Autonomous Sanitation Robot

University of Central Florida
Department of Engineering and Computer Science
+
Dr. Lei Wei
EEL 4914: Senior Design II

Group 9:
Abel Assefa: Electrical Engineering
Shawn Manohar: Electrical Engineering
Rishi Patel: Electrical Engineering
Allan McCormick: Computer Engineering

Group 9 Customers & Contributors:
All members listed under Group 9

Table of Contents

List of Figures

List of Tables

1. Executive Summary

With the way things are now people are more concerned than ever about cleanliness. This recent COVID-19 pandemic has shown everyone how easy it is to spread germs in an environment. This has changed the mindset of everyone, even after this pandemic is over people be more health conscious than ever, being more mindful of the area they are in and thinking more about how often they should wash their hands. With areas of the highest foot traffic being hallways we need to maintain the cleanliness of hallways the most, although many people do not spend excessive time in hallways everyone must use them.

With this in mind we thought of the Autonomous Sanitation Robot. This is a self-driving robot that would help people such as building managers ensure that high traffic areas like hallways and lobbies are always sanitized and clean. By replacing the human with a robot, we have decreased the amount of time a given area would be dirty. This is possible with having the robot constantly patrolling an area and cleaning said area all day. Unlike with a user doing this task the robot would be designated to a specific area to patrol and keep clean.

The robot consists of a robotic chassis with two motors utilized for movement, a color sensor to recognize dirty areas to clean, one sensor to detect the environment and avoid users and obstacles through audio queues, and a motorized sprayer with a reservoir to house the sanitation fluid. The robot would patrol areas such as hallways using the motors and sensors, the sensor would then recognize the dirty areas and the command would be sent from the microcontroller to send a beep for the recognition of the dirty area until a set distance. When the set distance is utilized with the color sensor and trigger a beep to recognize the area, then the sprayer constantly sprays the sanitation fluid until the color sensor has deemed the surface to be clean. This color sensor was placed at the front of the robot along with the motorized sprayer.

This robot should be light enough for the average user to carry, be able to discern a dirty area from a clean one, be able to autonomously seek out dirty areas, and have a low fluid shut-off function that also notifies the user if there needs to be a refill to the disinfectant solution.

The closest product that would compete with the Autonomous Sanitation Robot would be the household Roomba. The major difference would be that this robot does not vacuum or mop floors. This robot also does not “randomly” clean such as a Roomba, the robot essentially autonomously “patrol” an area and if it recognizes a dirty surface it then cleans this area and continues on its “patrol.” This allows the robot to cut down on unnecessary sanitation. Utilizing ultrasonic sensors along with a color camera the robot is able to autonomously search out dirty areas in places such as hallways and proceed to disinfect these areas until clean.

2. Project Description

In this project we defined our requirements, deliverables, and general specifications that are going to make this project succeed. The general form of this project is an autonomous robot that patrols a designated area and cleans surfaces with a mounted sprayer that fires a cleaning solution.

2.1 Project Requirements and Specifications

* “The system” refers to the irobot and/or all attachments

2.1.1 Housing

- The system shall be no taller than 3.62 inches in height, 13.39 inches in diameter and not weigh more than 7.9 lbs.
- The system shall contain an irobot/chassis, spray-bottle, motor, sensor(s) sensor control(raspberry-pi) and microcontroller .
- The system has 10 safe drilling/mounting templates on the faceplate.

2.1.2 Power Consumption

- The system has power-in through a 120V wall outlet.

2.1.3 System Controller

- The serial cable in the system sends commands from a computer/microcontroller.
- The system's pre-programmed behaviors can be controlled via Open Interface Commands (OIC).
- The system has a communication cable to program irobot.
- The communication cable sends that data to spray aerosol.
- The raspberry-pi sensor detect different colors and shapes for a demo and recognize its environment

2.2 Deliverables

In this section we specify the functions and parameters that this project encompasses. We are going to give a quantitative value to the functions that are by this robot and specify three deliverables.

Specification	Target
Run Time	2-3 Hours
Target Detection Range	<40 cm
Target Detection Accuracy	90%
Fluid Capacity	1 Liter
Time to Clean	25 Seconds
Recharge Time	<1 Hour
Camera Field of View	60 Degrees
Solution Detection	<25%

Table 1: Deliverables

Our deliverables are as follows, we had the sprayer trigger every 15 seconds and fire off the cleaning solution from the sprayer bottle. Next to our second deliverable we had the ultrasonic and color sensor combined in unison so when we detected a specific color, such as the color green, and when we were within range of an object, less than 40 cm, it would make the system beep. Finally for our last deliverable we had it where when the solution was under 25% in the bottle it would trigger a five second long beep to notify when the bottle was nearing empty and then would shut down the system.

2.3 Block Diagram

This section consists of block diagrams along with the responsibilities of each team member. This shows how the work is divided up amongst the team.

Color
Allan McCormick
Rishi Patel
Shawn Manohar
Abel Assefa

Table 2: Block Diagram Color Code

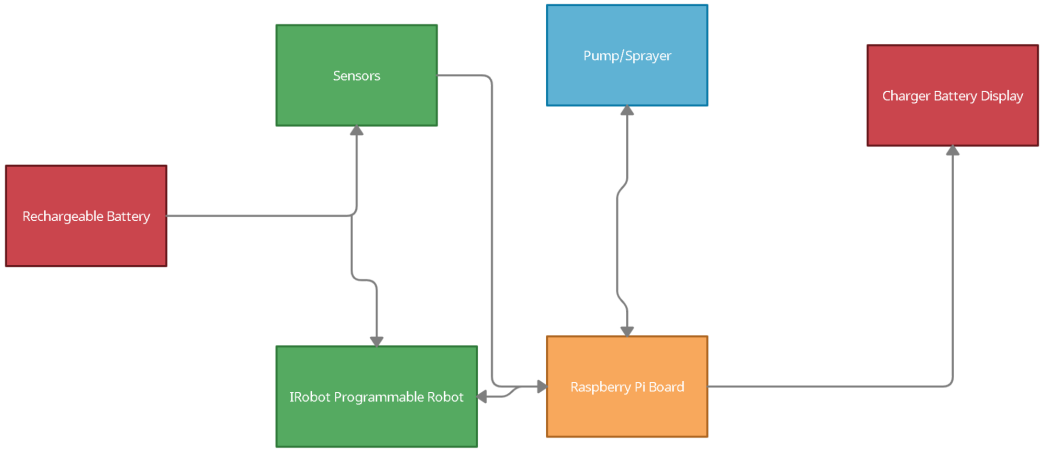


Figure 1: Block Diagram

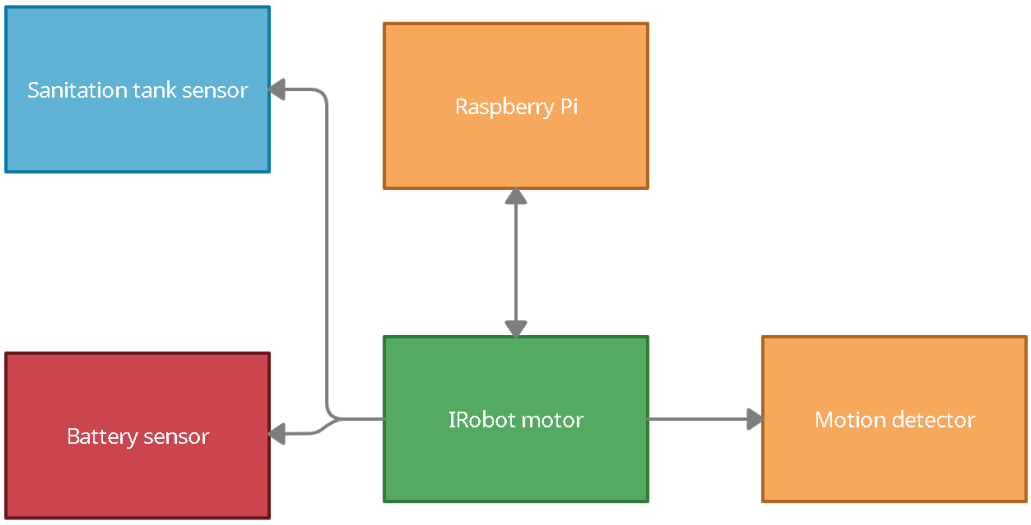


Figure 2: Software Block Diagram

Raspberry Pi Block

Status: Research

Input: The computer sends commands to different sensors that are needed.

Output: It connects to the IRobot motor and gives it commands.

Owner: Allan McCormick

Details: The code is in C.

Motion Detector Block

Status: Research

Input: The motion detector is connected to the motor and is powered by that and the Raspberry Pi.

Output: The motion detector detects objects and performs evasive maneuvers.
Owner: Allan McCormick

Battery Sensor

Status: Research
Input: The sensor is powered by the motor to detect low battery
Output: An LED light is emitted for low battery
Owner: Abel Assefa

Sanitation Tank Sensor

Status: Research
Input: The tank sensor is powered by the motor and detects when the sanitation is low
Output: An LED light is emitted for low liquid levels
Owner: Shawn Manohar

IRobot Motor/IRobot Programmable Robot

Status: Research
Input: Has a motor built into its frame that can be tweaked to detach and attach items on top
Output: The motor powers the devices and contains the wheels to maneuver around.
Details: The robot moves autonomously covering random areas without GPS tracking.
Owner: Rishi Patel

Pump/Sprayer

Status: Research
Input: Liquid is controlled by a certain amount sprayed and a timer.
Output: The spray emits the area whenever it isn't close to any objects.
Owner: Shawn Manohar

Charge Battery Display/Rechargeable Battery

Status: Research
Input: The motor powers and senses when the battery is in need of charging
Output: An LED is turned on to warn that the battery is low and it goes back to its charging station via Bluetooth.
Owner: Abel Assefa

Sensors

Status: Research
Input: Different sensors that are installed on the programmable robot.
Output: Turns on LEDs depending on the sensor tripped.
Owner: Rishi Patel

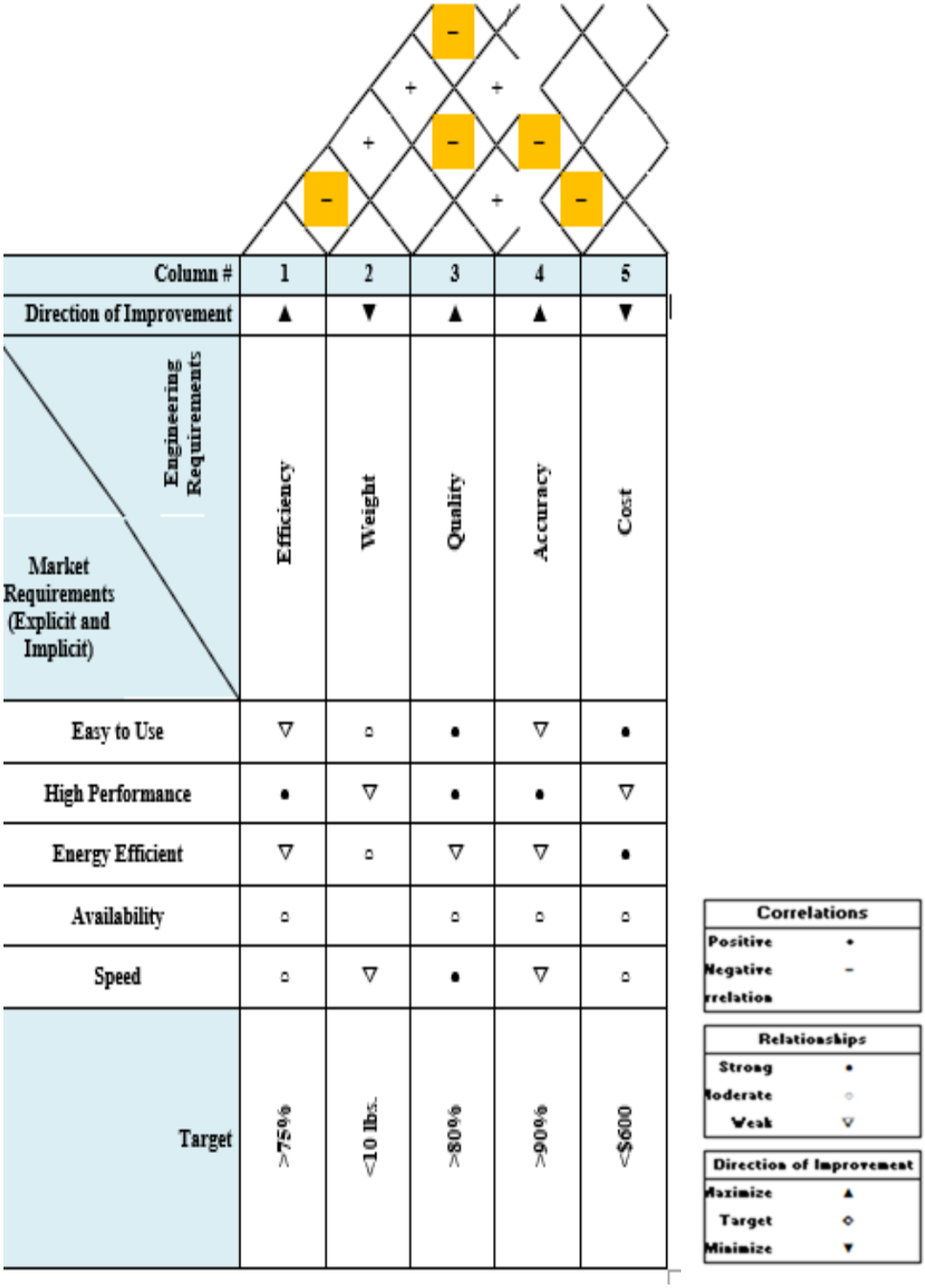


Figure 3: Marketing and Engineering Requirements/ House of Quality

3. Technology Comparison

This section consists of research into various relevant technologies for this project and if these technologies are useful. Similar technologies are compared to see if they are the right fit for this application.

3.1 Spray Mechanism

The initial option of the spray mechanism was a sort of aerosol spray. The would be rather simple to implement as the group would only have to add an actuator to initiate the spraying. Unfortunately, the aerosol does not provide enough force to clean a surface, the aerosol would only be able to coat the surface with the sanitation fluid. The aerosol also raised the problem of how we would refill it. The only way of refilling an aerosol can would be to replace it, as they are pressurized canisters. We believe this would cause unnecessary waste within the system. Although this would be the cheapest implementation for the system we believe it would be an unfit suit for this application.

The objective of this sprayer is to provide enough force to efficiently sanitize and clean the surface, as such there are various mechanisms for this operation. A simple household trigger spray bottle would suffice, but this would add unnecessary complexity as the group would have to devise a way of consistently pulling the trigger of this bottle, which would also entail more power consumption and additional weight to the system. The simplest method for this option would be to wire a servo motor to the main board that would essentially “pull” the trigger allowing it to spray, but this would be a slow process as the bottle would not be able to have a steady stream. The servo motor would have to oscillate at a slow enough rate to allow the trigger to reset and “pull” again. This would not be ideal as we believe too much time and power would be wasted on such an action. Although this would be the least expensive option with the addition of the servo motor and additional wiring this would raise the cost by more than double.

Another option would be a lawn and garden wand pressure sprayer. This would provide adequate spraying pressure to clean the surface and this sprayer is electrically operated so there is no need for complexity; the trigger and power can be wired to the main board and operated accordingly. Although this mechanism is ideal the reservoir that accompanies this device would be too unwieldy for this application. The cost of this option would be too great as this option would have to be heavily modified for this application. The common reservoir size for this device usually ranges from the 1-2 gallon mark. For this application, the group would require a much smaller reservoir such as about 1 liter.

Like the two above-mentioned spraying mechanisms, this one takes the electronically operated lawn and garden wand pressure sprayer and shrinks it down to the size of the household trigger spray bottle, see the figure below. This

allows the group to maintain the lightweight aspect while also lowering the complexity of the spraying mechanism. This spraying mechanism allows for variable pressure with the nozzle. With a turn of the nozzle the spray can go from a mist to a direct stream. With this the group can utilize this miniature spraying mechanism and even supply its own reservoir depending on its application.

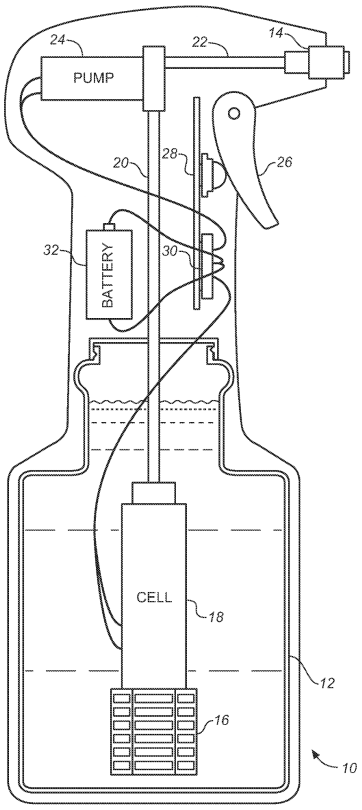


Figure 4: Example of Ideal Spray Mechanism

Due to its simplistic nature the device can be easily modified to suit the application with which it is being used. With this option the cost would be about the same as the cost with the household trigger spray bottle with the additional servo motor, but with the additional benefit of being a simpler mechanism to implement.With the device depicted in *Figure 4* group 9 be able to tap into the trigger, that is just a simple switch, and control it with the main board.

3.2 Sensors

A main component of autonomous robots is that of sensors. Many different sensors utilize different techniques and methods to fulfill their necessary purpose and achieve a desired result. In this project sensors were a key factor in achieving accurate and efficient results in our autonomous sanitation robot. Many different sensors range in pricing due to the complexity of that specific sensor. The goal for this autonomous robot is to select a sensor that accurately

recognizes any objects and or people in the way and act accordingly while being as cost efficient as possible.

3.2.1 Ultrasonic Sensor

A common sensor type is a distance sensor, and one of the most common distance sensors is an Ultrasonic Sensor. These sensors are commonly used due to their advantages. The ultrasonic sensor typically draws low power and current, can be utilized in many interfaces and is not affected by transparency or color due to the sound waves it emits. Some disadvantages are the limited detection range, low resolution and its simplicity does not allow for measurements of more extreme surfaces.

3.2.2 IR Sensor

These types of sensors are another type of distance detecting sensor that uses IR beams to calculate distances. The IR sensor is used in many applications, but it is most found in things like Security Systems, Laptops, TVs, etc. The biggest advantage that this sensor presents is it is a very small sensor, and its ability to recognize more complex surfaces and measure distance. Some disadvantages are that IR sensors have a very limited measurement range, and it is heavily affected by things in the environment and hard objects like walls and doors.

3.2.3 LIDAR Sensors

LIDAR is another type of distance sensor, and it stands for Light Detection and Ranging, which many have coined as the lasted distance sensor. LIDAR sensors are very key in application like forestry and land mapping. Along with that these sensors are key in machine control, robotic imaging, etc. The biggest advantage is its high measurement range and accuracy along with its ability to measure complex structures. These sensors also have fast update rates which work well for fast non stationary objects. They are also able to perform accurately at both day and night. A big disadvantage of this sensor is it has an extremely high cost, and the LIDAR pulses are very harmful to the naked eye.

3.2.4 Sensor comparisons

Now each of the sensors mentioned has their own unique way of recording distance measurements. Each sensor type has strong advantages and disadvantages. For the ultrasonic sensor, the biggest advantage of this sensor is the low power and current draw, but a drawback that comes with this is a lower distance accuracy and range. The IR sensor is very strong in its accuracy and distinguishes targets both stationary and moving targets, but it has relatively small and weak distance range. Lastly the LIDAR sensor is the best performing sensor available as it can do both what the Ultrasonic and IR sensors can do, but at an extremely high level. However, the drawback is that these sensors are

extremely expensive. In the next section specific components from a manufacturer of each type of sensor type be researched and expanded upon their benefits and drawbacks and at the end a final sensor be chosen

Sensor	Long Range	Power Consumption	Complex Surface Processing	Susceptible to Outside Influences	Cost
Ultrasonic	No	Low	No	Yes	Low
IR	No	Medium	Yes	No	Low
LIDAR	Yes	High	Yes	No	High

Table 3: Sensor Comparison

3.3 Battery Technology

Choosing the proper battery type is ideal for this robot. In this application a battery with a substantial runtime, but without a large physical footprint would be ideal as space within the robot is precious real estate. Most of all the battery type chosen needs to have enough capacity to run all the sensors and motors on this robot for a meaningful amount of time.

3.3.1 AA Batteries

AA batteries are very common and can be found at any convenience store making them readily available. Most AA batteries are Alkaline and thus non rechargeable, this makes them an extremely cheap form of battery, but also very wasteful. In order to use alkaline batteries in an application such as this would mean that we need about four to six batteries in an array to produce enough power to run the robot. The problem would be that the batteries have to be replaced after depletion every time.

Within the AA form factor there are rechargeable batteries. The most common amongst rechargeable AA batteries are known as Nickel-Metal Hydride Batteries or Ni-MH for short. These types of AA batteries are able to hold from 1200 mAh to 2500 mAh of power per a cell. Due to the improvement in their chemistry Ni-MH batteries are much more energy dense than Alkaline AAs with the added

benefit of being rechargeable. Although they do cost more than alkaline the end result is more savings from having to buy more alkaline batteries to replace the ones that went dead. The primary downside of these batteries in this application is that in order to recharge the batteries they must be removed from the robot and inserted into a charging bay, but due to this swappable nature it is possible to replace the depleted batteries with fully charged ones to keep the robot running while the other set of batteries is charging.

One of the downsides of using the AA form factor is that there is no easy way to wire them up to the main board unless we are using an enclosure. The only way to utilize batteries of this type would be to build or purchase an enclosure to house the batteries and run them in series that would have positive and negative terminals to connect to the main board.

3.3.2 Nickel Metal Hydride Batteries

Like mentioned before Ni-MH batteries do come in the AA form factor and can be used as a replacement for Alkaline AA batteries, but a much more popular form factor for Ni-MH batteries are in packs such as in *Figure 5*. In *Figure 5* it shows an array of Ni-MH batteries in a pack that is already wired. These packs come in a variety of sizes and voltages. With this they overcome most of their disadvantages of needing to utilize an external charging bay and removing the batteries from the robot.

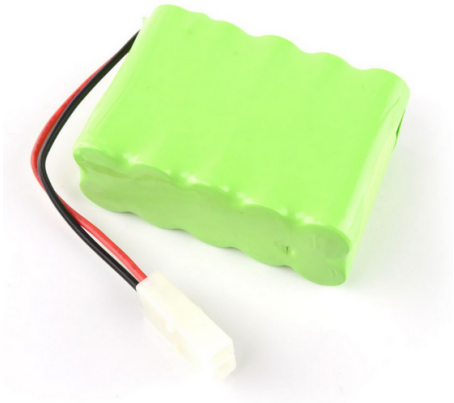


Figure 5: Nickel Metal Hydride Battery 1600 mAh

The cost associated with these batteries is relatively low compared to the cost of Lithium Ion and even Lithium Polymer Batteries. The weight would be about equal to the weight of alkaline batteries of the same volume, possibly lighter. These are often used in robots due to their cost effective nature and abundant availability. Problems with availability and sizing with the JST connection had our group opt out of using the Nickel Metal Hydride Battery and instead use 4 standard AA batteries in a battery pack.

