	12	3
MCU Board	JUMPER SSH-003T-P0.2-H X2 12"	0.75	12	12	36	27
					TOTAL	133.34

Figure Appendix D-1. MCU Board BOM



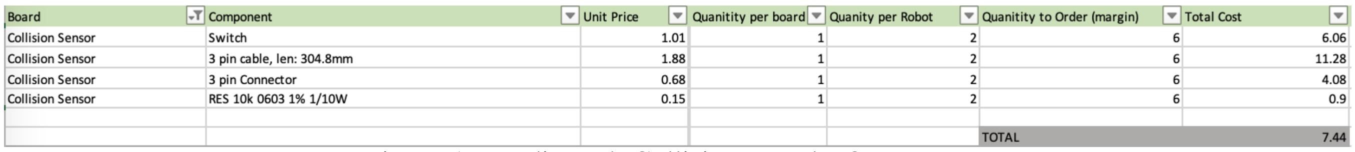
Board	Component	Unit Price	Quanitity per board	Quanity per Robot	Quanitity to Order (margin)	Total Cost
Charger Board	RES SMD 0 OHM JUMPER 1/10W 0603	0.15	1	1	3	0.45
Charger Board	CAP CER 0.047UF 100V X8R 0603	0.32	1	1	3	0.96
Charger Board	CAP CER 0.1UF 16V X7R 0603	0.22	7	7	21	4.62
Charger Board	RES SMD 10 OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	RES 100K OHM 1% 1/10W 0603	0.15	2	2	6	0.9
Charger Board	10 kOhms ±1% 0.1W, 1/10W Chip Resistor 0603 (1608 Metric) Moist	0.15	1	1	3	0.45
Charger Board	RES 0.01 OHM 1% 1/2W 0603	0.99	1	1	3	2.97
Charger Board	CAP CER 10UF 35V X7R 1206	0.97	5	5	15	14.55
Charger Board	RES SMD 12K OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	RCC0603 100 15K 1% ET1 E3	0.15	1	1	3	0.45
Charger Board	RES SMD 1K OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	CAP CER 1UF 50V X5R 0603	0.35	4	4	12	4.2
Charger Board	RES SMD 2 OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	CAP CER 2.2UF 25V X5R 0603	0.19	2	2	6	1.14
Charger Board	RES SMD 20K OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	RES 0.02 OHM 1% 1/2W 0805	0.8	1	1	3	2.4
Charger Board	RES SMD 232K OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	RES SMD 30.1K OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	RES SMD 33K OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	FIXED IND 4.7UH 1.35A 120 MOHM	2.27	1	1	3	6.81
Charger Board	RES SMD 4.02K OHM 1% 1/10W 0603	0.15	2	2	6	0.9
Charger Board	RES SMD 5.23K OHM 1% 1/10W 0603	0.15	1	1	3	0.45
Charger Board	TERM BLK 2POS SIDE ENTRY 5MM PCB	0.53	2	2	6	3.18
Charger Board	CONN HEADER VERT 2POS 3.96MM	0.26	1	1	3	0.78
Charger Board	DIODE ARRAY SCHOTTKY 30V SOT23-3	0.25	1	1	3	0.75
Charger Board	CONN HEADER SMD 2POS 1MM	0.87	2	2	6	5.22
Charger Board	IC BATT CHG LI-ION 1-3CEL 24VQFN	4.69	1	1	3	14.07
Charger Board	MOSFET P-CHANNEL 20V 8A 6TSOP	0.51	1	1	3	1.53
Charger Board	MOSFET N-CH 30V 10.9A 8-SOIC	0.83	2	2	6	4.98
Charger Board	Lipo 11.1V 1300mAh 75C (Burst 150C)	28.99	1	1	3	86.97
Charger Board	wall adapter	5.99	1	1	3	17.97
					TOTAL	60.1