3.3.3 Lithium Ion Batteries

Lithium Ion batteries also known as Li-ion are the batteries most commonly found in cell phones. They are very lightweight and have an extremely small footprint. The capacity and range in sizes from 100 mAh to 1000 mAh or even more, with common voltages at 3.6V to 7.2V. These types of batteries are very susceptible to heat, if given enough heat they are prone to explode. If the terminals of this type of battery are short circuited there is a possibility that the battery might explode. Due to this possibility a battery protection circuit is usually implemented into the main board to avoid any damage to the battery that might cause it to explode. The biggest advantage of these batteries is that they are extremely light weight compared to others. These types of batteries also have a high cost associated with them due to their high demand in electronics, but for this application the common voltages of 3.6 and 7.2 would not be ideal for this application as most of the electronics require a 3.3V or 5V power to operate.

3.3.4 Lithium Polymer Batteries

Lithium Polymer batteries commonly referred to as Li-Po batteries are the most popular type of batteries for drones, RC toys and robotics. They are very popular amongst hobbyists and would make a selection much easier due to the wide selection to choose from. These types of batteries, unlike Li-ion batteries, have a high protection layer to avoid combustion, these should still be handled with care as any bend or puncture can cause the battery to explode.

Like Li-ion batteries are very light weight and even thinner than Li-ion batteries with very high power density and similar nominal voltages to that of the Ni-MH batteries. The biggest thing with these batteries is that they come in many capacities and forms, but are the most expensive type of battery due their fast charge time and extended runtime. Due to this these batteries do tend to generate moderate amounts of heat and in some cases would require cooling.

There are instances of the outer foil of a Li-Po battery exploding due to the outer foil being punctured, but this also happened because the part was being mishandled. The internals of Li-Po batteries are very combustible when they come into contact with air. That is why every Li-Po battery is wrapped with a specialized aluminum foil to help prevent these punctures from accidentally happening and to also help dissipate some of the heat that is generated from these batteries.

3.3.5 Lead-Acid Batteries

Lead-Acid batteries are the known as the first type of rechargeable battery to be created due to this they have been refined over time. The most common type of Lead-Acid battery is the one found in automotive vehicles. These batteries are

much more durable than the Li-ion and Li-Po batteries mentioned before, and are able to withstand higher amounts of heat. There are two main disadvantages with this battery type for our application. The first being the weight, these batteries are the heaviest out of all these types mentioned before making it one of the lower energy dense types of battery. The second concern with this battery type is that the recharge time is much longer and would likely be a hindrance in the usage of the robot. The batteries range from 50% efficiency to 95% efficiency. These batteries tend to have lower charge cycles than more modern rechargeable batteries.

There is also a usage problem with these batteries as prolonged use can cause some battery acid to leak out from the terminals. This battery acid is corrosive and if it comes into contact with any other part of the robot it could cause extensive damage to whatever it touches. This kind of hazard is why lead-acid batteries are usually housed in a vehicle's engine bay or underneath the trunk away from passengers and sensitive equipment.

Battery Type	**Costs**	**Form Factor**	**Reactivity**
Lead Acid	Low	Very heavy similar to form to Car Batteries	Moderate, can produce battery acid
Ni-MH	Medium	Similar to AA can can be wired in series	Low, can only react if punctured
Li-Po	High	Very thin and light similar to cell phone batteries	High, if nent or punctured combustion.

Table 4: Battery Comparison

4. Parts Selection

This section looks to research, analyze and ultimately decide which components be used for our autonomous sanitation robot. Some factors that are looked at are power consumption, stock availability, pricing and many other factors.

4.1 Chassis Selection

There are many chassis to choose from and a multitude of vendors that produce robotic chassis. This section consists of a couple of options we have found that would be a feasible option for our application. We then compared these chassis to see which one was best suited for this project. We then decided to use the iRobot system as a whole to be the full chassis.

4.1.1 Robot Create 2 Programmable Robot

The Create 2 Programmable Robot was used as the chassis of our project. It autonomously moves around with our mount on top that holds our sprayer system. The programmable robot weighs 7.9 pounds, 3.62 inches in height, and is 13.39 inches in diameter. The package comes with the robot, rechargeable battery, home base, and communication cable. The communication cable can speak to a computer or microcontroller. The chassis has safe drilling areas to install parts on top or install microcontrollers like the Raspberry Pi. The robot is self-charging and can return to its home base and immediately start charging that can last for 3 hours.The first two pins relate to its battery charging that can enter a low power saving mode that sleeps the robot after 5 minutes of inactivity. Different modes can control the battery in different ways.It can move up to 500 mm/s. The robot uses an omni-directional IR receiver to detect objects and evade them. It has a IR cliff sensor, IR light-touch and wall-following sensors, IR receiver for its home base dock, and a dirt detection sensor. It has two wheels and three motors with built-in LEDs.

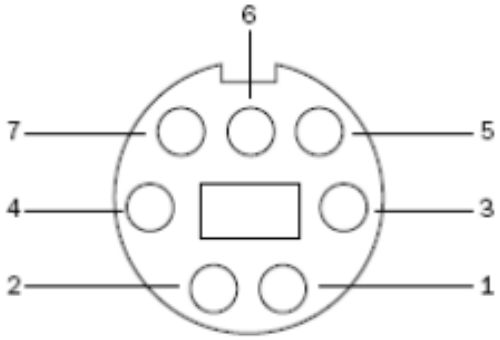


Figure 6: Roomba External Serial Port Mini-DIN Connector Pinout

We plan on adding additional sensors and LEDs mounted on the chassis like the Ultrasonic sensor to detect objects that determine how the spray system functions. We coded the microcontrollers associated with our robot with C.

IRobot Create 2 Open Interface Based on Roomba 600

The overall idea is to use this robot as the base and to have the autonomous spray mechanism and all other sensors on top. The reason for choosing this robot is because it allows for us to have a working chassis and ensures we don't run into any problems with trying to get the robot to move. The main problem and focus of our project is to ensure that the robot sprays when it's supposed to and doesn't spray in specified areas all while monitoring the level of spray solution until a specified value and alerting users of this low level.

The IRobot is also able to communicate with the microcontroller and this allows the microcontroller the ability to utilize the features that come with the Irobot. An example is the return to base command available to the Irobot. When the microcontroller receives a signal that the solution is low it communicates this to the Irobot to send a return to base command. This is a prime example of the way this robot can be interfaced and used in conjunction with our overall project idea. This is also one reason why this irobot was chosen instead of some self built chassis. The Irobot ends up being one of the more expensive things in our entire project, but the performance it brings and real makes up for this high price mark.

4.1.2 mBot Ranger 3-in-1 Coding Robot kit

This is another robotic chassis from makeblock. This chassis has a robot tank, a self-balancing, and racing car all in one set. The robot can be controlled and coded using the MAKEBLOCK app, and the coding can be done in Scratch and Arduino C to program the movement. Now onto the design and CPU of this chassis. This robot uses the same MCU chip which is found in another product line; the Arduino Mega 2560. The parts of the robot are made with a high-quality aviation aluminum alloy. Finally, this robot is made to make coding easy to use with the many coding modules and available sensors. This robot contains an ultrasonic sensor, line-tracking sensor, 2 light sensors, gyroscope sensor, 12 RGB LEDs, Buzzer, Temperature sensor, and a Bluetooth module.

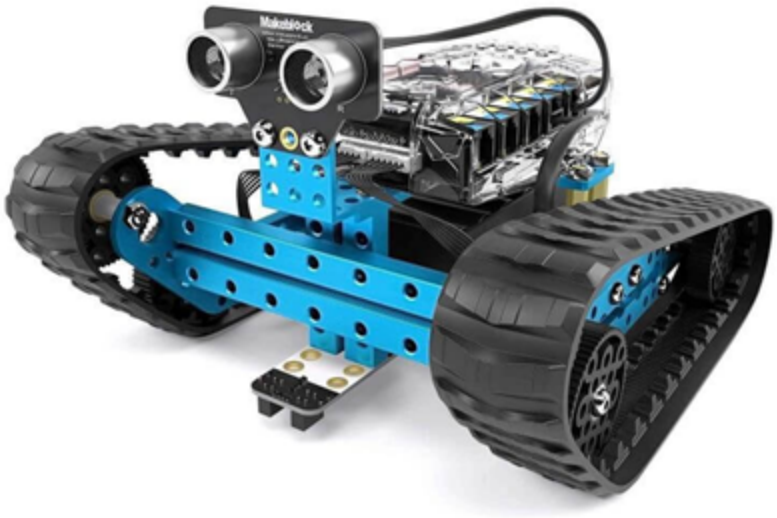


Figure 8: mBot Ranger

This robotic chassis has great advantages for our autonomous robot project. One big key advantage is the access to all the sensors. This mBot ranger has several different sensors available to be coded and utilized by using Scratch and Arduino C and help program movement in the robot. The number of sensors help in our project in accuracy and distance sensing to ensure that our project works as accurately as possible. One other advantage is the price point to this robotic chassis. This robot is listed at $129.99 which compared to the earlier Irobot, which is listed as $200, makes this robot extremely competitive due to this pricing gap. However, a big disadvantage this sensor contains is the amount of space available on top. The figure above shows the mBot and when studied it can be seen that there is not much space at the top of this robot. This is a key disadvantage because our project is based on placing things on top of our chosen chassis, specifically our spray mechanisms as well as all our other components. To correct this issue, we may need to build a wider platform on top to create the necessary space and this may take up too much time and money.

4.1.3 Zumo Chassis Kit

This is a small, tracked robot that has about 10 cm on both sides. It is made of an all-black ABS and features a compartment with four up to four double A batteries and sockets available for two micro metal gearmotors. The kit comes shipped

with two silicone tracks, two drive and two idler sprockets, a 1/16 acrylic mounting plate, and mounting hardware. The kit does not come with any motor and batteries.

Figure 9: Zumo Chassis

The figure above shows the Zumo chassis completely assembled, but this chassis has many working systems and drives that make it a unique and viable chassis. The battery compartment terminal pokes out through the chassis making it available and accessible for the top side of the chassis. The black plate is included with the kit and is to be used to hold the motors in place. Along with holding the motors the plate can be used to house electronics like a microcontroller and sensors. The drive system in this chassis consists of two black silicone tracks on each side. They are supported by a free spinning idler sprocket and a motor-driven drive sprocket.

The Zumo chassis is a very solid chassis that does not have many complexities to it but is simple in nature. The biggest advantage is the freedom this chassis allows for design and implementation. Unlike many development kits this robot is stripped down and has absolutely no circuitry in it leaving it to be completely designed and redesigned as the user sees fit. The design with the battery compartment and the black plate that can house many electronics is a bug because this type of chassis is something extremely viable and useful for our project. The other big advantage is that this chassis is extremely well priced. Since this is a very bare chassis this price point is to be expected, however it is still something very positive to overall budget constraints and limitations to this project. The low price point with the addition of the necessary sensors would still be less than any other development kit chassis that are available.

A big disadvantage of this chassis however is that implementation of the sensors is extremely difficult and lengthy. With this chassis not having any available sensors it would be part of our job in the project to ensure that all sensors are provided and accurately work. This would essentially turn our project into two projects. We would need to tackle the issue of getting the chassis to work autonomously and sense all appropriate targets while also recognizing when to spray and when not to spray. It can also work out that we may be able to implement that the sensors be able to detect movement and avoid obstacles while also tracking spraying, but this could lead to heavy coding and software design for our group. Trying to tackle two problems simultaneously is a risk that could lead to having a project that does not work in any capacity.

4.1.4 Final Chassis Selection

Through this section three different chassis were looked at and studied. Each one of these chassis have positive impacts on our project and drawbacks as well. However, with all factors considered we are selecting the IRobot Create 2 as our chassis.

There are some key features that the IRobot provides that we thought would be essential for our sanitation project. One is that the IRobot can communicate with microcontrollers meaning we are able to communicate with the IRobot and specifically the sensors inside to coordinate with our spray mechanism and spray when specified. The IRobot is the most expensive option out of all the listed chassis, but this is one of the only drawbacks to this option. There is also the case that with the Mbot and Zumo we may need to acquire more sensors than we would need to get with the IRobot which would essentially put the pricing closer to what the IRobot goes for. The IRobot also has mounting holes and a safe cutting area on top which allow us to place our mechanisms. In all the IRobot gives us an extreme margin of control over all the available sensors and it gives us the best results for our project.

Chassis	Sensors	Price in $	Language
IRobot 2	10+	200	Any
mBot Ranger	6	130	Scratch/Arduino C
Zumo	None	19.95	Any

Table 5: Chassis Comparison

4.2 Micro-Controller

With a microcontroller the level of input and output with the iRobot would be the simplest for both of the Microcontrollers to interact with each other. This level of synchronization could only be made possible by the Raspberry Pi which keeps the internals and the external brains talking to one another so they can solve problems real time. This fully fledged brain all be housed in the iRobot chassis and serve as the autonomous product that can interact with its environment around it thus creating a positive feedback loop.

When it comes to the brains of the iRobot, where the spraying, refilling detection system and environmental detection all be controlled by a powerful microcontroller. We are using a microcontroller that is embedded within the system so it can interact with all the different deliverables that we have setup throughout the iRobot. The microcontroller interprets data that it receives from its input/output peripherals using a central processor that's housed within the iRobot. This information is all stored as data in the system's memory. The information is then deciphered and is then applied.

Finally, the microcontroller using the input/output peripherals to communicate with the deliverables and execute the action it was given. This simple series of events is how we plan on keeping the interaction of the movement of the system and the execution of the deliverables to coincide with one another.

4.2.1 Microcontroller Processor (CPU)

There are about three main elements that we are interested in when looking at a microcontroller. One of the elements is the processor or the CPU, this element would act as the brain of the microcontroller. This part of the microcontroller would process all the various amounts of instructions that direct the microcontrollers' function. This would include the basic logic and input/output operations that we would need for the iRobot's deliverables. We also need a strong enough CPU to communicate with the other devices that compute logic on our system such as the raspberry pi and the Arduino that's located within the iRobot's system. This leads to the most important functionality of the microcontroller, which is the ability to perform data transfer operations. This makes it so we can communicate commands or logic to other components within or on the iRobot's embedded system.

Users need a strong and powerful CPU to act as the brains of the system. The CPU would need to have enough processing power to deliver instructions to the Arduino, that's located inside of the microcontroller, the raspberry pi and all the sensors on top of the iRobot's systems chassis. The microcontroller's CPU would tell the sensors to look for any obstacles in the way and if there was one then the sensors would tell the CPU and the CPU would then need to process instructions

that can be delivered to the Arduino so that the iRobot's system can move accordingly. Then the CPU would need to tell the Arduino how to get to the target and how to get there, all while taking in instructions from the ultrasonic sensors. Finally, when the chassis arrives at the respected target then the CPU would need to tell the raspberry-pi to fire off the trigger mechanism at the target they are looking at. This all needs to happen as efficiently and as quickly as possible because if the CPU takes too long to process the information or can deliver instructions back to one of the parts then the iRobot system could crash into something or miss fire and spray the wrong targeted location. This is why the CPU on the microcontroller is an essential element and needs to be considered when choosing what type of microcontroller we as a group would want to use.

4.2.2 Microcontroller Memory

Another element that we are considering when choosing what microcontroller we would want would be the amount of memory that the microcontroller can hold and process. A microcontroller's memory would be used to store data that the CPU, or processor, receivers and to respond to instructions that it has been tasked to carry out. There are also two types of memory within a microcontroller, program memory and data memory.
Program memory is used to store long-term information about the instructions that the processor carries out. These could be instructions that are constantly running and always need to be executed. For example, we would need a large amount of program memory for the iRobot's system because it would need to constantly look around its environment and be ready to get out of the way if it detects an obstacle. This type of memory is non-volatile, this means that it can constantly hold this information overtime without needing a power source.

The second type of memory that is housed in a microcontroller is the data memory, which is used for all the temporary data storage. This temporary storage would be held then forgotten as the instructions are given then executed. For example, we would need data memory for the interaction of when we detect when we need to spray a certain region, because we don't have to spray all the time; it would be temporary data that is being given out. This type of memory is volatile, meaning that when the data that it is given is temporary it is only maintained when it is connected to a power source.

4.2.3 Microcontroller Input/Output

The final element that is incorporated when choosing a microcontroller would be the I/O peripherals. The input and the output acts as the interface of the processor that allows for all the deliverables to be acted upon. The input port on the microcontroller receives information and then sends it to the processor via binary. The processor then looks through all the data and sends the instructions to the output devices that execute the deliverables on the iRobot system.

These are the basic types of elements that we would need to look out for when shopping for a microcontroller that fits in well with our project but there are some specific ones that we also need to look into. Analog to Digital converters and Digital to Analog converters allows us to communicate to the external analog and digital devices such as the sensors. The processor in the microcontroller can interface with the sensors so they can give directions to the iRobot's system and control its movement. The ADC and the DAC would be able to communicate the systems outgoing and incoming movement through the different external sensors. The serial port is another element that we need to look out for when shopping for a microcontroller. The serial port would allow the user to connect to external components such as the sensors or other microcontrollers. It is very similar to a USB but is different in the way it exchanges bits of information to one another.

4.2.4 8051 Microcontroller

The 8051 Microcontroller is one of the newer and more ideal choices in the microcontroller family. This microcontroller is an eight-bit microcontroller that was launched by Intel. It has a 40-pin dual inline package, 40KB of ROM and 128 bytes of RAM that is built within the microcontroller. It also has the capability of 64KB of external memory that can be interfaced with the microcontroller. There are two different sorts of memory that are used in the 8051 Microcontroller such as FLASH, NV-RAM and UV-EPROM.

There are four parallel 8-bit ports on the 8051 Microcontroller which are programmable and easily addressable when trying to connect them to the external sensors or other logic devices. The on-chip oscillator is integrated in the microcontroller which has a 12MHz crystal frequency. Also, within the microcontroller there is a serial input/output port that has 2 pins that we can access to integrate and execute the different commands from the processor. There are also two 16 bit clock timers that are also incorporated in the housing of the microcontroller. Both of these timers can be utilized internally as a timer or externally as a counter functionality.

The 8051 Microcontroller comprises 5 interrupt sources which include: Serial Port Interrupt, Timer Interrupt 1, External Interrupt 0, Timer Interrupt 0 and External Interrupt 1. The programming mode of the microcontroller includes all the general-purpose registers, special function registers and the special purpose registers. The 8051 Microcontroller would use these interrupt sources to adjust the internal clock of the processor. This way the microcontroller can take in permenent and temporary data and deliver instructions in the correct order. This way when we time an interrupt we can send the correct data stream to the correct sensors that then execute the task.

The 8051 Microcontroller gives the user a lot to work with in that you can apply a whole load of instructions and external devices that can be connected and

interacted with one another. The output pins give a whole range of motion and capability that would allow for any system to perform any necessary tasks or logic to be carried out. For the iRobot system I could see this microcontroller as a prime contender as the microcontroller that is utilized because it would be able to carry out all the necessary functions that need to interact with one another. It would also be able to communicate to the sensors at the top of the system and give instructions to the sprayer and raspberry-pi system to execute the given task. It is also very battery-powered friendly in that it consumes less power than the other processors. Finally, price-wise, the board is fairly priced and is very similar to the other microcontroller boards out there. Some of the constraints of this board would be that it is prone to overheating and it does have a drawback of not being the fastest board on the market for the price that it is listed at. But other than those two drawbacks it would be a prime contender of being one of the boards that we pick for our iRobot system.

4.2.5 PIC Microcontroller

The PIC microcontroller or the Peripheral Interface Controller provided by Microchip Technology is a very successful and powerful eight-bit microcontroller. PIC microcontrollers are very popular because of their wide availability, low cost, large user base and serial programming capability.

In the base-line architecture PIC microcontrollers make use of the 12-bit program word architecture with a 6 to 28 pin package. The PIC10F200 series, allows for an 8-bit FLASH microcontroller with a 6-pin package. There can also be a variety of package alternatives from the range of 8 to 64 pins, all with low to high levels of peripheral incorporation. These attributes would contribute to the variety of analog, digital and serial peripheral that would be on the external level of the system. Serial peripherals like SPI, USART, I2C, USB, LCD, and A/C converters would all be included in the embedded system of the microcontroller.

The PIC18 family would allow for 18 to 100 pins and makes use of the 16-bit program word architecture. The PIC18 appliances would act as high-performance microcontrollers with in house analog to digital converters. All PIC18 microcontrollers integrate a highly developed RISC architecture that supports all FLASH devices. The PIC microcontroller also has improved foundational attributes with a 32-level deep load and several inner and external interrupt capabilities.

All in all, the PIC family gives a wide range of freedom when it comes with the number of connections you can make. It would allow for a variety of integrations with the incorporated analog and digital converters. The PIC18 would be a suitable choice for the iRobot system because of its wide pin lay out and its high-performance A/C converters. If we were to choose this it would be because we would easily be able to program the microcontroller to the external and internal sensor in the iRobot’s system. This would be a critical application of the

board that users would select because we would need a fast and reliable communication stream between the movement of the iRobot and the sprayer at the top of the system.

Some drawbacks would be how much the PIC18 costs in comparison to the other microcontrollers that can do more for less cost. Also, another drawback would be the amount of programmable ROM and RAM that this microcontroller has in comparison to other competitors. But drawbacks aside, this is a great microcontroller for what we are trying to accomplish and would come in handy when trying to send instructions to all the different sensors in the system. We would need this as the top of the chassis of the system would be filled with ultrasonic sensors that would have to get instructions from the microcontroller as quickly as possible so it can make a decision. But if the microcontroller can store enough of the temporary data or the non temporary data then it would drastically slow down the process for which our iRobot's system would run. I would still say that this is a great option as a microcontroller for our iRobot's system, but we would just need to be looking into expanding the microcontroller's external ROM and RAM capabilities.

4.2.6 MSP430 Family

The MSP430 family, provided by Texas Instruments, is one of the most popular designs for a microcontroller that allows for a wide pin layout and several A/C interactions. It has a low cost, high availability, and large serial programmability.
The MSP430F5529 has a 12-bit SAR ADC and 128KB of nonvolatile memory that's incorporated in the housing of the microcontroller. It has 81 GPIO pins that allows for a deep load and has several internal and external interrupt capabilities. Within the MSP430F5529 there is a crystal clock frequency of 16 MHz that can be used for an internal timer or an external counter in the microcontroller. Also, the microcontroller has 16KB of RAM associated within it.

Some of the serial peripherals that are included in the MSP430 are as follows: ADC, UART, I2C and SPI. There are 16 ADC channels, 4 UART channels and 2 I2C channels and 4 SPI channels. These could all be used to program the microcontroller to carry out the various tasks and deliver the instructions to and from each of the deliverables. The price of one of the MSP430 models would be about 4.73 USD.

The MSP430F5529 serves as the half of the iRobot's brain where the spraying and re-filling detection system be held, this Microcontroller then co-inside with the integrated Arduino that rests within the chassis of the iRobot. As the Arduino acts as the eyes and extremities of the iRobot, the MSP430F5529 carries out the functionality of the iRobot. Both of these micro-controllers would be integrated on the iRobot and would use a Raspberry Pi to bridge the gap between the two functionalities. This way we can have the steering, recharge and detect where to

spray and talk to the other half of the iRobot's brain, where we spray for and when we need to refill our tank.

We chose to use the MSP430F5529 because it had the best relationship with both the Arduino and the Raspberry Pi. This way we could smoothly have it where we can send information to the Arduino using a communication cable and it would be able to send directions to the Raspberry Pi and the Raspberry Pi could deliver those directions correctly to the MSP430F5529. This system would allow both Microcontrollers to access multiple peripherals on the chassis, including the sensors on the sprayer, LEDs, light sensors and the motors. The MSP430 then is able to execute those directions smoothly when to spray the canister, flash the LEDS and run the motors that we have attached to the chassis. When the MSP430 detects when the canister is getting below a certain threshold then it delivers a message to the Raspberry Pi to tell the Arduino to head back so it can get refilled and recharged. The use of the development kit so we can fully hack the iRobot so we can make it communicate with the MSP430.

This is a cross-platform tool that allows for the software to be coded in C and lets the Raspberry Pi communicate with one another. Some other benefits of the MSP430F5529 would be that it has advanced sensing, DMA, LCD, is low-energy and has a real time clock. These features would be more than perfect to run our iRobot system. Some downsides to using this model would be that it is a little pricey, running at about 4.70 USD, when compared to the other models that were shared above. We chose to use this model in the end as we had the most experience coding this from previous classes. We as a group also decided on this model because it was the most abundant throughout the group and we didn't have to buy another one. This was the microcontroller that we could get the most done on without having to learn another coding language and having to buy another model of microcontroller. In the end we were able to get all of our deliverables met and meet all of our requirements for our SD2 project.

Specs and Return Time	8051 Microcontroller	PIC Microcontroller Family	MSP430FR 6989 Family
Number of Bits	Eight Bit Controller	12-bit program word architecture	12-bit SAR ADC
Memory	40KB of ROM and 128 bytes of RAM	50KB of ROM and 128 bytes of RAM	128KB of nonvolatile memory
Pin Layout/Type	40 Pin Dual Line Package	28 Pin Package	81 GPIO pins
Clock Frequency/Type	16-bit Clock Timer	16-bit Crystal Clock Timer	Crystal clock frequency of 16 MHz
Number of Channels	5 interrupt sources which include: Serial Port Interrupt, Timer Interrupt 1, External Interrupt 0, Timer Interrupt 0 and External Interrupt 1	Serial peripherals such as: SPI, USART, I2C, USB, LCD, and A/C converters	16 ADC channels, 4 UART channels and 2 I2C channels and 4 SPI channels
Cost Per Unit	$4.95 USD	$1.85 USD	$4.73 USD

Table 6: Microcontroller Specification Comparison

Above lists a table that formats and compares the following potential microcontrollers for groups. Here is where we determine which controller would best suit our needs for the iRobot system.

After some deliberation we have determined that the following MSP430FR6989, pictured below in figure 10, is a microcontroller that we are the most familiar with and is a microcontroller that can process all the functionality that the iRobot system needs to function.

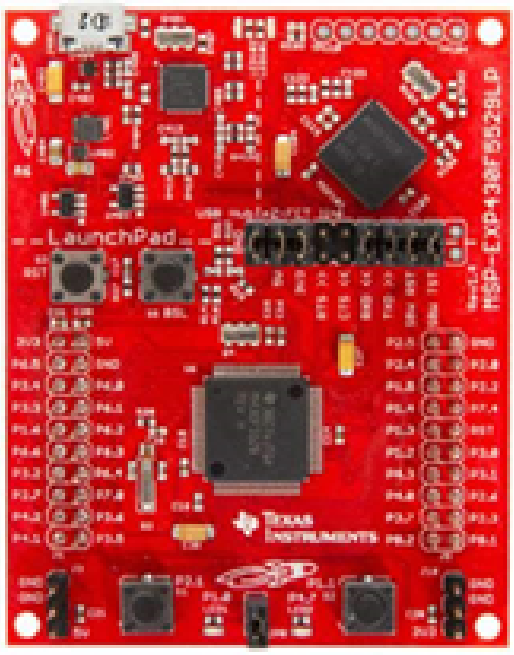


Figure 10: MSP430FR6968

4.3 Raspberry Pi

The Raspberry Pi served as the bridge between the two brains of the iRobot and acted as the spinal-cord of the whole system. The Raspberry Pi is computing the directions that communicate to each of the Microcontrollers. The Raspberry Pi would be coded using scratch, this allows the eyes and extremities to communicate with the external functions of the sprayer. We are able to use the communication cable to efficiently send over the information that the iRobot needs to both move around and spray the canister. All in all, the Raspberry Pi would be used to have the two different Microcontrollers interface with one another so they can communicate and solve problems real time. Without the Raspberry Pi and the communication cable the spinal-cord of the whole system would be severed and the Microcontrollers wouldn't be able to work together.

The Raspberry Pi would let the two Microcontrollers fire off the motors and at the same time detect when to spray the canister. Multiple deliverables such as having the LED's blink when the canister is going to spray, having a sensor detect when the canister is low on spray and making the iRobot go back to its docking station when it has to recharge. All of these deliverables are made possible by the Arduino in the iRobot and the MSP430 communicating with one another by using the Raspberry Pi to deliver instructions via scratch code. The software can be delivered by using the communication cable to set up the control peripherals, communication peripherals and the external peripherals.

Figure 11 shows the block diagram for the Raspberry Pi that shows all the different GPIO Ports, sensor Ports, Ethernet Ports, and USB Ports for the Raspberry Pi-3. This figure shows the typical layout of most Raspberry Pi models. This specific example is the Model 3 version, but the next few sections focus on different models of the Raspberry PI family and deciding which work best in meeting all the needs described for the autonomous robot. We want to find the best perfromaning model that also meets well in our budget constraints.

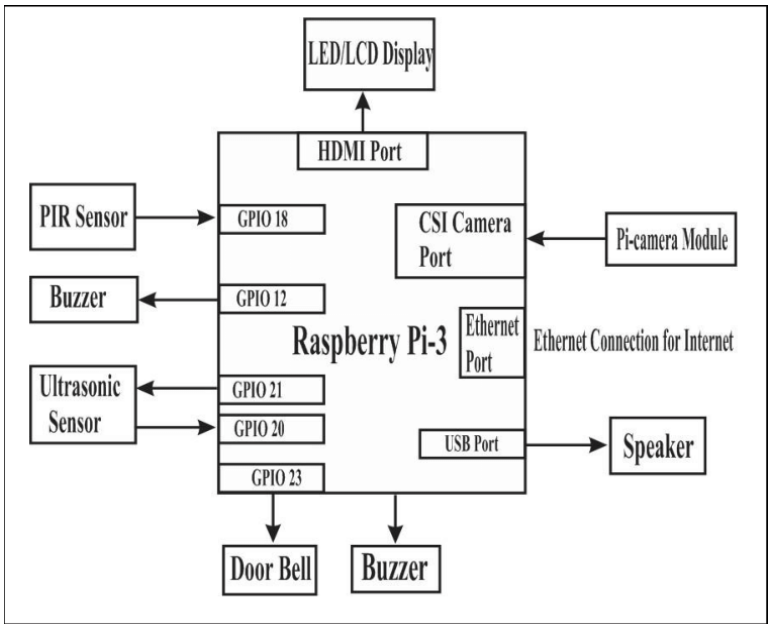


Figure 11: Raspberry Pi-3 Functional Block Diagram

In the block diagram you can see how the GPI0 18 Port corresponds with the PIR Sensor, the GPI0 12 corresponds with the Buzzer and the GPI0 21 and 20 correspond to the ultrasonic sensors input and output functions. The HDMI Port corresponds to the LED/LCD display and the CSI Camera Port links with the Pi-camera model.

This block diagram can be used to showcase how the iRobot's systems connect and communicate to each of the different parts on the chassis. We need the PGI0 ports to connect the ultrasonic sensors on the top of the chassis and we need the USB port to connect to the microcontroller and another USB port to connect to the inside of the iRobot's system.

This was all done while looking at a block diagram so we can pick out the correct model of Raspberry Pi, this way we as a group can ensure that all of the different brains and logic of the system would pair well with one another and would be able to talk to one another quickly and efficiently. We got the Raspberry-Pi to work and have it talk to the MSP430 Development board, this made it so we

could accurately fire off all our deliverables while talking to both the sprayer and the MSP430.

4.3.1 Raspberry Pi 3 Model B+

This is one of the latest products of the Raspberry Pi 3 models. This model runs a 64-bit quad core processor that runs at 1.4 GHz, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT. The dual-band wireless LAN comes with modular compliance certification which allow this board to be redesigned into other products with reduced wireless LAN compliance testing which is a great improvement in both time and cost Lastly this Raspberry Pi model has the same mechanical footprint as the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

The Raspberry Pi 3 Model B+ has an input power of 5 Volts DC and it draws a 2.5 A via some USB connector. The input power can also be achieved through a 5 V DC GPIO header or using a POE which is Power over Ethernet. The board has 4 USB 2.0 ports and has an extended 40-pin GPIO header. Lastly the board has video and sound capability with a 1 f\full size HDMI, a MIPI DSI display port, A MIPI CSI camera port, and a 4-pole stereo output composite video port.

This board gave us an extreme amount of freedom to use this board in many ways. There are 40 General Purpose Input and Output pins, which can be used in multiple ways. Users can combine the Raspberry Pi 3 Model B+ and combine it with other products like the raspberry Pi colour sensor, to perform necessary tasks. For our project we may use this combination in a demo to help show that our autonomous robot can recognize a given color and based on what we put it respond accordingly whether that may translate to spraying or not spraying.

Lastly the Raspberry Pi language is C, which is a well versed, powerful, and relatively easy to use language. Raspberry Pi also has multiple resources and guides to help many issues and coding programs that may arise. Lastly the pricing of this board is relatively well compared to its competitors. Some drawbacks to this are that the Ethernet port has been stated to not reach the fill speeds stated that it can run and that under high levels of computing loads there have been notes overheating.

We got the Raspberry-Pi to work and have it talk to the MSP430 Development board, this made it so we could accurately fire off all our deliverables while talking to both the sprayer and the MSP430.

4.3.2 Raspberry Pi 4 Model B

This model is the newest in the product line of the Raspberry Pi computers. This product has many new features and upgrades when compared to its older generation the Raspberry Pi 3 Model B+. This model has increases in processor speed, media performance, overall memory, and connectivity all while keeping being able to be backwards compatible with all designs having older Raspberry Pi models. This model also has a very similar power consumption to its previous predecessor the Raspberry PI 3 Model 3+.

Some of the specs of this board is a 64-bit quad-core processor with dual display support at as high as resolutions of 4k by a pair of micro-HDMI ports. There is also video decode hardware up to 4kp60, up to 8Gb of RAM, dual-band 2.4/5.0 Ghz wireless LAN, Bluetooth 5.0, Gigabit Ethernet and much more.

This board is very similar to its previous model, but with many upgrades and advances to the overall performance of the board. So naturally a big advantage is that the performance of our robot using this board is significantly better than using the Model B+. There are 40 of the general input and output pins on this board which is the exact same as the previous model. This model also faces a similar problem as the previous model in terms of overheating during heavy computation loads, but in all this board is clearly a better board in terms of performance. However, the biggest drawback is that this board has a higher average price point than the Model B+, which is expected as this is a newer board with better features. This comes down to looking at the performance of both and seeing if the Model B+ of the Raspberry Pi 3 is enough to take care of the needs for this autonomous robot or if it is necessary to spend the extra money and take the higher performing board.

4.3.3 TCS3200 Color Sensor

This is a color detector sensor that comes equipped with the TAOS TCS3200 RGB sensor chip and 4 white LEDs. The RGB chip has a vast range of detecting visible colors and has many applications from sorting by color, color matching, ambient light sensing and calibrations. This is done through the many photodetectors on the sensor itself that allows it to take in all the different visible colors and properly sort and identify them. This sensor also contains an internal oscillator that produces a square wave which has a frequency that is proportional to the intensity of a color that is chosen. This is how it reads a specific color and blocks out all other colors.

This sensor operates at a DC voltage of 2.7 Volts to 5.5 Volts, with high performances with increased voltage until the maximum value. This sensor works heavily with frequency throughout its components. There is a High Resolution Conversion that takes the light intensity and converts it to frequency. There is also a programmable color and full scale output frequency that is supported by this sensor. This also supports communication directly to a microcontroller or raspberry pi without the need to use another support hardware or features. Lastly

the size of this has the dimensions of this sensor is relatively small and compact with dimensions standing at 28.4x28.4 mm

There are also 4 switches; S0,S1,S2,S3, that deal with the output frequency and photodiode. Switches S0 and S1 directly control the scaling selection inputs of the output frequency. These two switches can control and range the output frequency that directly affects that of the square wave output, and the range of the typical output frequency runs from 2 HZ to about 500 KHZ. The two states that the switches are set are at a value of L corresponding to a low logic and a value of H corresponding to a high logic. *Table 6* shows the associated logic level of each switch and how it corresponds to the effects on the output frequency. S1 has a stronger impact as the power scaling percentage jumps almost 10 times when S0 switch is high and the S1 is low when compared to the reverse case.

S0	S1	OUTPUT FREQUENCY SCALING (fo)
L	L	Power down
L	H	2%
H	L	20%
H	H	100%

Table 7: Output Frequency Scaling Chart of TCS3002D

S2 and S3 are the other sets of switches that deal with the photodiodes and what colors are being read and subsequently what colors are being omitted. The switches S0 and S1 give us the different scaling factors by different combinations while different photodiodes are chosen through different combinations. Figure 10 shows the different combinations that get the three colors Red,Green and Blue to be read. With the switches in the corresponding place that respective photodiode type allow that specific color to be filtered through and the scaling factor of that output is determined by S2 and S3.

S2	S3	PHOTODIODE TYPE
L	L	RED
L	H	BLUE
H	L	Clear (no filter)
H	H	GREEN

Table 8: Photodiode Type of TCS3002D

The key application and use of the sensor in our autonomous robot helped demo our robot's ability to recognize surfaces that need to be sprayed and surfaces and or areas that don't need to be sprayed. This is one of the key objectives of our robot and with this color sensor we demonstrate that the robot is able to recognize a given color and based on user input the robot spray or not spray in accordance with the users specification. This is to show how our project is only a surface level design and one that can be added on and improved with more technology and design work.

With a project like this instead of using the color sensor the idea would be to show that the principle of when to spray and not spray is feasible and to turn this into something that revolves around advanced image processing where the robot be able to use advanced tools like infrared or other imaging tools to determine if an area should be sprayed based on multiple factors or not to be sprayed either due to humans are around or possible the hazards and dangers spraying in that environment might have.

We were able to use this color sensor to more accurately detect and recognize spaces that need to be sprayed. This was very useful in our final demonstration for our deliverables. Using this specific color sensor we were able to read and send a beep through the MSP430.

4.3.4 Final Raspberry Model Selection

Now the TCS3200 Color Sensor is the sensor already selected and paired with the two Raspberry Pi models discussed: the Raspberry Pi 3 Model B+ and the Raspberry Pi 4 Model B. As shown through the research into these two different models they are very similar with the Raspberry Pi 4 has better processor and all around performance while the Raspberry Pi 3 has less performance but is a better price mark. With everything stated we are using the Raspberry Pi 3 for three key reasons. The first is that the pricing is extremely friendly especially with the other components we are planning to possibly use in this project. Secondly the performance this board gives may be less compared to the model 4, but it is more than adequate for the needs we want. Lastly the TCS3200 is extremely

compatible with the Model 3. The color sensor is compatible to most raspberry pi and mictonrotleresl but the raspberry Pi website themself have an extremely useful help guide in helping integrate the color sensor with the model 3 board. These are all the listed reasons why we believe the Raspberry Pi 3 Model B+ to be the best available model for our project.

4.4 Sensor Selection

For this project a multitude of sensors are required for the robot to be autonomous. Not only do we need multiple sensors there are many sensors that we can use for the same application. In this section we research which specific sensors from which vendors work best for this application.

4.4.1 Ultrasonic Ranging Module HC - SR04

This is a sensor that comes from the company elecFreaks. This sensor works off an IO trigger pulse at a high level that must run for at least 10 microseconds. This sensor has a working voltage of 5 DC volts, and a working current of 15 mA. The range of the sensor has a minimum of 2 cm and a maximum of 4 m, and the measuring angle of the sensor is at a 15-degree cone radius. Lastly the dimension of the sensors are length 45 mm, width 20 mm, and height 15 mm.

The HC - SR04 Sensor can be seen and shown with its associated pins in *Figure 12*. The 4 pins there being the Power, Signal, and ground. The signal pin is represented by two pins: the trigger and echo pin; these two pins communicate with one another to deliver and receive messages.

The trigger pin and an echo pin which work off each other with a timing scheme that can be shown in *Figure 13*. In *Figure 13* it shows a 10-microsecond pulse is sent to start the ranging which sends an 8-cycle burst of ultrasound at 40 kHz and raises the echo. The range is defined as the time interval between the sending trigger and the receiving echo. The ranging can be calculated with multiple different formulas based on the unit of measurement you may want. For units of centimeters the formula is the microsecond pulse you send divided by 58 and for inches the formula is the same, but instead of 58 the number is 148.

Figure 12: HC - SR04 Sensor

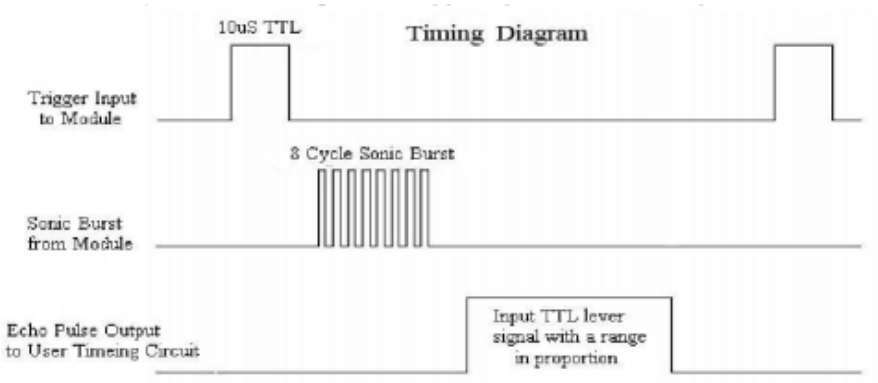


Figure 13: HC -SR04 Clocking Diagram

This sensor possesses many advantages for our autonomous robot. A big advantage is the given information of the detection range. With this information we can plan accordingly and understand the limitations the sensor may possess. One other advantage is something that is common in these types of sensors and that is the low power consumption and size. The small size and small voltage and current are always huge advantages especially with our autonomous robot that have multiple of these sensors. A disadvantage of this sensor is rather than one signal pin, it is split up into two separate pins: trigger and echo. This brings an added complexity to the sensor to ensure that the input signal is being sent correctly and that it is being read correctly.

4.4.2 Parallax Ping

This is an ultrasonic sensor from the company Parallax. This sensor provides measurements from a range from 2 cm to 3 m. It is also easy to interface with microcontrollers and only requires one I/O pin. This sensor is also supplied by 5

volts with a current that typically draws at 30 mA with the max current draw set to 35 mA. This sensor has communication with a bidirectional TTL pulse allowing for both 3.3- and 5-volt logic as well as the input trigger is a positive TTL pulse that typically runs at 5 microseconds. The pin out is 3 pin configurations with power, ground, and signal, respectively. Lastly the dimensions are set at 0.84 in x 1.8 in x 0.6 in, height, width and diameter and it weighs 9 g.

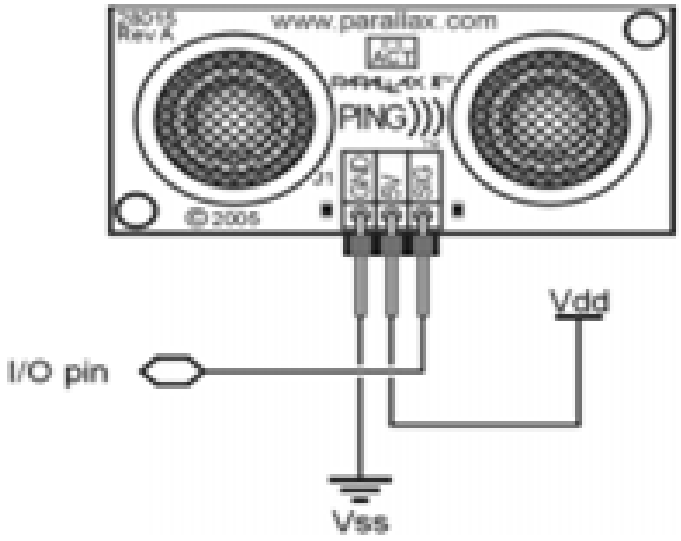


Figure 14: Parallax Ping Sensor

For our autonomous robot, this sensor has clear benefits and advantages. The low weight and relatively small size are a huge advantage to ensure the robot is not over cramped and heavily weighed. The low power consumption allows for better power consumption and allows power to be distributed to more needed components and/ allows for a redesign to use smaller power source and in turn saving cost. A potential drawback is that the range detection could potentially be too small to accurately read and properly sanitize the appropriate areas.

4.4.3 US-100

This is a product that comes from Adafruit that is very similar to HC-SR04, but with some added features. In the US-100 it can run on either 3 volt or 5 volts, thus making no need for any logic level computation, simply it runs on the power given by the microcontroller. The US-100 can run like a HC-SR04, or it can run in a Serial UART mode. The two modes are determined and dictated by a jumper on the back of the sensor. This jumper along with the sensor and its components can be seen in *Figure 15*. When the jumper is taken off this sensor is exactly like the HC-SR04, with all the respective pins like trigger and echo. When the jumper is in place then a 9600 Baud UART is used to communicate with the sensor.