Figure Appendix D-2. Charger Board BOM



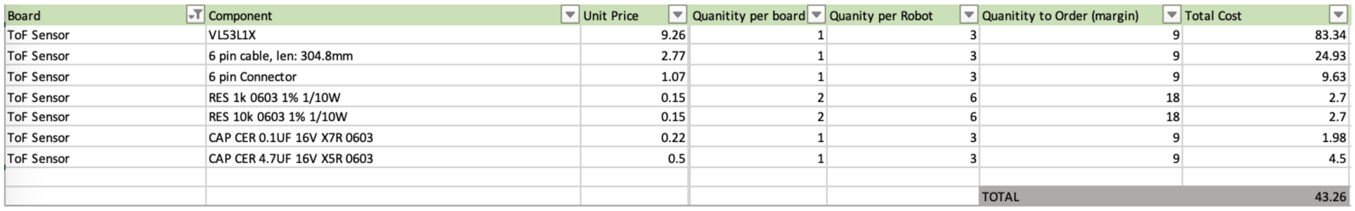
Board	Component	Unit Price	Quanitity per board	Quanity per Robot	Quanitity to Order (margin)	Total Cost
Driver Board (Haptic Driver)	piezo actuator	15.33	1	1	3	7.665
Driver Board (Haptic Driver)	555 Timer	0.53	1	1	3	1.59
Driver Board (Haptic Driver)	Gate Driver	1.42	1	1	3	4.26
Driver Board (Motor Driver IC)	7 pin female socket	0.84	2	2	6	5.04
Driver Board (Motor + Encoder)	6 pin Connector	1.09	2	2	6	6.54
Driver Board (Encoder)	4 pin Connector	0.81	2	2	6	4.86
Driver Board (Haptic Driver)	2 pin connector	0.24	1	1	3	0.72
Driver Board (Haptic Driver)	dual NMOS	0.44	1	1	3	1.32
Driver Board	RES 510 1210 1% 1/2W	0.21	1	1	3	0.63
Driver Board	RES 680 1210 1% 1/2W	0.21	1	1	3	0.63
Driver Board	RES 10k 0603 1% 1/10W	0.15	2	2	6	0.9
Driver Board	2 pin connector - male	0.4	3	3	9	3.6
Driver Board	2 pin Connector	0.66	1	1	3	1.98
Driver Board	4 pin Connector	0.79	2	2	6	4.74
Driver Board	6 pin Connector	1.07	2	2	6	6.42
Driver Board	yellow LED	0.36	5	5	15	5.4
Driver Board	DIODE SCHOTTKY 40V 3A SMA	0.57	1	1	3	1.71
Driver Board	DIODE SCHOTTKY 30V 1.5A SFLAT	0.69	1	1	3	2.07
Driver Board	IC REG BOOST ADJ 1A SOT23-5	2.3	1	1	3	6.9
Driver Board	FIXED IND 8.2UH 2.7A 55 MOHM SMD	0.81	1	1	3	2.43
Driver Board	FIXED IND 8.2UH 4.5A 27 MOHM SMD	1.65	1	1	3	4.95
Driver Board	FIXED IND 10UH 2.5A 58.6MOHM SMD	0.81	1	1	3	2.43
Driver Board	FIXED IND 8.2UH 4.5A 50.5MOHM SM	2	1	1	3	6
Driver Board	DIODE SCHOTTKY 40V 2A SOD123F	0.43	1	1	3	1.29
Driver Board	IC REG BUCK ADJ 3A TSOT23-6	2.36	1	1	3	7.08
Driver Board	IC REG BUCK ADJUSTABLE 3A 8SOIC	2.36	2	2	6	14.16
Driver Board	RES 100 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 100k 0603 1% 1/10W	0.15	5	5	15	2.25
Driver Board	RES 49.9 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 10.2k 0603 1% 1/10W	0.15	2	2	6	0.9
Driver Board	RES 13k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 1k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 2k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 6.04k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 16.9k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 43k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 3.24k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 0 0603 1% 1/10W	0.15	4	4	12	1.8
Driver Board	RES 13.3k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 1.58k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 620 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 249k 0603 1% 1/10W	0.15	1	1	3	0.45
Driver Board	RES 270 0805 1% 1/8W	0.15	1	1	3	0.45
Driver Board	RES 4.3k 1210 1% 1/2W	0.21	1	1	3	0.63
Driver Board	CAP CER 10UF 35V X7R 1206	0.97	4	4	12	11.64
Driver Board	CAP CER 10000PF 50V X7R 0603	0.15	5	5	15	2.25
Driver Board	CAP CER 47UF 16V X5R 1210	1.26	2	2	6	7.56
Driver Board	CAP CER 560PF 100V X7R 0603	0.17	1	1	3	0.51
Driver Board	CAP CER 0.1UF 16V X7R 0603	0.22	1	1	3	0.66
Driver Board	CAP CER 47PF 50V C0G 0603	0.15	1	1	3	0.45
Driver Board	CAP CER 0.022UF 16V X7R 0603	0.26	2	2	6	1.56
Driver Board	CAP CER 2.2UF 50V X7R 1206	0.51	1	1	3	1.53
Driver Board	CAP CER 4.7UF 50V X7R 1206	0.64	2	2	6	3.84
Driver Board	CAP CER 75PF 50V C0G/NP0 0603	0.17	1	1	3	0.51
Driver Board	CAP CER 22UF 10V X5R 0805	0.73	3	3	9	6.57
Driver Board	CAP CER 0.1UF 100V X7R 0603	0.39	1	1	3	1.17
Driver Board	CAP CER 47UF 25V X5R 1206	1.45	2	2	6	8.7
Driver Board	CAP CER 10UF 63V X7R 1210	1.34	3	3	9	12.06
Driver Board	CAP CER 10UF 50V X5R 1206	0.91	3	3	9	8.19
Driver Board	CAP CER 1UF 50V X5R 0603	0.35	1	1	3	1.05
Driver Board	CAP CER 2.2UF 50V X5R 0603	0.43	1	1	3	1.29
Driver Board	CAP CER 2200PF 100V X7R 0603	0.15	1	1	3	0.45
Driver Board	CAP CER 10PF 50V C0G/NP0 0603	0.17	1	1	3	0.51
Driver Board	CAP CER 6800PF 50V X7R 0603	0.24	1	1	3	0.72
Driver Board (Motor Driver IC)	H bridge IC	5.69	1	1	3	17.07
					TOTAL	68.495