The US-100 runs on a typical voltage range of 2.4 - 5.5 Volts DC and a drawing current of 2 mA. The radius of measure that the sensor can perform is a little less than 15 degrees and the accuracy is best at 0.3 cm with about a 1% margin of error. Lastly the sensor's dimensions measure in at 45 x 20mm with a weight of about 9 grams

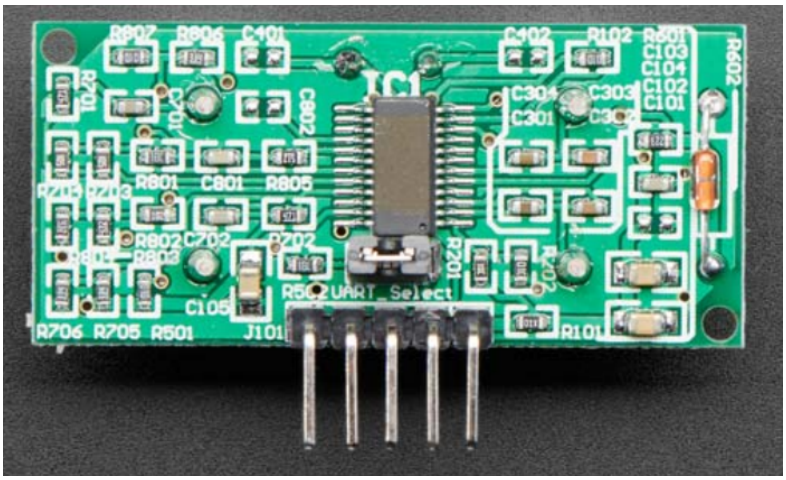


Figure 15: US-100 Sensor

For our autonomous robot, this sensor could be utilized very well. The US - 100 has all the advantages of the HC -SR04, but with added features specifically that with the voltage levels and no need for any voltage switching. One other advantage is that with or without the attachment of the jumper the sensor is compatible with Arduino and CircuitPython. This makes it possible for the sensor to run using the standard HC -SR04 configuration or allows UART communication to be utilized with the microcontroller. A possible disadvantage with this sensor is that it does have a lower radius of measure than some of the other sensors and the price point of this sensor is also listed at about double what the SR04 is listed as.

4.4.4 IR Break Beam Sensor

This is an IR sensor from Adafruit and specifically this is the version with 5 mm LEDs. This sensor works at a range of 3.3 - 5 VDC with 5 VDC generating greater range. The range distance at full 5 VDC is about 50 cm, and it draws about 10 mA at 3.3 VDC and 20 mA at 5 VDC. The receiver in this sensor is an "open collector transistor output" meaning that a pull resistor is necessary to read a digital signal off the wire. This transistor output has a current capability of 100mA sink. The sensor has a response time of less than 2 microsecond and the receiver angle of measurement is about 10 degrees. Lastly the dimensions of the sensor are 20mm x 10mm x 8mm and a total weight of 6 g including the two 3 g halves of the sensor.

For our project, this sensor can be very helpful in its ability to discern from complex surfaces and be able to calculate the distance from there and allowing for the proper response. However, there are a few disadvantages to this sensor. One is the power and current it draws is not efficient with the relatively small output of range that it produces. The sensor also has a very small angle of measurement making it a complete possibility that our robot using this sensor can possibly miss a target or object and spray when it was not supposed to spray. Lastly the biggest drawback is that this specific part is listed at quote stock with a note stating that expected stock to be in soon. When contacted about this there was no answer within a couple of days which is a bad sign if the sensor was chosen by us and tried to be bought later. This combined with the other option stated is most likely why this sensor may not be chosen but is good to reference and compare to other sensors that may be chosen in terms of technical ability, pricing, and availability.

4.4.5 LaserPING Rangefinder Module

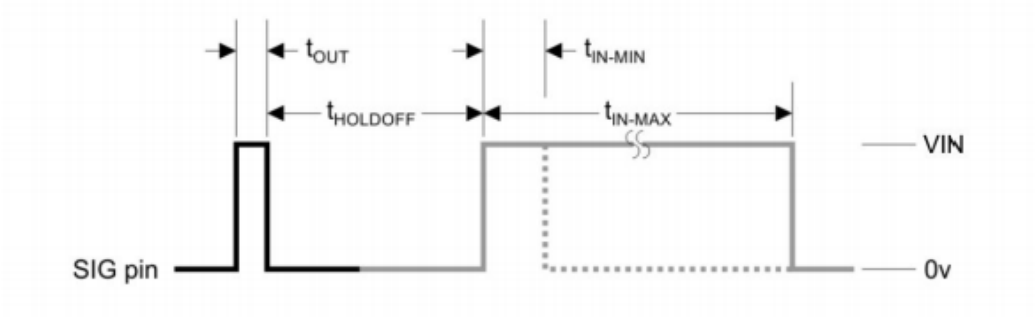
This is an infrared sensor from the company Parallax. This type of sensor is best utilized for distance measurements of either stationary or moving targets. This sensor can be utilized by most any microcontroller using its PWM mode or serial mode. Along with this are built in co-processors in the sensor that ensure the rate logic levels. This means any 3.3 volt, or 5-volt logic microcontroller can be used without the need for any voltage switcher and dividers.

This sensor has a 850 nanometer VCSEL (Vertical Cavity Surface EMitting Laser) with a range of 2 - 200 cm and a resolution of about 1 mm. The communication is a PWM set at idle low, or a serial 9600 baud rate set at idle high. The refresh rate for these modes is 15 Hz for the PWM mode and 22 Hz for the serial mode. This sensor draws about 3.3-5 volts and 25 mA with an operating temperature of -10 to +60 degrees Celsius. The field of view is set at 55 degrees with a PCB dimensions of 22 x 16 mm.

The PWM mode is designed to communicate with 3.3 V or 5 V TTL or CMOS microcontrollers. *Figure 16* shows the pulse width which uses a bidirectional TTL pulse that is interfaced on a signal I/O pin called the SIG. This SIG pin is set to be an idle low while both the input and echo pulse are set to a positive high determined by the VIN voltage which is defined as the specific logic level for the microcontroller. Each set interval of the PWM influences the performance of the sensor. With Pulse width between 115 to 290 microseconds there is reduced accuracy in the measurement. From 290 microseconds to 12 ms is the range with the highest accuracy. At 13 ms this is classified as invalid as the target may be too far, 14 ms is an internal sensor error and right after at 15 ms is a sensor timeout

The pulse width is directly proportional to the distance being measured and it remains relatively unaffected by external conditions. You can convert the

distance in mm or inches by using the Pulse width. The equation for inches is Distance = Pulse Width * 171.5 and to convert for inches just multiply by 6.752 rather than 171.5.



	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min)
	LaserPING Sensor	Holdoff	$t_{HOLDOFF}$	750 µs
		Return Pulse Minimum	t_{IN-MIN}	115 µs
		Return Pulse Maximum	t_{IN-MAX}	15 ms
		Delay before next measurement		65 ms

Figure 16: PWM of LaserPING

For our project, this sensor could work very well as it can accurately read and detect stationary and moving targets. For an autonomous robot maneuvering around a room, it encounters both kinds of targets and to be able to detect both kinds and respond appropriately is a key feature and a deliverable we are trying to reach. A sensor like this with its accuracy can help in doing so. A disadvantage they may be prevalent in the future if this sensor is chosen is the limited distance the sensor itself can measure it. If the range is tried to be extended by increasing the pulse width, we can enter the higher ms times where the sensor causes some errors and eventually reach a sensor timeout. Other times can also give inaccurate results and readings, which in our project can equate to the robot spraying when it's not supposed to and vice versa.

4.4.6 GP2Y0A41SK0F

This is an infrared distance sensor from the manufacturer SHARP. This sensor is composed of a position sensitive detector (PSD), infrared emitting diode (IR-LED) and a signal processing circuit. This sensor is very resilient to any changes that may occur from the reflectivity of the object, temperature and prolonged operation duration. This is largely due to the triangulation method that is used to measure distance and detection. This sensor can also be used as a proximity sensor because the sensor outputs the voltage in correspondence with the detection distance.

This sensor is very typical in autonomous robots specifically those focused around cleaning and sanitary. The sensor has an operating supply voltage of 4.5 to 5.5 volts and an average current draw of 12 mA with a max current of 22 mA. The distance measuring range is from 4 to 30 cm with an operating temperature of -10 to +60 degrees celsius and lastly the package size is 29.5 mm x 13.0 mm x 13.5 mm.

4.4.7 GARMIN LIDAR-LITE V3

This is a LIDAR sensor from the manufacturer SparkFun Electronics. This sensor has a working DC voltage from 4.5 to 5.5 and has a current consumption of 135 mA during continuous operations and 105 mA during periods of idleness. In terms of performance this sensor has a range of 40 m with a resolution of about +/- 1 cm. The accuracy of this sensor from a distance less than 5 m is about +/- 2.5 cm in terms of the measurement. The accuracy of the sensor from a distance greater than 5 m is about +/- 10 cm, with about a +/- 1% mean and ripple of the max distance. This sensor has a typical update rate of about 270 Hz, and a 650 Hz fast mode, but note that in the fast mode there is a reduced sensitivity in the sensor. Lastly the laser used is at a 905-nanometer wavelength with total laser power of 1.3 W. The laser operates at a pulsed train with a frequency of 10-20 Khz with 256 pulses max.

The interface of this sensor has 3 main components: the I2C, the PWM and the external trigger. The I2C bus connects the sensor to the microcontroller and activates the communication link. The I2C interface runs in a fast mode that is set to 400 kbits per second with an internal register access & control with a default address set at 0x62. The PWM works in conjunction with the external trigger input. The PWM output works off an output that is proportional to the distance at 10 microseconds per cm.

The sensor also has an associated wiring harness with each corresponding wire color with a function. It is also possible to repurpose the wiring harness for any personal needs by obtaining the listed components in the datasheet from the specific suppliers and modifying the port’s function.

This sensor works extremely well in its accuracy and range. With a range of 40 m this could be applied to our autonomous robot, and it can work extremely well as this sensor can also accurately scan and access complex and different environments. This sensor also has a great accuracy across its entire range making it one of the more efficient distance sensors available. Some of the biggest drawbacks with this sensor is the large amount of current it draws and the pricing. The current is relatively high compared to many other distance sensors in the market, as well as the pricing is extremely high compared to other distance sensors. If our robot may need more than one sensor, this sensor could hinder both our performance and our budget.

4.4.8 GARMIN LIDAR-LITE V4

This is the updated LIDAR sensor of the Garmin LIDAR-LITE V3. This sensor comes with a Bluetooth 5.2 SoC called the nRF52840. The SoC is compatible with the ARM Cortex processor that has 1 MB of flash memory and 256 KB of RAM. There is also a 2.4GHz multiprotocol radio and a S340 SoftDevice which helps support ultra-low power wireless technologies.

This sensor has an operating voltage from 4.75 to 5.25 VDC, and current consumption of 85 mA during operations and a 2 mA under idle conditions. Along with this is the range the sensor can measure which is from 5 cm to 10 m. This sensor runs on an I2C interface that requires a max signal voltage of 3.3 for all the different pins. The sensor can also run on an ANT which is a wireless network running in the 2.4 Ghz ISM band. The I2C has an update rate that is greater than 200 Hz and the ANT also has an update rate of 200 Hz, but it's only to a 90% reflective target indoors at 2 m in normal operating conditions. Lastly the dimensions of this sensor are the following with the format being length, width and height; 55.2 x 24 x 21.2 mm. The sensor also has a weight of about 14.6 grams and an operating/storing temperature of -20 to +60 C (-4 to 140 F) and -40 to +85 C (-40 to 185 F) respectively.

This sensor is extremely similar to its earlier model the V3. Like many LIDAR sensors this sensor has an extremely great range and accuracy. With this sensor all the same conditions stated for the Garmin LIDAR-LITE V3 apply. For us to use either the V3 or V4 sensor for our robot we'd have to extensively breakdown how we want performance to go. Using these LIDAR sensors would give the best performance out of any distance sensor, but the pricing is a big obstacle and given that we may need multiple sensors on our robot it makes that pricing an even greater issue. There is also the argument that this sensor provides too much performance than what is necessary or what can be achieved easily with another sensor that is better priced. All these things make both the Garmin V3 and V4 an unlikely sensor choice for our project.

4.4.9 Final Sensor Selection

Through this section 3 main classes of distance sensors have been studied. They have been the ultrasonic, IR and LIDAR. Each possesses positives and drawbacks for our autonomous robot and ideally any of the sensors would work well. However, the distance sensor we are going with is the HC - SR04 Sensor.

This sensor has a great distance detection range at a minimum value of 2 cm and a maximum of 4 m. Along with the detection range is the sensor's great power and current consumption. This is great for the overall power consumption of our robot and since this sensor meets our performance criteria it is a great choice. The lower power consumption works well for our project because we most likely need multiple sensors in our design so having a low power

consumption sensor is definitely an advantage. The other big factor about this sensor is that it is priced very well. Compared to the competitors in the distance sensor field this sensor has a great price point. Both places like Mouser and Digikey have available stock and pricing at 3.95 and 5.95 respectively. These factors are some of the key reasons why this is the distance sensor that is being chosen.

Sensor	Voltage	Range	Price
HC - SR04	5 DC	2 cm - 4 m	$3.94
Parallax Ping	5 DC	2 cm - 3 m	$29.99
IR Break Beam	3.3 - 5 DC	50 cm	$5.95
LaserPING	3.3 - 5 DC	2 cm - 200 cm	$22.99
Garmin Lite V3	4.5 - 5.5 DC	40 m	$131.25
Garmin Lite V4	4.75 - 5.25 DC	5 cm - 10 m	$68.75

Table 9: Distance Sensor Comparison

4.4.10 Optomax Digital LLC200D3SH-LLPK1

The previous sections focused on many different distance sensors and comparing the advantages and disadvantages each one possessed. With the autonomous robot it is necessary to have the proper distance sensors that accurately detect and be able to spray when necessary and not spray in required situations. One other aspect of this robot is being able to accurately detect when the spray solution is under 5%. Now this less than 5% threshold can be calculated based on the spray container's total volume, but there still needs to be some way to measure this value and to know when this condition is met and the robot needs to return to home to refill.

The Optomax digital is a liquid level switch that contains a sensor that has an infrared LED and a photo transistor. The infrared sensor measures the amount of liquid when submerged in the liquid. When the sensor is in the liquid the light leaves from the infrared LED and the photo transistor turns off. During this state the sensor detects the amount of or lack of liquid inside and reports that. It is very versatile and is not affected by abinet lights or any kind of foam or bubbles that may be confused for actual liquid.

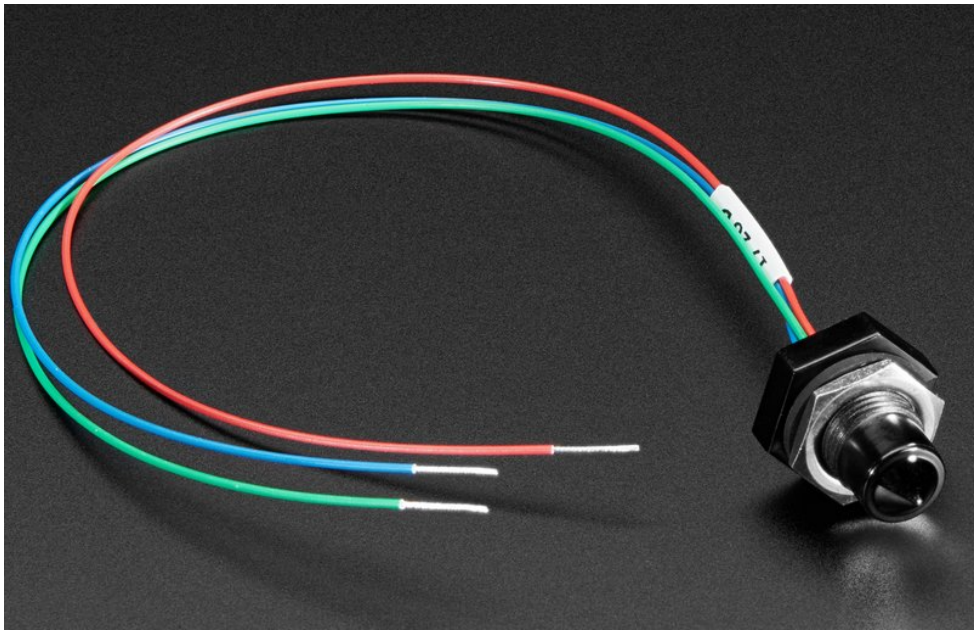


Figure 17: Optomax Digital Liquid Sensor

This sensor has a working voltage of 4.5 VDC and 15.4 VDC, and a current consumption of 2.5mA. The source current also has a current consumption of 100 mA. The operating temperature is set to be at -25 C to +80 C, with a storing temperature of -30 to +85. This sensor has listed many features that makes it a very versatile sensor. This sensor has a reverse polarity protection as well as the wide spread voltage range. The sensor is also compatible and microcontroller friendly at both logic levels of 3.3 and 5 V. This device is also a solid state which entails that it is fully enclosed and no parts are moving around which may cause unreliability.

This sensor, as mentioned before, serves to measure the amount of liquid in the chosen spray bottle and monitor the levels until it reaches below the designated level which informs the user with an LED and prompts a return to home command. This sensor has an extremely fast response time so the values stay updated and the robot moves accordingly and accurately. The sensor has a wide supply voltage range which works great in designing our robot so we have some room to redistribute the power consumptions. This sensor also has a great detection of many different liquid solutions so with possible different sensor selections that may occur down the line this sensor still be able to act accordingly.

Lastly, the biggest advantage is the size and pricing point. Compared to many different sensors of this kind this specific sensor is priced very well and is readily available. Along with the great price point the size of the sensor is very small making it something that can be easily placed anywhere and along with the small size is the small weight of the sensor. With a measured weight of 8.4 g the

sensor is all around relatively small and does not affect the performance of the robot. We were able to use this liquid level sensor in our final demonstration to accurately tell when the bottle is getting low on liquid and then trigger the deliverable so that the MSP430 beeps for five seconds and then triggers the entire systems shut down.

4.5 Sprayer Selection

The sprayer is one of the most important parts to this project. In this section we are comparing 3 different sprayers with different form factors and different mechanisms to decide which one suits this application the best.

4.5.1 Airchr Automatic Spray Bottle

This is a motorized spray bottle that is made from a company called Honorway that is in Shenzhen, China. The bottle consists of an ABS plastic body and a high-density polyethylene spout. This sprayer has a rated voltage of 3.7V and includes a rechargeable battery with a capacity of 2000 mAh. The reservoir that is included with this sprayer has a capacity of 1 liter and is composed of ABS plastic. This unit is charged over USB with a charging DC voltage of 5V, the estimated charge time is approximately 3 hours, and the estimated runtime is about 5 hours. The nozzle has an adjustable spout that can be turned to adjust the spraying pattern from a fine mist to a single stream. This adjustment is ideal as the user can adjust the spray pattern for various situations.

Figure 18: Airchr Automatic Spray Bottle

The ideal part of this sprayer is that there is no trigger to pull to activate the mechanism. To activate the spraying mechanism a button press is required, simplifying the device. As shown in *Figure 18* the activation button is just below the spout. This enables our group to be able to easily tap into the contact switch and hardwire it into our mainboard. *Figure 18* also shows the simplistic exterior of the device, with only having a rechargeable battery that is USB powered. The

sprayer head includes light that indicates how much battery power remains. Our group can tap into this led and utilize its signal to indicate on a larger led to the user that the sprayer needs to be recharged.

One of the drawbacks of this design is that this is a whole unit reservoir and pump combo. It would be difficult to modify this device if a different reservoir is required for this application. With that in mind this would be a device that is a possible candidate for the final build as it checks most of the requirements needed, such as the mechanism used. the bottle size, and the low cost nature of the device.

4.5.2 PetraTools Automatic Battery Sprayer

The PetraTools Automatic Battery Sprayer is similar to the previous Airchr Automatic Spray Bottle except that it can detach from its included reservoir and is able to use any reservoir that is threaded to fit its cap. As shown in *Figure 19* the spraying mechanism and nozzle is separated from the reservoir, only connected by a tube to carry the fluid. This would enable us to be able to better position the sprayer head in the most optimal area for cleaning and be able to put the reservoir away from any of the sensitive electronics.

This sprayer is powered by two AA batteries housed in the sprayer head, the included reservoir has a fluid capacity of 32 ounces or 909 ml. The tube extends 9.5 inches or 42 cm into the reservoir meaning if another reservoir is needed the included tube should be able to reach the bottom with ease. The exterior tub that connects the spray head to the reservoir is 36 inches or 91.5 cm long. The spray head does have adjustments to go from a mist to a full stream, like the Airchr. The mist spray flow is 6 ounces/minute, and the full stream flow is 8 ounces/minute or 170 ml/minute and 227 ml/minute, respectively. The pump operates on 3 volt power with a pump speed of 4200 rpm. This product weighs approximately 15.8 oz or 448 g unloaded and 48.8 oz or 1.27 kg with a full load of solution.

This sprayer utilizes a trigger switch, where there is a trigger to pull to activate the device but, this trigger only activates a switch to turn the device on. Although a bit more difficult, this system would be manageable to tap into and control with our main board.

Figure 19: PentraTools Automatic Battery Sprayer

Unfortunately for this sprayer it only runs on two AA batteries, this is both a pro and a con. The good thing about these batteries is that they are readily available and easy to replace. Adversely, it would be much harder to implement a battery indication system for the AA batteries in this sprayer. The cost of this sprayer is similar to the cost of Airchr sprayer only varying by approximately $2, this puts this product in contention for the ideal automatic sprayer for this application.

4.5.3 Paint Sprayer

A paint sprayer such as the one shown in *Figure 20* would be the most ideal type of sprayer to be used as it has the high and even pressure characteristics we are looking for in the sprayer. Like the other sprayers this nozzle is adjustable, but this is only done by changing the spray tip and not by twisting. The paint sprayers also contain an adjustable flow control knob to dial in the exact flow rate that the specific application requires. This specific unit has a 1 liter reservoir which is like many units researched.

Figure 20: REXBETI Ultimate-750 Paint Sprayer

Unfortunately for this unit another reservoir cannot be retrofitted, and the included reservoir must be used. Another main disadvantage of these types of units is the power required. The unit in *Figure 20* requires 500 watts of power at 120 volts and to supply power to a unit such as this would require an extensive power management overhaul. Although there are battery operated versions of paint sprayers the sheer weight of the unit with a full load of cleaning solution would exceed our parameters for being lightweight. There is also no simple way of tapping into the trigger of this unit as it appears the trigger is a variable analog trigger. This would mean the further down the trigger is pulled the more the flow rate increases.

The cost of the unit in *Figure 20* is approximately $50 which we have found to be the average price of entry level paint sprayers, and the average price of the entry level battery operated paint sprayers are approximately $100. We believe the price of this sprayer exceeds our budget.

4.5.4 Sprayer Final Selection

Through the research done on the sprayers above our group went with the PentraTools Automatic Battery Sprayer. This was due to the sheer versatility of this sprayer. We have the option of placing the reservoir almost anywhere on the robot we choose. With this option we also have the opportunity to change the reservoir for a more suitable one if we deem the included reservoir to be unfit for our application.

Overall the decision was mainly between the PentraTools Automatic Battery Sprayer and the Airchr Automatic Spray Bottle. The Pentratools spray bottle edged due to versaltilty, the paint sprayer would have been a good option if the device was not as bulky, did not require excessive amounts of power and had a simpler trigger mechanism.

This was the sprayer that we opted to use in our final demonstration in the system. This was a good choice as we were able to have it so the sprayer could directly make it to the relay system on the PCB. We did have some trouble with the amount of liquid that would come out of the nozzle and the power of spray. We could have opted for a more powerful car nozzle and jet sprayer to fix this issue, but this sprayer got the job done more accurately.

Sprayer	Mechanism	Reserv oir Size	Power Source	Misc
Airchr	Single Button Switch	1 L	Rechargeable 2000mAh Battery	Micro-USB interface
PetraTools	Trigger Switch	909 mL	Two AA Batteries	Can use any reservoir that tube fit
REXBETI	Variable Trigger Switch	1 L	110V Wall Outlet	High Pressure with variable spray pattern

Table 10: Sprayer Comparison

4.6 Cleaning Solution

For the cleaning solution there are a couple of routes we can take. We can choose a medical grade disinfectant cleaner, we can go with a consumer grade multi-purpose/multi-surface cleaner, or we can choose to make our own solution from isopropyl alcohol and water or even use a mixture of hydrogen peroxide and water.

4.6.1 Hospital Grade Cleaning Cleaner

This hospital grade cleaning solution from Micro-Scientific named Opti-Cide3 is a disinfectant cleaner and sanitizer solution. It advertises a 2minute kill time on clinical surfaces and as Anti-Viral, Anti-Bacterial, Anti-Fungicidal, and kills Tuberculosis. Overall, this product shows it can safely sanitize any surface.

With a product such as this depending on the application it can be used as is with no mixing or it can even be diluted to clean more conventional surfaces. This would be up to the user on how strong of a mixture would be needed for the application. Unfortunately, with this product there is no scent associated with it, as some reviews indicate it leaves a “strong medical alcohol” smell when used. Although it is yet unsure if mixing with a scent and water would change this for the better.

Figure 21: Micro-Scientific Opti-Cide3

Some advantages of the Micro-Scientific Opti-Cide3 is that it is advertised as a low alcohol formula, is EPA registered, and is health care grade. The key to this product is health care grade, this would lead to believe this product is made of higher standard versus another product that is not health care grade.

Cost wise this product is approximately $24 per a gallon and depending on the application this could last a decent amount of time with the sprayer used, since automatic sprayers are more efficient at spraying liquid. One important factor is if this product has any adverse effects on the sprayer being used, as it is noted that the product has an alcohol smell. The alcohol in this product could cause the plastic in the sprayer to deteriorate and cause the automatic sprayer to stop working. This would require further testing as it is noted that ABS plastic would withstand the use of an alcohol based solution under lower temperature situations only.

4.6.2 Consumer Grade Cleaning Solution

For consumer grade cleaning solutions, we found Mrs. Meyer's multi-surface cleaner to be amongst the most popular. With this solution specifically being a concentrated solution, this raises the need to always mix with water before every use. This solution is scented and comes in six scents. This product advertised no harsh chemicals.

Due to this being a household product it is advertised as being “Tough On Dirt” and being able to clean floors, founders and sealed surfaces. With dirt being one of the primary factors in this application the product could prove to be essential. The proper mixture would need to be tested to ensure proper sanitation is occurring. The cost varies from $16 to $18 for a two pack of 32oz bottles for a total of 64oz of concentrated multi-surface cleaner. It is recommended to use 2oz of concentrate with 1 gallon of water. This would mean for these two packs of concentrate we could make approximately 32 gallons of solution. With this line of thought this would be the most cost effective solution to use.

Some reviews indicate it is possible to mix the concentrate with alcohol and water to improve the cleaning capabilities of the solution. This would depend on the way alcohol affects the reservoir and the automatic sprayer.

4.6.3 Self-Made Solution

If this application requires it we could make our own cleaning solution from a mixture of vinegar, alcohol, peroxide, and water. Each of these chemicals is known to have cleaning properties. Simply diluting each of them or diluting a mixture of these chemicals into water with one part chemical and five parts water could be suitable for this application.

All the necessary chemicals are readily available, and the solution could be fine tuned to match the situation that would arise. The downfall of this idea is that either a bulk of this cleaning solution is made at once or the user would mix this solution every time the robot would need a refill, which would often depend on how dirty the area is.

Although unlikely due to the amount of testing that would be needed to ensure this cleaning solution would work, there would be a better way to use this time elsewhere in this project.

4.6.4 Final Cleaning Solution

For this application we saw that the hospital grade cleaning solution Opti-Cide3 made by Micro-Scientific works well. This is because it is a stronger cleaner in general. On their datasheet it lists the numerous microorganisms that this product kills and its plethora of applications. This gives us confidence that this product be able to properly clean and sanitize an area of all germs and dirt.

Cleaning Solution	**Cost**	**Effectiveness**	**Misc.**
Medical Grade	$24 per Gallon	HIghly effective	Ready made solution to be used immediately
Consumer Grade	$36 per Gallon Concentrate	Moderately effective	Concentrate to be diluted
Self-Made	$10 per gallon	Moderately effective	Needs to be self mixed to proportions

Table 11: Cleaning Solution Comparison

4.7 LED Selection

In this application there are two LED functions, once function would be used to alert the user that the reservoir is at less than 10% capacity, and the second function for the LEDs would be to alert the user of the robot's presence to help with avoidance. These LEDs would be soldered onto the main board and should be bright enough to be noticed in a well-lit environment.

4.7.1 Addressable RGB LEDs

These LEDs can be almost any color on the visible light spectrum. As shown in *Figure 22* these LEDs have a 5 pin layout and are to be soldered to the main board. These specific LEDs in *Figure 22* also have a built in diffusion layer within the 5mm package. They require 4.5V to 6V DC power and have an internal frequency of 800 KHz.

Figure 22: Sparkfun Electronics 5mm Addressable RGB LED

The disadvantage of utilizing these LEDs is that they need to be coded for each specific color the user desires to be displayed. The coding required to make a set of these LEDs display the colors we desire would over complicate the board. This would cause more complexity within the system and possibly take up unnecessary CPU performance.

4.7.2 RGB Color Beacon

Like the individual addressable RGB LEDs shown in *Figure 23* this beacon can display about 16.8 million colors. This utilizes a four wire I2C sensor for communication with an I2C maximum bus speed of 100 KHz. This unit requires 5V at a max current of 50 mA. The unit utilizes a three wire analog interface and four wire I2C interface to communicate with the main board. The device has mounting holes at 48mm x 24mm.

Unlike most LEDs this unit has built-in analog sensors which include a rate gyro, optical distance sensor, touch sensor, light sensor, and magnet sensor. The unit

also includes a four wire digital I2C sensor which includes a compass, integrating gyro, range sensor IR locator 360, sound generator IR seeker V3, color sensor, and color beacon.

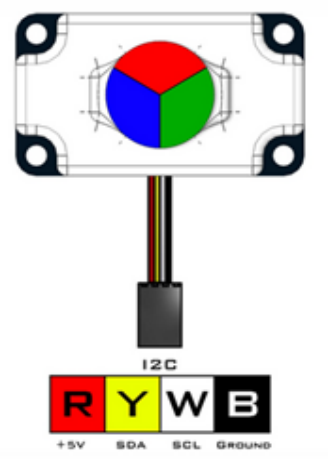


Figure 23: Modern Robotics RGB Color Beacon

Although it is unlikely we will use this unit for LED lighting, it is very probable this unit will be used for its multitude of sensors included in it. With its cost being just under $20 and a combination of sensors with a sound generator this would be a valuable part to add in this application. So long as we do not exceed the limit of the I2C protocol.

4.7.3 Ultra Bright LED Kit

This is a kit that contains 25 LEDs consisting of red, green, yellow, blue, and white (5 LEDs for each color). These are very simple two pin LEDs that would be soldered onto the main board. All these LEDs are 10mm and diffused for even lighting. The red LED has a brightness of about 900 millicandela with a max forward current of 20 mA and a forward voltage of 2V. The greed LED has a brightness of about 1500 millicandela with a max forward current of 20 mA and a forward voltage of 3V. The yellow LED has a brightness of about 900 millicandela with a max forward current of 20 mA and a forward voltage of 2V. The blue LED has a brightness of about 550 millicandela with a max forward current of 20 mA and a forward voltage of 3V. The white LED is the brightest and has a brightness of about 2000 millicandela with a max forward current of 20 mA and a forward voltage of 3V.

With LEDs such as these we should be able to attain the desired brightness for a user to be able to readily notice the robots presence. The implementation of these LEDs would be rather simple as they are only two pins on these. The coding required would be much simpler rather than using the Addressable RGB LEDs. As we would only have to code the on and off frequency to simulate a flashing LED.

4.7.4 Final LED Selection

With most LEDs being very similar the differentiating factor for LED selection was the brightness and the size of the LEDs. The Ultra Bright LED Kit from Gravitech would make the selection for the LEDs to be soldered to the main board. Pulling miniscule power for being 10mm LEDs, they are very power efficient. With these large LEDs we are confident in their abilities to draw attention to the robot and notify the user of the robot's presence and if the reservoir would have to be filled.

We had to opt out of using these LEDs and instead had test LEDs on the PCB and used the LEDS on the MSP430 development board to meet this requirement. We had the one red and one green LED on the board flash when the ultrasonic was within range and had the other flash when the color sensor detected the color green. When both requirements were met both LEDs would be flashed on, showing both the red and the green LEDs.

LEDs	Color	Form	Misc.
Addressable RGB	16.8 Million Colors	5 pin connection	Must be programmed to display colors
RGB Beacon	16.8 Million Colors	I2C connection to microcontroller	Has multiple sensors that can interact with the microcontroller
Ultra Bright	Single Color (5 colors in the pack)	2 pin connection	Only requires power

Table 12: LED Comparison

4.8 Speaker Selection

In the application the speakers would be used to alert the user of the robot's presence and that is it cleaning surfaces. These speakers should be loud enough for a person to be able to hear it from more than 10ft away. These speakers should also be power efficient and small enough to implement onto the robot.

4.8.1 Degraw DIY Speaker Kit

In this speaker kit it includes two 4 ohm, 3 watt speakers with the dimension of 31mm x 70mm. The kit also includes a mini class D digital audio amplifier (PAM8403) that can be wired to the main board or a raspberry pi, the dimension of the amplifier is 28mm x 20mm. The amplifier includes an on board volume knob that can also be used to turn off the speakers.

These speakers are similar to speakers often found in computer monitors and small televisions. Which means for their size these speakers should produce adequate sound for this application, while maintaining a small footprint on the robot.

The amplifier has an operating voltage of 5V and could be powered from a USB power supply or 3 AA batteries. The output power is 3W + 3W (2 channel) for each speaker connected. There are mounting holes on the PCB of the amplifier and there are included breakaway headers for more connections.

Figure 24: Degraw DIY Speaker Kit

With this kit it would be relatively easy to implement into the system as everything is included with the kit. The only part that is not included with this kit is the power supply, but a simple 5V power header on the main board would suffice for this device. The only drawback would be that it is possible that these speakers could draw too much power and cause us to increase our battery capacity if this is a solution we choose to go with.

4.8.2 MakerHawk 2pcs Speaker

With the MakerHawk speakers there is not an amplifier included with these and can be directly connected to the main board or a raspberry pi. This would simplify the wiring with the exclusion of the amplifier. These speakers weigh approximately 40 grams and are 31mm x 28mm x 15mm. The speakers can vary from 4 ohm to 8 ohm depending on the selection chosen at checkout and they draw 3 watts of power. The frequency response is 500 Hz to 20 kHz.

An interesting capability of these speakers is that they can operate at 5V or 3.3V depending on the installation method. That added versatility can further simplify our system by having multiple devices with the same rated voltage.

With the utility of these speakers, they have a smaller footprint than most speakers and advertises good sound quality, but with that we are unsure about the loudness of this speaker since it plugs straight into the raspberry pi.

4.8.3 Gikfun 1.5” Full Range Audio Speaker

These speakers are like the MakerHawk speaker except that they are slightly larger and do not have an enclosure. This product is the bare speaker alone, all wiring has been provided and done by the user, as well as an enclosure and mounting. There are no mounting holes or hardware so we would have to figure out how to place this device onto the robot.

Figure 25: Gikfun 1.5” Audio Speaker

These speakers appear to be louder than the MakerHawk speakers, but it looks like these would need an external amplifier to be able to power them as it is unknown if they can be powered by the raspberry pi. Unfortunately for these speakers there are too many external add-ons that would have to be purchased in order to implement these speakers into the system, raising the price of implementation higher than what our budget is for speakers.

4.8.4 Final Speaker Selection

We believe that the Degraw DIY Speaker Kit would be the best fit for our application. The speakers are easily mountable to the robot, and they have a small enough footprint to mount. There is an included amplifier, so there is no need to do additional research to find an amplifier that would match up with these speakers while attempting to maintain a low power draw. The only drawback of these speakers is that they could be high power draw, but if their speakers are louder than expected then we could use the volume knob on the amplifier to turn down the volume which should in turn lower the power usage as well. The amplifier also has power headers so we can easily tap into that with our main board.

The MakerHawk and the Gikfun speakers did not meet our requirements and were not as good of a value as the Degraw speakers. All three of these speakers cost within the $10 to $15 range and only the Degraw comes bundled with an additional amplifier. The only reason they would need an additional amplifier would be if they needed more power to produce a higher volume and it would be better to have more headroom for more volume instead of replacing a whole unit for being too quiet.

For our final demo in senior design 2, we had to look at another speaker that had the same specifications as the Degraw as we were unable to get it working on the PCB board in time. We then looked and saw that the JBL speaker would be the best fit, as we would meet the same specification as the Degraw speaker and our group already had one in our possession. This made it easier on our budget and was very easy to mount on the system. With virtually the same specifications as the Degraw this was a clear choice and replacement for the speaker selection.

Speaker	**Form**	**Operating Voltage**	**Impedance**	**Misc**
Degraw	31mm x 70mm	5V	4 Ohm	Included amplifier
MakerHawk	31mm x 28mm	3.3V or 5V	4 or 8 Ohm	Operate at 3.3V or 5V
Gikfun	mm diameter	5V	4 Ohm	Not Prewired
JBL Mini	31mm x 70mm	5V	4 Ohm	N/A

Table 13: Speaker Comparison

5. Related Standards and Realistic Design Constraints

In the following section, we discuss standards and constraints related to our autonomous sanitation robot. Standards and constraints are applied to what consumers and businesses utilize everyday. The guidelines listed by the IEEE and other various organizations are followed by us to make sure we produce a quality product. Any standard or constraint not followed could lead to consequential problems like disposing batteries in the incorrect way is hazardous to the environment.

5.1 Standards

Standards surround our daily life in almost every way. A big picture overview of standards can represent many different meanings. It can reflect safety measures to the performance of the materials or products. Everything around us is governed by standards like the homes that we live in. Organizations develop standards that allow us to live an easier life like any organization that has created building codes and standards. These rules create a safe environment that allows our society to function by building jobs made specifically for regulating these rules so that consumers can then buy and use quality products that we know should adhere to a safe life.

Any product has to follow multiple guidelines of rules so that it can be approved by organizations that oversee that area. These rules have restrictions so that an item cannot destroy or have a negative impact on society and the environment. For example, the USDA (U.S. Department of Agriculture) regulates all aspects of crops produced in the United States. Standards include: limiting the usage of pesticides and safety for a crop. Without these standards, the majority of the population could develop diseases and become sick from corporations/farms concocting dangerous food items to save or make more money.

Our autonomous robot has all of its parts that follow organizational standards such as IEEE (Institute of Electrical and Electronic Engineers). Our robot without any standards could pose risks to the environment such as having a battery within the chassis that could have a massive carbon footprint. In other ways, the robot could be a safety hazard to others by spraying a disinfectant that is unregulated by the health administration. The important regulations are technical regulations that would definitely apply to our disinfectant spray that already went through a grueling process to be deemed suitable for our health and the environment.

The voluntary standards applied to our project are not solely based on environmental and safety hazards. They also apply to the reliability and quality of the build. Our project cannot work only for a partial amount of time. It must work consistently to provide a user a well-rounded experience. These standards are captured by how society deems what is worth purchasing. Our product, if

developed for the masses to make money; the manufactured product would not make it very long if no one wanted to buy a good that was failing all the time even though it passed all inspections for the well-being of the environment.

The major components in this project that need to be voluntarily and technically regulated are:

- Motors
- Batteries
- Circuit Board Electronics and Wiring
- Disinfectant Sprayer and Tank

A motor in any device such as a car has to be thoroughly inspected for a lot of reasons. A motor of a small rc-car is just as important because it could lead to the harm of the user if it was made defectively.

When ordering or designing a PCB, it is expected that the creation process needs to be monitored closely. The IPC has a standard for the safe and reliable creation of PCB's. Some of the most important standards created by IPC are listed below.

1. IPC-2581: Anyone that designs a PCB that then be sent to a manufacturer adheres to the standard of submitting data to the manufacturer so that the product can be made correctly every time.
2. IPC-6012B: Sets the standard for the quality of the structure, soldering, and layout design for PCBs.
3. IPC J-STD-001: A standard that can be followed to use materials and processes to create the best soldering connections for a board.

Specifically, these IPC standards are utilized by our team. These standards keep us from making mistakes in designing our PCB before having it sent to be created. The PCB will be used in our Autonomous Sanitation Robot on top of the chassis to control all aspects related to spraying the disinfectant.

IEEE and ANSI have standards for the use of lithium and rechargeable batteries. IEEE 1625-2008 covers our rechargeable battery inside our IRobot Chassis. It is a voluntary standard that covers all levels from the quality of the battery to the reliability when it is in use. ANSI C18.2M covers specifications and safety standards related to batteries. This particular standard gives producers the knowledge that all producers making batteries are able to create very similar products that are safe and effective. This allows for reduced costs and high expectations for consumers because they know that whatever and whoever they buy a product from that contains a battery, that it should be reliable, safe, and of high quality. ANSI lists standards for every possible scenario when it comes to batteries including vibration stress to thermal temperature misuse.

We are adhering to a combination of IEEE and ANSI standards for everything related to our Autonomous Sanitation Robot.

Coding standards are applied to the language we end up using for the Raspberry Pi. Between C, Java, and Python, these languages all have standards that make them function. Coding conventions for each language gives readability to any user that looks at the code in the future.

The Autonomous Sanitation Project deals relatively heavily on the logical side for its software. The sensors alert the chassis to report back to its charging station and maneuver around obstacles primarily while spraying disinfectant. This is important because the logic can be messy and standards can be attributed to develop a good software design.

Each language deals with conventions that are discernable for that specific one. The creation of variables differ etc. These rules have to be followed to have a program that works efficiently.

5.2 Project Constraints

Constraints have to be made for any project ever. Constraints can come from the physical ability of what is possible to the financial limitations. Each constraint listed below leads to the overall analysis of what must be done to complete the project. These constraints are realistic and not outlandish for the real purpose of finishing the project. The constraints we are looking at are listed below.

- Financial and Time
- Environmental, Social, and Political
- Ethical, Health, and Safety
- Manufacturability and Sustainability
- Testing

5.3 Design Constraints

Our Autonomous Sanitation Robot is meant to transport a disinfectant spray and sanitize locations as needed. Therefore the cargo represents a constraint for the IRobot Programmable Robot. The weight cannot exceed a certain amount otherwise the chassis would be bogged down.

5.4 Financial and Time Constraints

Financial constraints are an important factor to the overall quality of the project. It is very important to research all parts beforehand while weighing a cost analysis for each part to determine if it is worth it. After researching and investigating, a project's bill of materials and parts list is necessary to manage the budget within the group. Time constraints are also an important factor because the project

cannot have an unlimited time span just like it can't have an unlimited budget. With this in mind, our project was designed not to have parts that are extremely expensive and an assembly that is so complicated it would take years.

Sticking to the budget is important because spending could easily get out of hand. If the group doesn't follow its pre-planned list of items needed, the project could possibly fail. The more money spent also doesn't necessarily mean a higher quality project.

For our project, our budget split between the four of us is 700. We have a higher budget than our bill of materials in case we end up having to spend more for emergencies. Our IRobot chassis and custom PCB board accounts for a large portion of the budget. The custom PCB board until actually ordered can have different costs depending on how limited the parts are or how complicated our design be. We have also included Misc. electrical components to factor in the possibility of having to acquire different parts based on the failure of other items.

So that we don't go into excessive spending, technologies be researched with a cost analysis in the forefront. We are looking at sensors for our robot to detect objects and humans nearby so that we aren't spraying areas where we aren't supposed to. Sensors can be cheap or expensive so we are focusing on making sure a sensor like the ultrasonic sensor can fulfill our needs because it is inexpensive compared to a FLIR IR camera that costs hundreds. This works in tandem with the design constraints because our project isn't being built directly for the military which would need the best components to output the best service of the robot. This robot could be upgraded in the future by switching out parts that would make it have an even better service.

With time as a constraint, we first have to acknowledge when planning and designing our project that we cannot add so many features that it leads us scrambling with no time left by the end trying to get it working. A time constraint can be expressed in two different ways. There can be a problem by not having enough time to complete the project fully because there were too many complicated objectives. The second time constraint stems from the fact that just like budgeting money, budgeting time is also necessary so that the group stays on track to finish.

That first time constraint is researched beginning with the creation of the idea for the project. That then is analyzed even further by researching it more for the divide and conquer. Our team is represented by a majority of electrical engineers so we discussed at length about creating a project focusing on the electrical side more so than the software side. If we decided on an entirely computer engineering project, we would have to spend more effort learning how to complete complicated coding problems rather than focusing on developing a thorough project.

The second time constraint needs to be more specific than just having the research done by the end of this semester and starting the project by the next. We have set up currently the exact amount of pages needed to be written and therefore researched every week leading up to due dates for the project. By the start of the fall semester, we have a planned calendar system with goals to make sure we are completing our project in a timely manner. This is important because without a schedule during the building of the autonomous robot, we could be in the middle of the semester and not know exactly where we are to finish. That could lead to a disaster by running out of time or really hurrying to finish and neglecting polishing the project.

5.5 Environmental, Social, and Political Constraints

In modern day society, environmental concerns are at all time highs. As technologies get more advanced and products keep getting consumed, environmental pollution is becoming staggering because of improperly discarding electronics like cellphones. These problems have been leading to the cause of global warming and climate change. Solutions have been posed by governments and companies to start recycling a lot and control emissions with standards that require products to have certain restrictions on how they affect the planet. The US EPA is a government agency that is made to protect the environment by tagging restrictions on products that for example burn fossil fuels.

Batteries have started to be at the forefront of discussion for their role in climate change. Batteries are the leading movement to replace fossil fuel consumption in products like vehicles. Batteries that recharge can be effective in reducing nitrogen being emitted into the air. Motors are being ushered into this new era of being powered by batteries. Batteries are a problem if they leak into the environment if not handled properly. Mobile Phones for example contain batteries that can leak and decay into the environment which is toxic to everything. It is hard to recycle batteries because they aren’t like regular electronic equipment. They are notoriously hard to recycle and not available all around the world to be recycled in the first place. Most rechargeable batteries are also being charged over and over again with electricity that has been created by fossil fuels so it is very difficult to truly help the environment.