Figure Appendix D-3. Power and Driver Board BOM



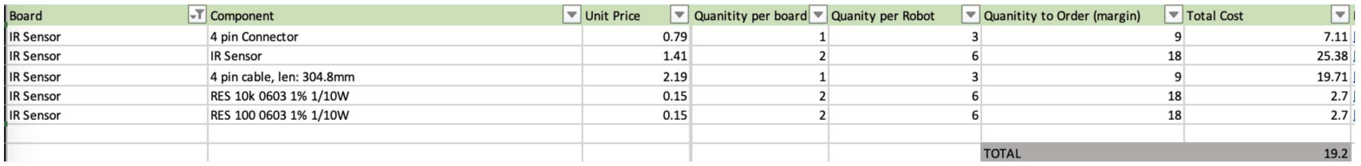
Board	Component	Unit Price	Quanitity per board	Quanity per Robot	Quanitity to Order (margin)	Total Cost
Collision Sensor	Switch	1.01	1	2	6	6.06
Collision Sensor	3 pin cable, len: 304.8mm	1.88	1	2	6	11.28
Collision Sensor	3 pin Connector	0.68	1	2	6	4.08
Collision Sensor	RES 10k 0603 1% 1/10W	0.15	1	2	6	0.9
					TOTAL	7.44

Figure Appendix D-4. Collision Board BOM



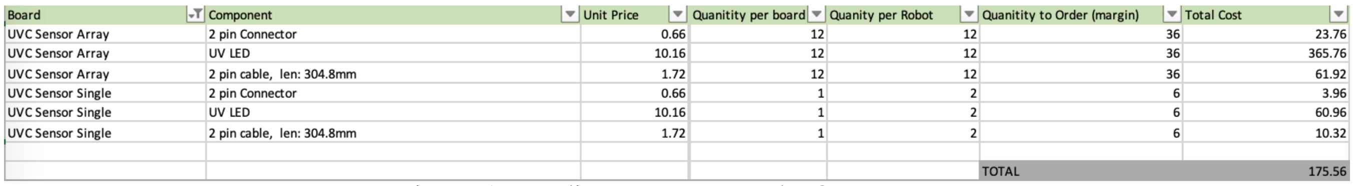
Board	Component	Unit Price	Quanitity per board	Quanity per Robot	Quanitity to Order (margin)	Total Cost
ToF Sensor	VL53L1X	9.26	1	3	9	83.34
ToF Sensor	6 pin cable, len: 304.8mm	2.77	1	3	9	24.93
ToF Sensor	6 pin Connector	1.07	1	3	9	9.63
ToF Sensor	RES 1k 0603 1% 1/10W	0.15	2	6	18	2.7
ToF Sensor	RES 10k 0603 1% 1/10W	0.15	2	6	18	2.7
ToF Sensor	CAP CER 0.1UF 16V X7R 0603	0.22	1	3	9	1.98
ToF Sensor	CAP CER 4.7UF 16V X5R 0603	0.5	1	3	9	4.5
					TOTAL	43.26

Figure Appendix D-5. TOF Board BOM



Board	Component	Unit Price	Quanitity per board	Quanity per Robot	Quanitity to Order (margin)	Total Cost
IR Sensor	4 pin Connector	0.79	1	3	9	7.11
IR Sensor	IR Sensor	1.41	2	6	18	25.38
IR Sensor	4 pin cable, len: 304.8mm	2.19	1	3	9	19.71
IR Sensor	RES 10k 0603 1% 1/10W	0.15	2	6	18	2.7
IR Sensor	RES 100 0603 1% 1/10W	0.15	2	6	18	2.7
					TOTAL	19.2

Figure Appendix D-6. IR Board BOM



Board	Component	Unit Price	Quanitity per board	Quanity per Robot	Quanitity to Order (margin)	Total Cost
UVC Sensor Array	2 pin Connector	0.66	12	12	36	23.76
UVC Sensor Array	UV LED	10.16	12	12	36	365.76
UVC Sensor Array	2 pin cable, len: 304.8mm	1.72	12	12	36	61.92
UVC Sensor Single	2 pin Connector	0.66	1	2	6	3.96
UVC Sensor Single	UV LED	10.16	1	2	6	60.96
UVC Sensor Single	2 pin cable, len: 304.8mm	1.72	1	2	6	10.32
					TOTAL	175.56

Figure Appendix D-7. UV Board BOM