Our Autonomous Sanitation Robot contains motors, batteries, and rechargeable batteries. Our robot does not output any emissions directly regarding nitrogen dioxide. The environmental concerns that can come from our project are related to properly recycling the batteries and motors after the assignment is complete. Motors can be recycled easier than batteries. The electrical wiring and circuit boards used throughout the robot are implemented correctly so it doesn’t cause any problems. The circuits and wires throughout the robot are also addressed by being properly recycled and should not be hazardous to the environment.

Finally, our robot is spraying disinfectant chemicals that can also affect the environment in a negative way. Headlines have been made that the overuse of disinfectants from Covid-19 might be hurting Urban Wildlife. To minimize this we chose a disinfectant liquid that is environmentally conscious.

This project can be important for society based on the idea that it can kill bacteria and virus particles which is important for humans. This can be completed while having a minimal impact on pollution based on the batteries being used and recharged. The IRobot programmable robot chassis also has the ability to run on a low-power mode to run longer on a single charge.

Socially, autonomous robots have a mixed perception on whether they are good or bad for society. They are designed to accomplish tasks humans have made for them and work together to increase our efficiency in our daily lives.

People that view autonomous robots as a problem to the future of our society view the potential of jobs being replaced by robots. While this can be true, the goal for these robots is to make our lives better and to do jobs that are potentially hazardous for humans. For example, a cleaning household robot allows our honor to dedicate their time to something else other than cleaning. This is a positive aspect that should be the goal for any robot that is made in the future. There should definitely be discussions amongst experts to make sure that companies cannot replace everyone once robots become more advanced.

In regards to our autonomous sanitation robot, this robot can be important for jobs that are dangerous to humans. This project has the potential to work in hospitals in areas where humans can get sick. This device can work in tandem with humans cleaning bacteria laden areas.

Our autonomous sanitation robot has no political constraints.

5.6 Ethical, Health, and Safety Constraints

Safety is of the utmost importance when releasing a product into the real-world. No matter what field the product is released in, a product cannot be hurting anyone or anything.

Autonomous robots are relatively new in modern society therefore there aren’t a lot of standards and restrictions regarding them. Most places in the world aren’t using them with humans around them. They are mostly used right now as gimmicks like cleaning robots. They aren’t at the point where they can be a major risk factor for humans in their daily lives.

Autonomous robots must have good electrical components and programming to be safe. They need both of these parts working together to detect obstacles and

avoid getting stuck. They need to be quick, agile, and reactive to their environment.

Specifically, our autonomous sanitation robot is using the iRobot programmable chassis as a way to transport our sanitizer around. This iRobot utilizes IR camera sensors to maneuver around its local areas. This can pose a problem to the safety of any human walking around it. Autonomous robots like ours are smaller than adult humans and could have the potential to be tripped on. The robot has software to quickly turn away from any obstacle in front of it. The chassis frame has bumpers in case it ever does make contact with something, it cannot do much harm. It also moves slowly at a pace that cannot inflict any harm to something not moving.

Our team is also using extra sensors to determine when not to spray our disinfectant mist. This mist is set on a timer and is paused whenever it is deemed to be in an area that it cannot spray. These sensors also turn on a bright LED to alert anyone nearby to be aware of the device. The robot is also noisy so that it cannot sneak up on anyone unexpectedly. We are also considering adding a speaker to the robot to add to the noise alerting whenever in the area of an object or person.

The electrical wiring from the custom PCB board is safety-proofed. Every connection is grounded correctly to prevent anyone from possibly getting shocked. Ethically, the proper wiring of the robot is important for any user to not get hurt. Every cable shall also be securely fastened to the robot, as to not be able to snag onto anything that gets close to it. We also are not cutting any corners when it comes to assembling the components so as to not create an unsafe product.

5.6.1 Manufacturability and Sustainability Constraints

An important constraint is the availability or manufacturability of components that have been finalized for the project. Since Covid-19, there has been a major chip and silicon shortage. Technological goods have increased in price or have simply become unavailable. We have to be aware of the high possibility one of our technological components have gone outside our budget or have sold out.
We plan on confirming the technological components availability before submitting our final paper. We custom order a PCB board which could pose the hardest to get if they are backed up on orders. The chassis also contains a lot of chips, circuits, and wiring that can definitely be affected by the shortage that isn’t looking any better.

For the sustainability of our autonomous sanitation robot, the chassis should be able to last for 2-5 years or around 400 charges. With proper maintenance, it should be able to last longer. If not, the major component that would have to be

replaced is the battery and it is relatively inexpensive. The battery degrades over time with constant use.

The autonomous sanitation robot must remain indoors with flat solid flooring to prevent the wheels and tracks from deteriorating. The temperature inside should also be monitored so that any electrical component does not overheat. Specifically, the batteries throughout the robot have the possibility of getting ruined from overuse. Hotter temperature areas also increase this possibility.

5.6.2 Testing Constraints

The Autonomous Sanitation Robot team all live in different areas in the state of Florida. Because of Covid-19, we have been communicating online only. I anticipate that the use of the Senior Design Lab at UCF could be hard to attend to work on the project based on the possible restrictions in the future still from Covid. If online classes continue into the fall semester, the team could also still be residing elsewhere from campus therefore making in-person building sessions more difficult. Testing and building could be a challenge to overcome that can be addressed based on the current situation of how Florida is controlling the virus.

With the decision to use the color green comes from the bacteria. Some bacteria have evolved with the capability of producing different pigmentations. Some of these bacteria are Agrobacterium aurantiacum, Staphylococcus aureus, Chromobacterium violaceum, Serratia marcescens, Bacillus Spp, Flavobacterium. We have the color sensor sensing green because in the given spectrum of bacteria pigmentation the color green is not very common. This would allow our robot to accurately spray on all bacterias. This is an application that will be very useful for future implementations of this project and maybe adding in an IR camera in tandem with the color sensor will help in achieving the goal of sanitation.

5.7 Existing Products and Relevant Technologies

There are just a handful of Sanitation Robots that are available to the general public for purchase today, and each one has a specific couple of features that limits it to only one environment. The most common feature for these robots would be that it is fully autonomous, in that it can track its environment around it and act accordingly. Some products that are available have very niche markets in that they may just pick up dust or partials, and some that just wipe away messes and stains.

We want to build a fully autonomous robot that sanitizes a whole area. Some of these robots are being produced today but at a large price tag that is not available to the general public. We want to combine the best of both worlds and have it where we can make a product that is available to the general public and has an affordable price tag. With the world still getting past pandemic public

spaces are a bit of a hassle to complexly clean. Gyms, movie theaters, and other wide variety of public gathering spaces need a fast, efficient and reliable sanitization robot to keep these spaces bacteria free.

Below we have listed three of the most relevant technologies that are within already existing projects that we wish to adopt into our very own iRobot system. We want to have the best of each of the following technologies and by doing so have created a system that is both efficient and cost effective.

5.7.1 Roomba Vacuum

One of the features that we would like to explore and have on our iRobot is the ability to have the chassis of the system be able to detect when there is an obstacle in the way of it and then react to its environment appropriately. One piece of technology that does it very well is the Roomba Vacuum cleaner by iRobot. They are able to accurately have ultrasonic sensors that are incorporated into the internals of the chassis that react to walls, people and whatever else it may encounter.

We wanted to adopt this principle in that our robot would have similar sensors on top that can detect where to spray and then have the sensors that control the chassis move the robot to the area without running into any obstacles. We also want to have the principle that the system choose the shortest and quickest possible path to get to the selected target area. As shown in the figure below, iRobot made an excellent diagram where the sensors are located and how they interact with its environment around it.

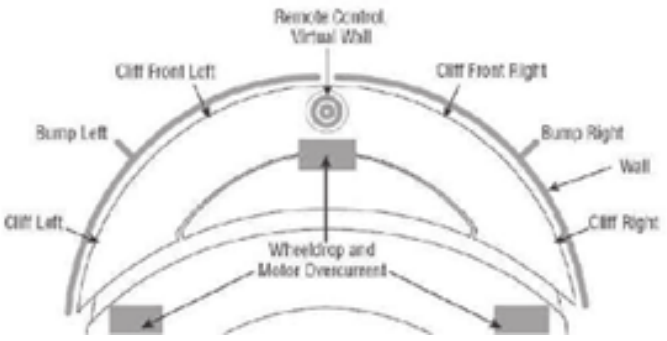


Figure 26: Roomba Sensor Diagram

As you can see above there is a single sensor in the front of the chassis, one on the right and another on the left. Then the rest are just bumpers that push the chassis left and right depending on what obstacle or wall it hits. We want to take bits and pieces of this logic so the sensors just outright avoid all walls and obstacles and go straight to the target without any collision. To do this we would

have multiple ultrasonic sensors at the top of the chassis that can detect when it gets too close to a wall, person or any sort of obstacle. Then when the iRobot system detects that there is a wall or obstacle in the way it then makes the decision to go around or find an alternative path to the given target. This differs from the bumpers on the Roomba because we don't have to constantly bump into things to see where we are, we have the ultrasonic sensors detect when we are getting close to an obstacle and then make the appropriate choice to avoid said obstacle.

5.7.2 Jetbot Mop

Another technology that we would like to adopt is the Jetbot Mop by Samsung. This autonomous robot is just like the Roomba Vacuum but it has sprayers at the bottom and top of the chassis that are then moisturized into cleaning pads that spin into the floor. We want to adopt and use the technology they utilized to automatically spray their solution onto the floor. This is something that we want because it combines the two technologies into one single technology that we are trying to utilize. We want what the Jetbot Mop is able to do, in that it is able to move around its environment with no problem and at the same time can disperse a solution that can clean an area that needs to be scrubbed clean.

This is where our iRobot system would differ, we would want a spray that isn't a liquid and more of an aerosol spray so it can kill viruses and bacteria rather than moving dirt around. As you can see in the figure below, they have a liquid spray that is shot out the nose of the chassis but we would have an attached sprayer at the top of the chassis that can be easily replaced/refilled and would be more of an aerosol can rather than a liquid solution.

Figure 27: Spray Trigger Mechanism

The trigger mechanism would act the same as the one that is used in the Jetbot Mop, but instead of a liquid solution it would be an aerosol can that can be inserted at the top of the chassis. We would also utilize the technology where the

robot can detect when to spray and trigger the solution to be sprayed by using a microcontroller to detect where to spray, then have the iRobot system go there then have the microcontroller launch instructions to fire the trigger mechanism that would spray our solution to the targeted area. The reason why we want to use a spray rather than a liquid is because we want to disinfect rather than mop or clean where the targeted area is. The Jetbot Mop works so well in a housing setting for floors but it doesn't work as well for workstations and desks. This is why we would keep the same technology that allows for the solution in the Jetbot Mop to be transferred and sprayed but switch out the solution for a spray that would collect on more surface area.

5.7.3 CLOi the Autonomous UV-C Robot

Finally, another technology that has a similar mission to the product that we are trying to create is the CLOi Autonomous UV-C Robot by LG. This is a commercial robot that is fully autonomous and is expected to set the new standard for commercial environments such as gyms, offices and movie theaters. This is a robot with similar feats to the previous two, in that it can easily move its way around chairs, tables, and other objects so it can clean an area. But rather than just clean an area it also disinfects and irradiates touchable surfaces by using UV-C lights. This would be a similar goal to what we as a group would like to accomplish in that we would want to be able to have it where we can clean and disinfect an area. But since most of the autonomous robots just clean and don’t disinfect we have chosen to use a spray that would do both. The only idea that we would adopt from this level of technology is its ability to fully disinfect a workstation.

The idea of the UV lights would be a neat technology to adopt into our own but it could come at the cost of using more materials and jacking up the price so it would be unaffordable to the average consumer. The CLOi UV-C Robot comes out to a 1500-dollar price tag and is only available to a select few. We as a group want to avoid this and make it so the market price of our iRobot is cheaper but at the same time accomplish the same goal as the Robot by LG.

The disinfectant technology is something that we want to adopt into our own model by using and testing out what sprays can be utilized to clean a workstation. When exposing the UV-C lights to an area that is infected will neutralize 99.99% of germs, bacteria and viruses. This is where the CLOi Robot shines in that it can be fully autonomous and can fully disinfect an area. While UV-C lights would be the perfect solution to our problem, the application and cost of keeping the lighting running for a long period of time to fully cleanse an area would be financially draining. Also, the amount of time that is needed for the robot to be in the area would also be a burden as the UV-C light would need to be there for a long time in order to fully disinfect an area, this can be shown in *Figure 28*. But this is where, as a group, we would be able to shine in that we

could have a spray that can accomplish the same task as the CLOi Robot but it would take half as long to accomplish.

One perk that we wish to have the same would be the CLOi Robots ability to fully disinfect high to reach and high traffic areas such as desks and chairs. We as a group want to make it cost effective and can't implore tactics such as buying large LEDs. We also want our iRobot system to work as quickly and as efficiently as possible without sacrificing one over the other. This is why we as a group have implored the solution of using a spray canister rather than using LEDS or a liquid solution. With the added bonus that comes with a spray we would be able to reach hard to clean spaces that a robot couldn't normally do. The large area of effect that a disinfectant spray would have would solve all our problems for it being cheap, quick and efficient.

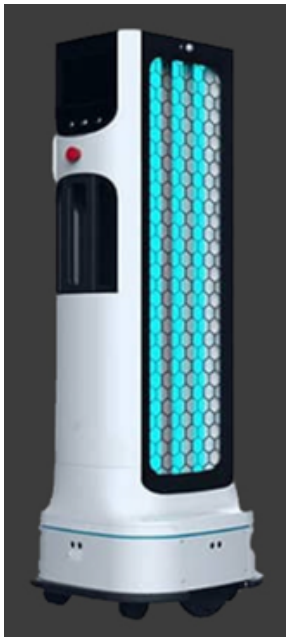


Figure 28: CLOi UV-C Robot

The main goal for researching the UV-C Robot, as pictured above, was to get a glimpse into how to disinfect an area and when to know how to disinfect an area. The UV-C Robot is on a timer where at a certain time each day it launches and begins to clean all programmed routes. This was another principle that we wanted to adopt into the building and research of our model is the when and how to detect an unsanitary workstation. This can easily be solved by launching the iRobot system on a timer that can efficiently spray an area every so often and does so every day on the hour.

With all these technologies combined we would have the perfect system for our group's fully autonomous sanitization robot. It would have the pathfinding and environment detection of the Roomba, the trigger system and solution system of

the Jetbot Mop and the sanitation power of the UV-C Robot. All of these relevant technologies would be working together to create a product that can both be very useful in battling illnesses and would be financially pleasing to the general public.

6.0 Project Hardware and Software Design Details

This section looks to give all our design plans and for both our hardware as well as our software designs. In the previous sections we discussed the factors and standards that we used in selecting our components for our project. Now we discuss how these components helped us design and complete our project. This section also takes 3 main subsystems of our project and breaks down how each system works and their respective tasks as well as how it came together. This section contains flowcharts, schematics, and testing of our parts and overall design.

6.1 Initial Design Architectures and Related Diagrams

Our project uses an iRobot two create paired with a raspberry pi 3 and a spray mechanism controlled by our microcontroller mounted on top of the iRobot. The big picture idea of the project is that we used the iRobot programmable as our chassis and we mounted our spray mechanism on top of the iRobot. We mounted our components and spray mechanism directly on the iRobot with a platform that housed all our components and the spraying mechanism. For reasons discussed in earlier sections the iRobot was the best option for what we want to do. The main aspect of our project and what we show through our design and deliverables is having a working autonomous spray system that can accurately detect when to spray and when not to spray. This was done through our 3 major subsystems: the iRobot, the microcontroller, and the spray mechanism.

First up is the iRobot subsystem, which contains the iRobot, Raspberry Pi, and a TCS3200 Color Sensor. The iRobot has a serial cable that can communicate with a microcontroller or a computer, and in this case, it was used to communicate with the raspberry Pi. The Raspberry Pi was the brains of this entire project, as it was responsible for communicating with iRobot and microcontroller. The Raspberry Pi along with the TCS3200 color sensor are also responsible for detecting areas that need to be sprayed and areas that do not be sprayed. This information can then be relayed to the next subsystem which is the microcontroller.

The second subsystem is the microcontroller along with the ultrasonic sensors. If the Raspberry Pi is the brains that the microcontroller can be seen as the body. It takes the commands from the raspberry Pi and responds by activating the spray mechanism through connection points. In addition, the microcontroller has some ultrasonic sensors connected to it and these sensors are used for the spray mechanism and prevent any spraying from occurring when a human is at a certain distance from the spray. This is true even if a spray command is given by the Raspberry Pi. In all this subsystem serves to follow the commands of the raspberry Pi and send all the information and commands given to the third and final subsystem: the spray mechanism

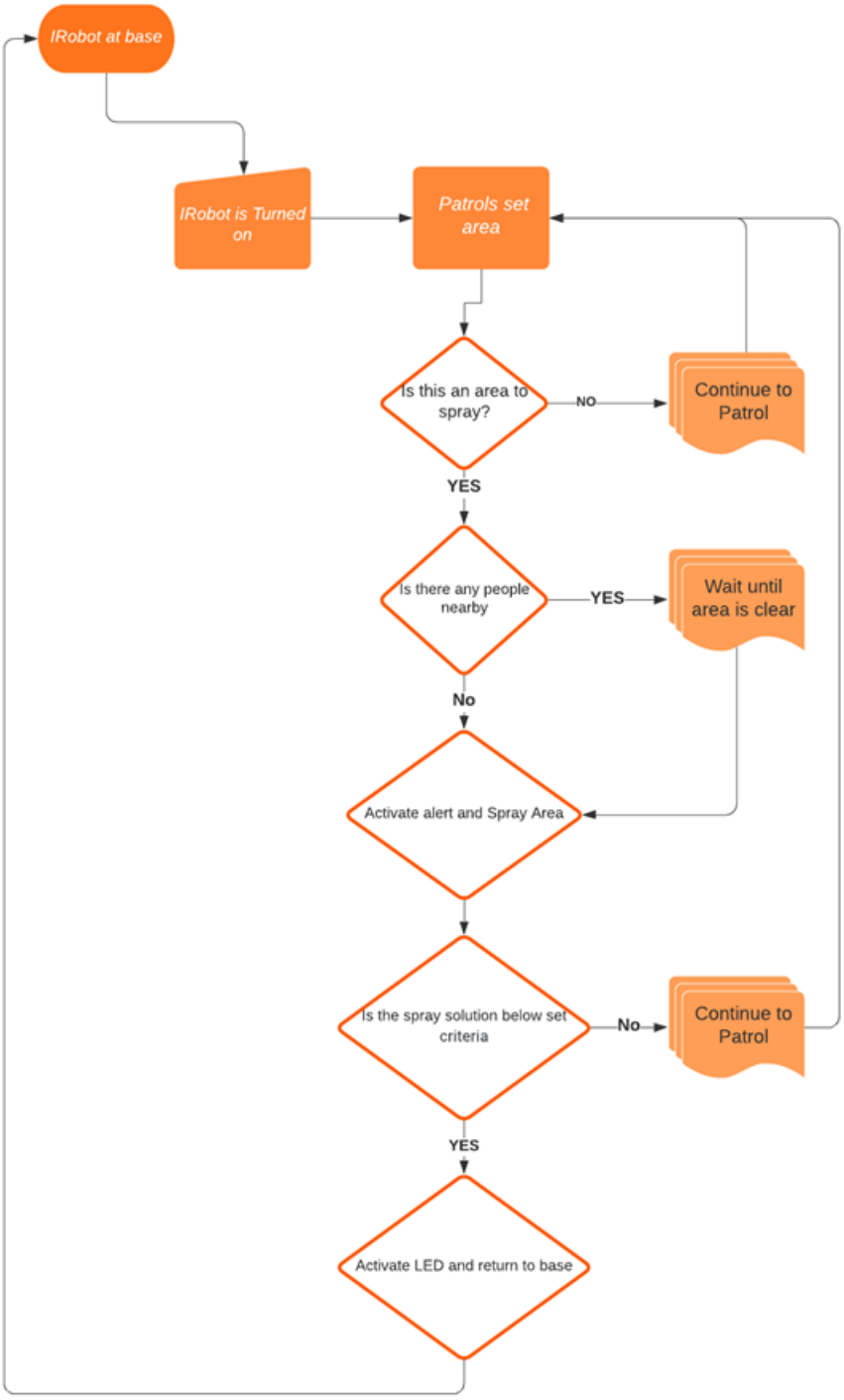


Figure 29: Project Design Flowchart

Lastly the Spray mechanism is our last subsystem that works with our Liquid level sensor and speaker. The Spray mechanism and liquid sensor work together

to have the sprayer spray the solution and the sensor to monitor the levels until it reaches a set level. The speaker is used to alert users of the robots' activities. The speaker goes off when it is about to spray.

The Figure above shows the basic logic followed by the autonomous robot as it begins its tasking all the way to the end of its tasking. By detailing the logic through this flowchart, it helps understand the different subsystems roles needed at each junction of the robot's task. The following sections focus on the different subsystems and their key associating documentation and schematics

6.2 First Subsystem, Breadboard Test and Schematics

The first subsystem is the iRobot, TCS 3200 Color Sensor and the Raspberry Pi 3 Model B+. Looking at the iRobot and taking a deeper look into it's design we see how it works and how it interacts with the Color sensor and the Raspberry Pi.

The iRobot software is an open interface which is made for controlling and manipulating the robot's behavior. This ability to edit and utilize many of the sensors is why this is a solid development kit and a great chassis. The Open interface software allows users to manipulate and read all the sensors on the iRobot. Using the open interface software, the Raspberry Pi reads and controls the sensors on the iRobot.

To access the Open Interface, it is stated to use the Connection Cable and plugging in into a high-powered processor that can generate the serial commands and the Raspberry Pi is a very powerful computer and works perfect for this. This is done by connecting the iRobot Create USB cable and plugging it into the Raspberry Pi in one of its USB 2.0 Ports. The connector is plugged from the Raspberry Pi to the Mini-Din connector on the iRobot. This connector provides the ability for the Raspberry Pi and the iRobot the ability to connect and communicate at a TTL of 0-5V levels. The connector also gives the Raspberry Pi access to the battery of the iRobot which can be monitored to see if levels are low and return to base is necessary and is also used to operate and power some of the available open interface applications

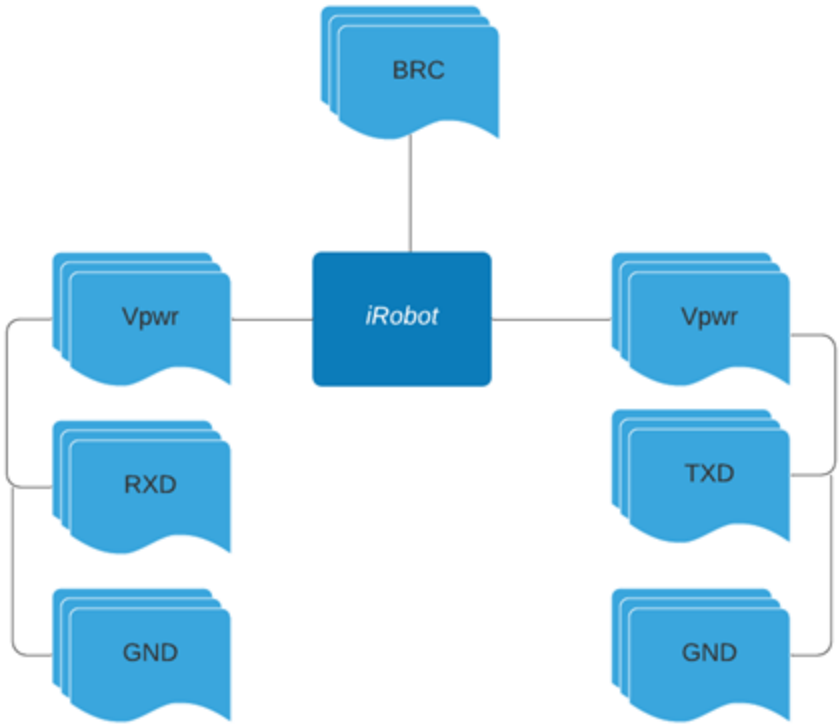


Figure 30: iRobot System Diagram

The iRobot has seven pins associated with its design. The iRobot does not have many pins by design as most of the modification and readings come from the use of the open interface. However, these pins are crucial in understanding how the iRobot operates and helped in troubleshooting problems when began to implement and modify this. Pins 1 and 2 are standard unregulated power and specifically the Roomba's battery. Pins 6 and 7 have a very similar functionality with these pins dedicated to the ground of the Roomba's battery. Pins 3,4, and 5 are very interesting and key applications associated with it. Pin 5 is the port that deals with the Baud rate, and the iRobot has a default baud rate at 115200, but if a processor cannot support this level there is a way to force a switch to 19200. Lastly the last two are associated with the TTL level communication set at 0 – 5 volts with 3 being RXD which is associated to the serial input and 4 being TXD which is the serial output

The RXD and TXD uses 0 – 5 Volt logic which is different from the serial port used. A solution to this is to use a level shifting device to match the differing logic levels. Fortunately, the iRobot Create USB cable has level shifting capability thus no need to construct a voltage level shifter.

The next part of this subsystem is the combination of the Raspberry Pi 3 Model B+ and the TCS3200 Color Sensor. The idea and logic behind using this combination of our project also speaks to the overall versatility of this project. Due to this project and its nature, there were many limitations due to time and budget constraints. We wanted a way to show that when a signal is registered that our spray mechanism reacts in accordance with that signal. So, in this circumstance we used the color sensor to read a specified red, green or blue with one of the three being the signal that has our speaker beep and the other two should not elicit a response from our robot. This is something that can be improved on in future implementations of this as you can add a camera and now you can use image processing techniques to accurately detect areas that need to be sprayed and avoid spraying in areas that are not needed and potentially harmful to people and the environment of that area. The design of the color sensor and raspberry Pi model was because as a group of 3 EE and on CPE it would be extremely hard to have one person try to create a whole image processing software to be used. We had decided that the use of colors would sufficiently show the key aspect of our project and specifically the spray mechanism.

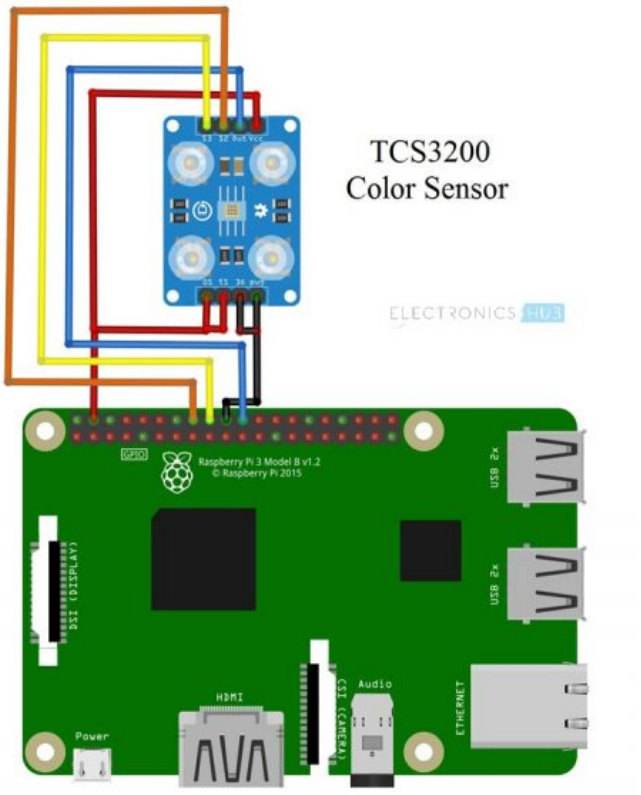


Figure 31: Raspberry Pi and Color Sensor Connection

Now for the actual setup and understanding of how the Pi and the Colors sensor interacts with each other. First off, the Color sensor has an 8x8 array of

photodiodes which is covered with either a red, green, or blue filter which ultimately detects what kind of color that the sensor reads. The setup is then placed by utilizing the general-purpose input/output of the raspberry Pi. There are 40 of these pins that can be utilized in any way that the user specifies, and the Color sensor is connected to these pins to begin working with the board. The wiring all depends on which S0, S1, S2, S3 that you want to hook up. These switches determine the output frequency and which photodiode type is reflected which essentially means which color is read by the sensor. So, these switches control how well the color sensor is able to read the color and what color it is reading.

The raspberry pi website has a blog reporting on using one of its model's and how to connect it with the TCS3200. The article also provides a video illustrating how the sensor works with the raspberry pi module. Figure shows the TCS3200 being connected to the Raspberry Pi Model though it's general purpose input/output pins

Now for this subsystem with all the components talked about a big issue that arose is that of power. For the iRobot it comes with a charging dock station that is plugged in through a wall outlet giving 120 V, thus making it the one thing that does not need to be powered through our own design. The Raspberry Pi and color sensor combination needs 5 volts with a current draw of 2.5 A. The microcontroller is connected to a PCB board that supplies the power necessary to operate. So, in terms of power this subsystem did not need much besides that of the Pi module.

So, in conclusion these three items are combined to create our first subsystem. The biggest thing was to try to implement and have everything interact with one another without overstepping and affecting the work of each of the individual components. The next sections focus on explaining the other two Subsystems while also making note and mention of the software design that go into our project.

6.3 Sprayer Sub-System

The first and most important part of the sprayer system is the sprayer itself. As we went over in section 4.5, the sprayer that was required had to be motorized to simplify the system. With that in mind the PentraTools Automatic Battery sprayer was chosen as the ideal sprayer for this application. The most important part about this sprayer is that it is motorized, and the activation method is a trigger switch. Utilizing the trigger switch we modified the spray head and removed the trigger altogether and only used the switch contacts to initialize the spray head.

The easiest way to initialize the spray head is to connect wires to each of the contacts to a relay. This relay is connected to the microcontroller and awaits a signal from the microcontroller to turn on the sprayer. When the relay receives its

signal, it completes the circuit for the spray head thus initializing the sprayer. To turn off the spray the same is done. A signal is sent to the relay, and this causes the circuit to open causing the sprayer to turn off.

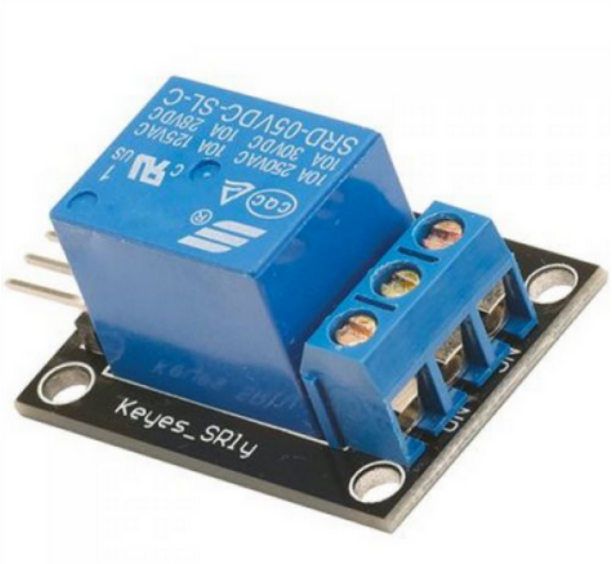


Figure 32: Example of Relay to be used

Unfortunately, the manufacturer has not provided a datasheet or schematics for this part so further testing must be done to ensure the system works flawlessly.

There is a sensor mounted within the reservoir to gauge when it is running low on solution. When this sensor flags the microcontroller, this initiates the commands that it no longer initializes the relay for the sprayer. This is the first command from the microcontroller to take over the sprayer. The microcontroller uses the relay to initiate the sprayer whenever the timed interval of the sprayer is started. Only then the microcontroller sends the signal to the relay to start the sprayer. The microcontroller also tells the relay when it should stop spraying too. This is a crucial step in the sprayer system.

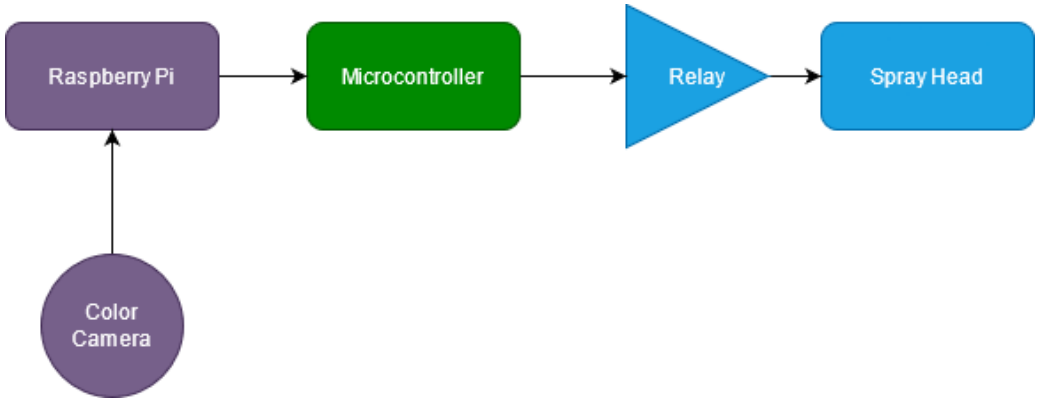


Figure 33: Block Diagram of Spray System

If the microcontroller does not send a command to the relay, then the sprayer continues to spray either until it runs low on solution or until it runs out of battery. In which case that's why we couldn't have the sprayer hardwired because then the entire robot dies. However with the long battery life of the sprayer we were able to connect it to our relay which was connected to our PCB board.

6.3.1 Sprayer Mounting

The mounting position of the sprayer is almost more important than the sprayer itself. With the iRobot Create platform, we used a mountable surface to place all our additional devices. In terms of the sprayer and reservoir, the reservoir was mounted as far back and away from any sensitive electronics as possible to avoid any contact with the cleaning solution. The tube was run around all the electronics to avoid any accidental spills. The spray head was mounted onto a platform that enables it to have the most optimal angle of attack when spraying down a surface. The reservoir and the spray head are firmly mounted to the chassis to avoid any unwanted movement, but the spray head has a varying degree of adjustment to find the perfect angle. The sprayer head is sat upon a wedge block. This allowed for an optimal angle for sprayed and allows any dripping to be caught be the block.

6.4 Alert Sub-System

This is going to be a system used to alert the user the solution is running low, and to notify the user of the robot's presence. This was done using speakers to notify the user of any actions that need to be taken.

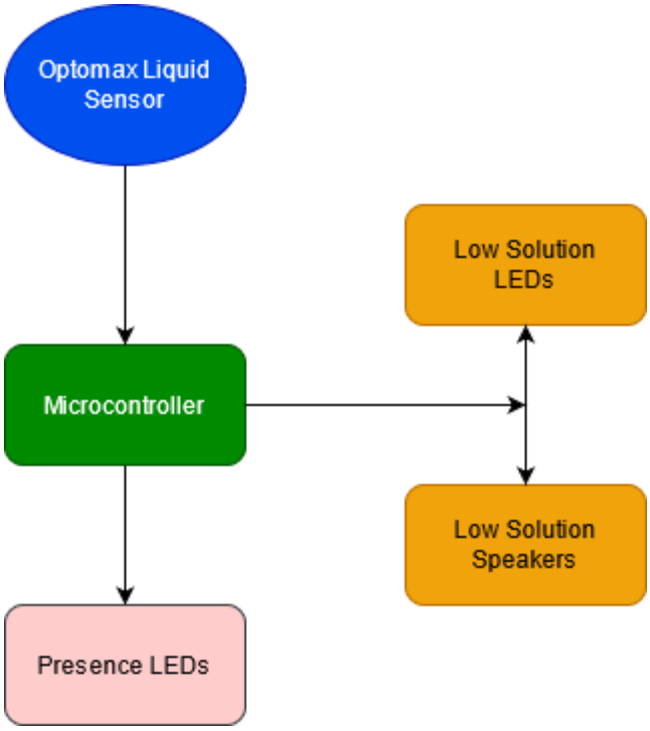


Figure 34: Block Diagram of Alert System

There is also a JBL speaker that plays sounds only when deemed necessary. This is specifically done during any sprayer instance that occurs and when the liquid level sensor detects no more solution. The two sounds for these cases differ. This would enable an almost stereo soundscape and should be audible to the user from approximately 7 feet away.

6.4.1 LED Alerts

With the Led alerts, most of the LEDs were to be mounted at the center of the robot to draw attention to itself so that the users would not accidentally bump into it. When spraying some of the lights were to flash to show progress being made. This was to be done utilizing the io pins on the microcontroller. Since these are single color LEDs they only need to be powered. Depending on how much voltage and at which frequency we apply the power to the LEDs we can obtain a strobing action and solid color. The microcontroller was to be programmed to flash some of the LEDs when it sends the command to the relay for the sprayer.

A couple of the LEDs were to be mounted closer to the reservoir, these were to be used to notify the user that the solution is running low. To do this the Optomax Digital Liquid sensor was used to sense if there is a solution above a certain level. Once the solution drops below the level this triggers the sensor to alert the microcontroller which sends the command to the special LEDs next to the reservoir.

All the LED alerts are controlled by the microcontroller, this enable us to use simplistic commands since most of these LEDs are only being used to give alerts from a sensor. Utilizing the sensors and LEDs we can notify the user of almost every state that the robot is in and be able to visually alert the user of when the robot needs servicing such as a recharge or solution refill.

SD2 UPDATE: We decided that this LED would not work as well as we had anticipated given the platform we created would cover the LED. This would essentially make the LED useless in the nature it was designed for. Along with that we decided that a speaker system is more practical.

6.4.2 Speaker Alerts

Utilizing the speaker that is mounted on top of the robo with the platform, the robot plays tones to notify the user of when it is about to spray. The speaker is wired to the raspberry pi. The microcontroller receives a signal from the sensors such as the liquid sensor that would then cause the microcontroller to send a signal to the raspberry pi to play a specified sound given which sensor is sending the data.

The audio cable from the speaker is run from the speaker itself and to the audio jack on the raspberry pi. The raspberry pi has significant onboard storage so we were able to store any types of sounds that we needed

The raspberry pi is programmed in a way that when it receives a certain signal from the microcontroller it plays the corresponding sounds for that action such as when the reservoir is running low it plays a chime.

This system needed to be able to interact with the raspberry pi flawlessly since the raspberry pi was going to be the device to store and play the sounds that we choose. The microcontroller is only there to tell the raspberry pi when it needs to play the sounds by constantly getting readings from the sensors onboard the robot.

6.4.3 Alert Sub-System Summary

The purpose of the alert system is to notify the user of any action needed to be done around the robot or to the robot. These actions include: refilling the solution, charging, and approaching too close to the robot during cleaning. The entire system started off trying to encompass a series of LEDs and two 3W speakers that obtain the attention of the user and depending on the LEDs lit-up and chimes heard the users action are needed. Through some changes in SD2 we decided just to go with the speaker system on its own. The user is able to hear the robot alert so that it draws as much attention as possible without being a hindrance or obnoxious in any way.

6.5 Software Design

We are using a Raspberry Pi to communicate between our custom PCB board and our chassis. Our team decided to use either Python, Java, or C to write the logic for the features we want to implement. The IRobot Create 2 Programmable Robot has built in IR sensors that allow for object detection and obstacle evasion. There is built in logic to try and maneuver out of stuck positions as well. Advantages and disadvantages for each language are discussed below

Our first big implementation of logic was for when the disinfectant liquid inside the container is low at a specific percentage. A sensor triggers to alert the chassis to create a sound and shut down the robot. The chassis uses a bluetooth connection with logic that allows it to return back to its base if prompted to. Our chassis uses a random autonomous algorithm. It covers territory randomly and could possibly run over the same spots periodically.

Our next feature to be implemented was the sensors to communicate with the IRobot that it is near an object, human, or area so that an audio queue is triggered to alert anyone to be aware of the robot. The sanitation dispenser is on a timer to release fluid every 15 seconds. The sprayer is using the relay on its timed intervals to spray while alerting users that falls into the ultrasonic range

We used a color sensor to detect areas that let the robot know when to alert before it sprays. Our logic detects within a certain amount of feet and a certain color to alert the users in that area. An example is our robot used in a hospital, certain wings of the hospital could have certain color tape in areas and distance by users to determine when to alert of the robots presence and its sprayed function.

Before determining that we are using Agile Scrum Methodology, we looked at different software development systems. Looking at the Waterfall model, it is linear in nature while Agile Scrum is circular because it constantly goes back to the beginning updating the tasks that need to be completed.Waterfall is basically what a team would do if they didn’t think of a development system to follow. Waterfall plans in the beginning and then goes through each feature as a team without roles. The work is completed without a real solid plan and keeps going until everything is complete. At the end, bug fixes are addressed instead of in the middle. After bug fixes, testing is started unlike the testing that happens in the middle of Agile Scrum. Waterfall's main problem is that it isn’t clear exactly the amount of progress that is being made. There aren't any pre-planned week-by-week goals to know how far into the software side of the project we are in.

While Waterfall is more relaxed, Agile Scrum can be a bit tedious. The entire team has to be dedicated to completing their tasks for each sprint. If one member doesn’t complete what they were assigned to do for that sprint, it could start to

grate on the nerves of everyone else. It is a really hands on way of making sure the project is completed. Waterfall could be good for our team if everyone has jobs and other places to be during the week. As long as the tasks are completed by each team member, there really isn't a problem to be brought up. There would not be as much of a commitment every single day doing daily or weekly meetings just for the coding side while we still had to do everything on the physical side building. Without a framework at all, it can be a potential disaster.

Our team ended up using Agile Scrum Methodology to complete all the necessary coding features for our chassis, Raspberry PI, and microcontroller. Scrum is an organizational tool that can help groups complete projects easier. Without a good system, progression towards the goal of completing all the tasks could end up being a problem. Agile Scrum forces the group to complete their responsibilities.

We started at the beginning of the fall semester with a product backlog. This holds everything we hoped to accomplish when it comes to the software side. It was edited throughout the completion of the assignment. We assigned the product owner to Rishi Patel and he was responsible for the product backlog.

The sole computer engineer, Allan McCormick, was the Scrum Master. He was responsible for keeping everyone on task and dulling out responsibilities that need to be assigned. We did sprint planning after the completion of our product backlog. Sprint planning is right before the actual sprint hence the name. In planning, the team came together for a meeting and discussed whether the next objective is feasible for the week or if it requires more time.

Sprints contain the actual completion of the tasks assigned from the planning. It was a week for easier tasks and two weeks for tougher tasks. We were doing daily scrum meetings as well to make sure everyone is doing ok with their tasks. This was a really short meeting just to discuss any problems or concerns anyone might have. These daily meetings were only open during sprint weeks.

We then commence with sprint reviews at the end of each sprint. This is where our team made sure that what we have completed has been done correctly. We also voiced any concerns we had with the progress we were making in these reviews. The product owner, Rishi, went back and edited the product backlog checking off what that sprint completed. At the end of this sprint review, we had a sprint retrospective and talked about what they liked or disliked from this organizational scheme. We adapted and changed anything that we thought could make the process smoother.

This system is a very easy to understand process that gets the job done. It isn't hard to buy into so it can be very flexible to any member on the team. There was a sprint for each feature, therefore, we had about 5 sprints in the fall. Each sprint

was at different times depending on the progress being made physically building some of the components.

We looked at different IDE's before determining which one we to go with. An IDE is a specific personal preference that is a subcomponent of the whole completion of the project. Visual Studio Code is a very good system that can download different language extensions. We thought we could go with an IDE like Eclipse when we are working with just Java. PyCharm is a good IDE for Python. Dev-C++ is a good IDE for C. Splitting up each language to different IDEs would be a little chaotic if a bug occurs in one. It is more convenient to keep it all in one IDE and shift over easily to different project folders for each component. All of the IDE's mentioned are free so it works well for staying within our budget.

As seen in the graphic below, programming languages can be chosen for a project based on the subjective feelings of the team. Python has eclipsed Java recently because of how user-friendly the language is especially to beginners. We have learned in most of our classes about C and Java. Specifically for C, we have learned it more than others based on how close it can be to the hardware level. This language connects to the others as a building block.

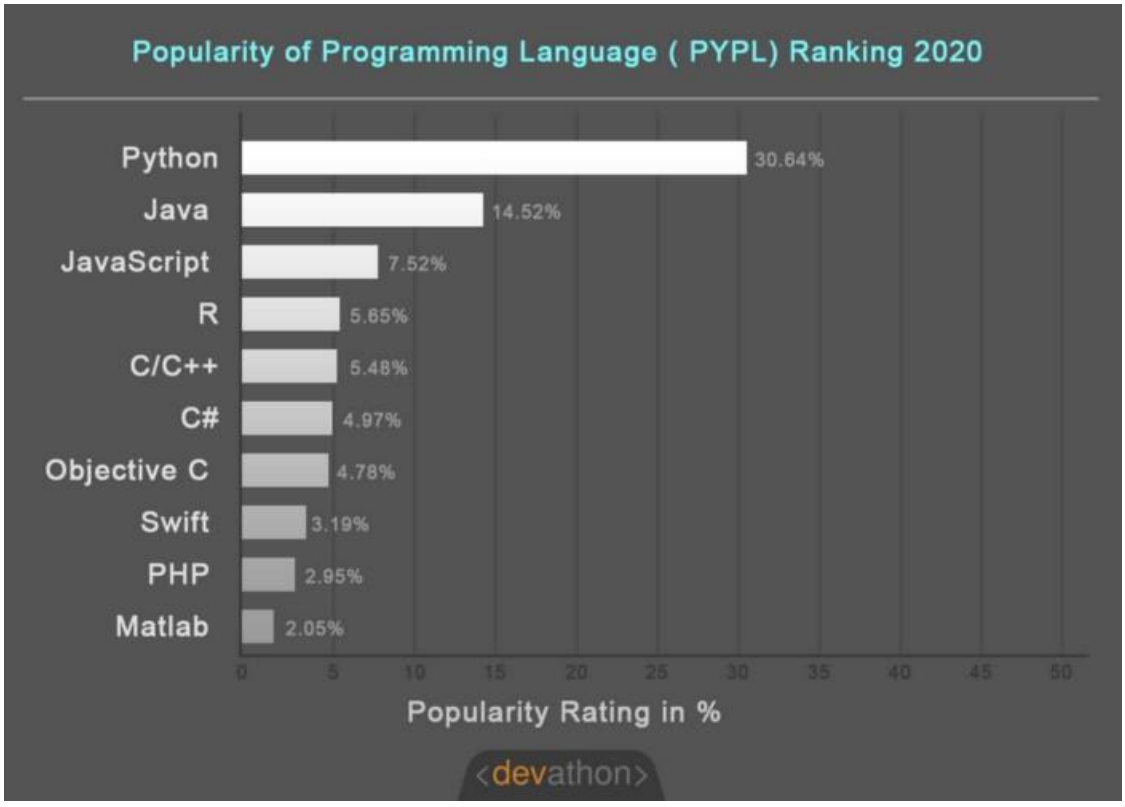


Figure 35: Programming languages based on popularity

Specifically for Python, it is a high level object-oriented language that is known to be intuitive and easy to understand if anyone has started on another language before looking at Python. Python was released in 1991 and came about by fixing the flaws of the language ABC. Python has been through many iterations and updates. It is currently at Python 3 while Python 2 seems to still be the most used. For our project, we looked at Python 3 as it has a lot of good tutorials online for it. Python has become very popular recently in the internet age, therefore, the language has a lot of resources and support if anyone has a problem with what they are doing in Python.
The three biggest benefits of Python are:

- Reliability and Stability
- Easy to understand syntax
- Extensive amounts of tutorials

Java is also a high level object-oriented Language that combined what was best from other languages from before and made it better. Java was created in the 1990s and has been through many iterations and updates. It actually has not changed that drastically from its initial JDK 1. Java is good for large classes. We needed an easy to understand language to be able to communicate between the different technologies we were trying to mesh together. Java was thought to be very good for our team because we were mostly electrical engineers therefore, we would benefit from the Java virtual machine taking care of allocating any kind of memory we are using. Other languages don't give us that luxury and it could have been a bit confusing to learn how to allocate memory for complicated tasks especially when we were working on tight deadlines and physical technologies we had to get working. The Java Virtual Machine works by interpreting each line of code allowing the code to be used on any other machine. It works within the JDK and JRE.

C is a high level language that is not object-oriented. It is a procedural language. C was created in the 1970s and became extremely popular among any company using it for hardware. The official standard of C was then created a few years later by ANSI. C is good for a combination of both low and high level applications. It is a benefit for microcontrollers because it works well with low level aspects. The low level side for the microcontrollers is easier to understand than most languages and someone could go into that part without really knowing how to code well. C focuses on machine inputs instead of libraries so it helped our Autonomous Sanitation Robots microcontrollers function easier than working with other languages that we might struggle on. We as a group have also taken classes working with embedded systems practicing with the language of C so we feel more comfortable with it. The differences between procedural C and the next step of object-oriented C can be seen below in the table.

Procedural Programming	Object-oriented Programming
It splits up the code into smaller parts calling them functions; however, it is not easy to create new functions.	It splits up the code into small parts known as objects so that anything new added is simple to be added.
It isn't that secure because data cannot be hidden at all	It is secure because its data is protected
It is older lower level languages that can still be used today but less frequently for complicated tasks.	All modern high level languages are Object-oriented

Table 14: Procedural VS Object-Oriented Programming

C++ could be seen as a child of C. It was made in the 1980s as an object-oriented language. It is really fast because it is close to being a low level language even though it is a high level language. It is good for working with hardware but also for software to develop products.C# is also another child of C. It was created by Microsoft. C# is not that old compared to some of the other languages and while Python is easier to learn, C# can be compared to it because it has similar functionalities however, C# can be faster at the end during its runtime.

Coding Language	Advantages	Disadvantages
Python	Very easy to learn syntax and how it works.	It is slower than C and C++ when it comes to runtime.
C	Excellent for hardware because of how close it is to assembly language.	It isn't as easy to error check as other languages
C++	It is faster and easier to use with hardware than most languages.	The heavy use of pointers is a tedious and complicated subject to learn as a beginner,
Java	It is intuitive to understand and set up on a computer.	It has a heavy overhead of memory usage.
C#	Fast at runtime compared to other languages.	It isn't that easy to learn the syntax.

Table 15: Popular Coding Languages Pros and Cons

6.5.1 Summary of Software Design

We used Microsoft Visual Basic Studio to implement our software. This IDE was used specifically for C that was loaded from the computer's USB ports to the serial cable that plugs into the Roomba. The Raspberry PI was also installed inside the Roomba that was coded to instruct it how it moves. This mini computer communicated with the disinfectant payload housed above the Roomba. The Raspberry Pi was talking to the custom PCB board.

The Raspberry Pi was responsible for using all the sensors together to create this cohesive system. In C, the Raspberry Pi was instructing the robot to activate the speaker if the disinfectant tank is low. The disinfectant tank has a sensor being monitored by the Raspberry Pi. The Raspberry Pi works in conjunction with the sensor that detects colors and with the ultrasonic sensor. These values are addressed by alerting the switches and pins on the PCB board up above to start utilizing the sprayer capabilities. The Raspberry Pi was toggling the timer of the sprayer to off and whenever the ultrasonic sensor detects something right in front of it or if the particular color detected is activated the alert sub subsystem.

The language of choice was determined to be C based on how much experience we had using the language compared to others. We would have to learn Python before experimenting with our autonomous robot and even though Python is known to be very easy to learn, we felt more comfortable with C. The flowchart of logic can be seen in the graphic below.

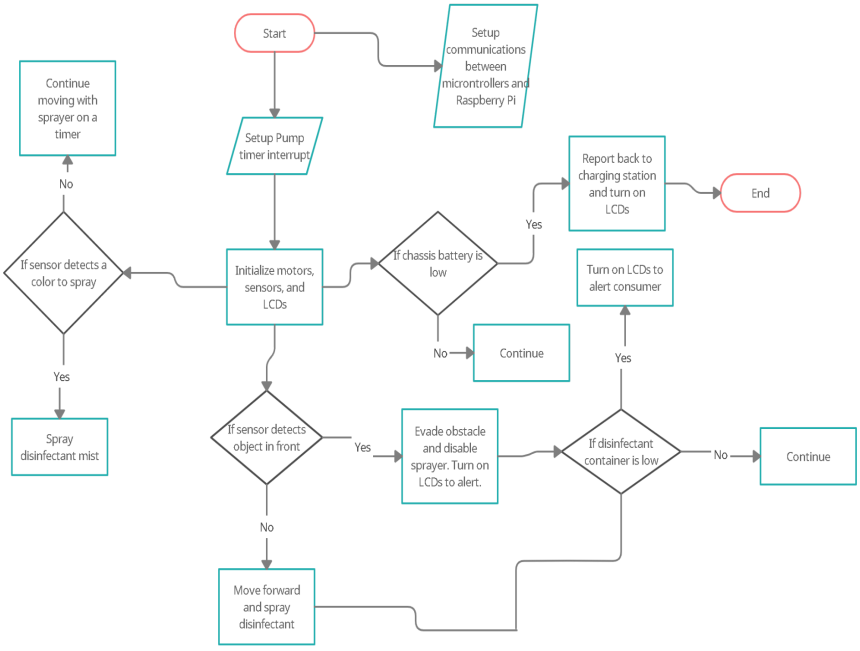


Figure 36: Software Design Flowchart

6.6 Final Summary of Design

Throughout this section we discussed the three main subsystems: the iRobot, the sprayed and the alert subsystems. Each one of these subsystems works together and communicates with each other to make our project work completely. If one failed it could lead to failures seen all throughout the robot. Our design was heavily reliant on how well we integrated our subsystems along with our software design. We had many changes in our designs in hopes to achieve the best possible outcome we can reach

SD2 Update: Many of our subsystems had to be changed due to our changing of our deliverables. The biggest one is instead of having our color sensor detecting for potential color we had to work in tandem with the ultrasonic system to work and activate our alert subsystem. This came due to a lack of integration and time with our PCB board due to vendor lead times.

Our Raspberry Pi also went from having the speaker, irobot and color sensor to just the irobot and speaker. Our color sensor went to communicate with the MSP430 chip along with the ultrasonic sensor. This was a design decision as we saw that it would work alot better to have the color and ultrasonic in tandem with the chip that is actually controlling the spraying. If the pi and msp430 try to talk during timed decisions like that there can be a mistiming in the sensors.

The next section focuses on the integration of the subsystems we have mentioned. We want to expand on how each of the three work in tandem with one another and how if one fails to meet it’s deliverables how it could have greatly impacted each. The next section also greatly focuses and transitions into PCB information, design and vendor information.

7.0 System Integrating

With regards to testing we were to be able to show that our system is capable of correctly executing each of the deliverables accurately. If one of the deliverables did not work then it is imperative that we, as a group, determine what the problem is and what are the possible solutions to the problem. So, in order for us to have a clear and concise system then integrating and testing must be established very thoroughly. Every single component of the system must be individually tested and then tested in unison to see if they can work together accurately. This includes all of the microcontrollers, raspberry pi, sensors, wires and PCB, all of these units have to be able to be tested for accuracy. After both large and small components were tested then the subsystems had to be tested in order to establish if the design was implemented and created properly. Finally, the overall system was tested and it would lead into the end products that we as a group created. The integration of the system is all about designing correctly and quality control rather than testing. When connecting each of the subsystems brings the question of which communication protocol we used, how power was distributed and the logic behind the system. These topics and more would be more looked at in the subsections down below.

7.1 System Integration

Integration in a system is how you connect data-lines and power lines across each of the subsystems through the electrical components. In the case of software, the microcontrollers of our system are able to get information and transfer that information across all of the subsystems. Below we look at more specific cases of how integration can be seen in our system.

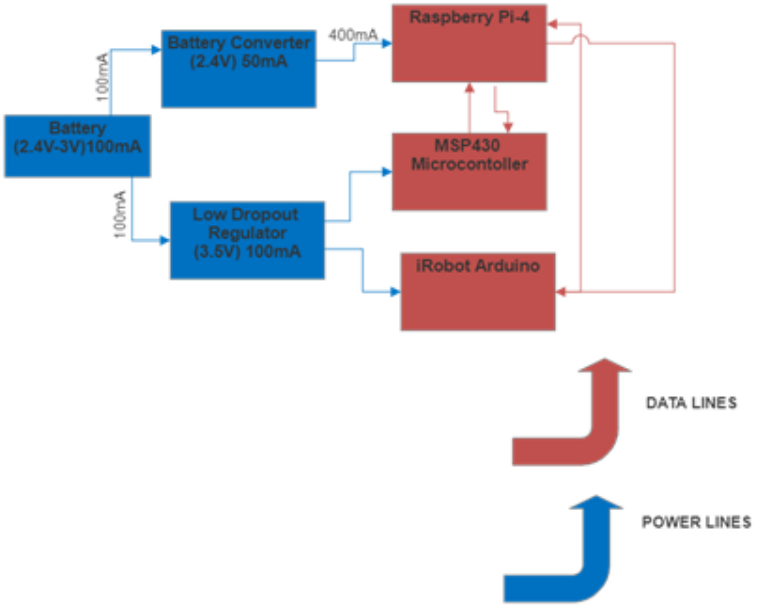


Figure 37: Block Diagram of iRobot System

In the figure shown above we can see how the Raspberry Pi-4 is what all the data-lines lead to in the end and how both the Arduino and the MSP430 communicate with the Raspberry Pi first to get their instructions. This is how the system would be integrated and how all the parts communicate with one another. The integration can be broken down to show that the data lines lead into one of the three subsystems that make up the whole iRobot system. The block diagram above doesn't show all the small components that make up the remainder of the subsystem but rather the brains that control those components. This is because we wanted to test to see if the brains of each subsystem could communicate to the other brains of the other systems. For example, the Raspberry Pi subsystem, as shown in the diagram above, needs to talk to the Arduino and vice versa, this was done through a data path that passes through each microcontroller. We made sure that we can integrate these types of subsystems correctly or else the entire system could fail and not be able to transfer data correctly to one another. This could have been one of the problems that we would have had to iron out because we essentially have three microcontrollers and three subsystems that all need to deliver instructions to one another in a fast and efficient way so the whole system as a whole can function. When it comes to the printed circuit board, we as a group have to design our own rather than getting a premade one off the internet. A custom printed circuit board was another issue that we faced, as we needed to determine which systems need to be attached to the board and how many modifications need to be put onto the board itself. Even though using a board that we as a group didn't have to design would be arguably easier it would be very hard to modify and make it fit our very specific integration process. There was room for error when designing the PCB and we ultimately needed to run a lot of trial and error testing when going through the design process of our PCB.

7.2 Sub-System Integration

There are multiple systems that are being used in our iRobot system and the integration of each of the systems is integral and cannot be cut and pasted from subsystem to subsystem. This is because there are multiple parameters when going through each of the sub-systems and they need to be changed for each of the microcontrollers. For example, there are some sensors at the top of the iRobot chassis that need to first communicate with the MSP430 and then need to communicate with the Raspberry Pi then needs to communicate with the Arduino, so the iRobot can see around its environment. If this basic level of logic was integrated in any other order, then it would fail as a sub system and the whole iRobot system would fail. The software integration must consider the fact that the one system has to get the instructions and then the instructions need to be passed to the next system in the correct order or else the system would fail. Some of the sub-systems in question are: The Arduino system, the MSP430 Microcontroller system, the Raspberry Pi-4 system, the battery system, and the software GUI that would control all the smaller components of the system as a whole.

The integration of the Arduino system was the most crucial and the most tricky to fully implement into the system as a whole. This is because the Arduino system is what is used to program and control the iRobot chassis. The Arduino that's located within the system can be reprogrammed using a data cable, has to be able to communicate with all of the other components that was mounted on the top of the chassis that is independent of the system as a whole. The reason why this is the most crucial is because if we can't have a connection between what makes the iRobot system move around and a connection between the sprayer at the top of the module, or any one of the other deliverables we have, then the iRobot system wouldn't be able to tell where and when to spay the disinfectant. To solve this issue, we need to incorporate another system that can bridge the gap and make it so the data stream of instructions can pass from the Arduino to the microcontroller that is controlling the sprayer. The Raspberry Pi system is the solution, in that we can send the instruction first to the Raspberry Pi and then send then the Raspberry Pi can deliver the instructions to the Arduino first and then to the sprayer when looking at the target. But this brings up the question and another problem, where we didn't know as a group if the flow of data and instructions will return fast enough and is able to pass freely from one system to the next without any drawbacks on the efficiency of the product.

The next system that needs to be integrated in the Raspberry Pi-4 system; this one was in charge of the logic of the entire system. When the iRobot system as a whole encounters an object that is in front of it, we need the microcontroller that can look and move around its environment to tell the microcontroller that controls the sprayer to spray or not. This was done in a multitude of ways but we figured that the Raspberry Pi-4 would be the best solution to have two microcontrollers integrated with one another. For example, if there is a person that is walking in front of a pre-selected target that requires spraying, we would want to detect when they are moving in front of the system and tell the sprayers system to halt on spraying disinfectant. So, there was a data stream from the sensors in the Arduino that detects potential objects and if it detects an obstacle, it sends those signals to the microcontroller that is in charge of the sprayer to stop spraying. The Raspberry Pi-4 is what bridges the gap between the eyes of the iRobot system and the deliverable of the iRobot System. We needed this device to make sure that both microcontrollers can communicate with one another at an instant and they can make discussions on the fly on whether to or not to spray, on when to go back home and recharge or when to move out of the way from potential obstacles in its environment.

The final step for this system to be integrated is the battery system and how we as a group are going to power the entire system as a whole. We need to have it where the systems can draw power from a single source and that source of power can be brought up and down and delivered to each of the individual systems on its own. For example, the amount of power that the Arduino sub-system needed is much more than the amount of power that the Raspberry

Pi-4 subsystem needed. We needed to divide up the number of amperes that get delivered to each of the different subsystems.

One way to do this was through the use of a battery converter that can increase the number of amperes that can be delivered to one sub-system while still keeping the original amount of voltage the same throughout the whole system. This was integrated to make it so the Raspberry Pi-4 system works as intended. Another idea that we had as a group was to use a low voltage regulator, this would drop the overall voltage for one specific sub-system while still keeping the overall original voltage in tack. This would be very useful for the MSP430 microcontroller as this type of microcontroller would need a very low amount of voltage to power it overall. The battery system for the system as a whole was integrated with a single power source to each of the different subsystems. This made everything very clear and concise when working with the different electrical components.

7.3 PCB Fabrication

The fabrication of a Printed Circuit Board is a process that has been used by engineers over the years and has been supported by a multitude of programs that can help evolve/expand the process. In the past it was very hard to get the fabrication of a PCB done and involved a complex and expensive chemical process that eroded conductive layers of the PCB. But today's technologies have allowed the production of PCB's to be printed by printers or made by machines within a few short hours. A design first has to be drawn up or created by manufactures for a full-scale design because printing a circuit board for long term use isn't the best because they don't last as long and aren't as sturdy. Today there are many manufactures that take your custom PCB design and print them for you. Companies such as Ninja Circuits, Autodesk Eagle, APCircuit, Millennium Circuits Limited and PCBart take a Gerber file of your PCB design and then print out your custom PCB design. The cost varies from each of the companies and how many components you put onto the PCB but it ranges from around $20 - $250+. These costs also take into account large orders of PCBs that other companies use to print multiple variants of the same PCB. Some companies look at the fact that many people may want to print multiple PCBs in bulk and they have a lowered price to match said bulk order but it increases the time it takes to manufacture the boards.

Before we printed and manufactured the PCB, we needed to first design the board in software. There are multiple PCB software's that are available for us to use and design the custom board that we desire. Some software environments include EagleCAD, Altium, KiCAD and OrCAD. Out of all of these software environments, EagleCAD is a free software that we all have access to and is the software all of us as a group are most familiar with. Altium, KiCAD and OrCAD are much more complicated and require paid licensees that we do not have access to. With access to any of these software environments we were able to

go from drawings, to schematics and then to analysis and manufacturing. The paid software environments have a more extensive library that can be used by drawing up a schematic for the PCB and can be entered into a PCB simulation via PCB editor. This allows for quality control on the PCB to make sure it works properly for what you want it to do and makes sure the schematic design shows the three different layers of the PCB. The model, symbol and footprint make up the three different layers of the PCB schematic design.

The model layer is where the analysis of all the components is simulated by a PCB editor; they can be shown using AC simulations, a DC Sweep or a transient response. There are many other simulations that can be shown but those three are the main ones for the model PCB layer when showing the components. The symbol layer is the schematic pin input and schematic pin output, essentially it is the pin layout for each part on the PCB. The footprint layer represents the size of the schematic and the layout of where all the components are placed on the PCB. The footprint is essentially the blueprint of the PCB and can show all the smaller components that are locked on the PCB. One all three of the layers of the PCB are designed and finalized, then it links all of them together on the PCB and give the file to a PCB editor.

A PCB is made up of two basic parts that include substrate and the copper traces that link all the components on the board. The substrate holds all parts together and acts as an insulator for the entire PCB.

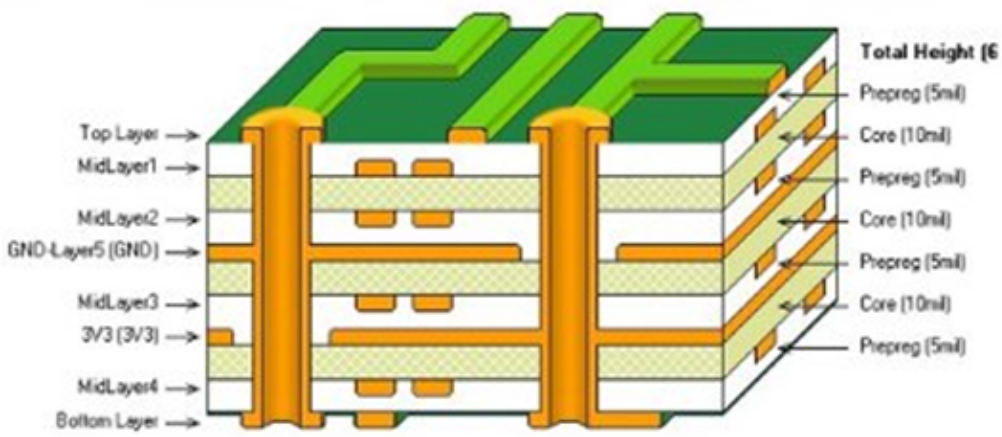


Figure 38: 8 Layer PCB Stack-Up

A PCB board can consist of the cores or the copper substrate, these layers can be stacked on top of one another as shown above and then creates a multilayer board. A PCB stack goes from plated top layers to un-plated mid-layers to plated layers with drill holes. This can lead to very complex PCB board designs, like the

eight-layer PCB stack shown above, that act as very good boards that can hold all the necessary components while still linking and insulating all of them.

The inner layers of the PCB were patterned during fabrication and the cores were aligned and put together. Then the PCB was drilled for its mounting holes on the ends of the corners of the PCB. Finally, all the traces of the PCB were aligned and some of the layers acted as soft planes for them and provide a low impedance as to not disrupt anything in the above stack. This acts as a good connection to power and ground, these are often found on the inner layers and the signal layers which can be found above and below the planes of the stack. There is a clearance between the plated hole and the plated layer, this acts as an isolation between the two layers. A solder mask was added to the top and the bottom of the PCB stack that acts as a thin polymer layer that can reveal the pads and the holes on the PCB. This also prevents oxidation and prevents solder bridges between the clearance holes and pad spaces.

The PCB editor is the final step in the process of manufacturing a PCB, this process describes the board layers for the machines to print out. There are copper traces on top, on the bottom and within the layers of the PCB, drill holes, mounting holes, and part placement all finalized on this step of the process. All the layers were distributed equally and not all the layers are apparent while designing the PCB in the editor. The files while designing generate more than 20 layers. Extensions while using EagleCAD used the file extension “. brd” when designing the board and the extension “. sch” when designing the schematic of the board.

Now that the drawing for the PCB is done, we now finalize the schematic for the PCB in EagleCAD. This is the step that we as a group are most familiar with. In this step we added all of the smaller components on the PCB that act as the control of all the other larger components on the system. We needed to find each special component in the library that is included within the EagleCAD environment and if we couldn’t find the special part that we need to continue with our system then we needed to import a new library with the specific component that is included. After all the parts were imported into the new library and placed on the board via Eagle then we needed to continue the model and make sure we have a power source to power all the components on the PCB. Once we created this model we had created a symbol of the device and showed the number of pins that are active on the PCB. Then the footprint of the PCB is created using PCBeditor and it shows all the corresponding elements and components on the PCB. With all the components listed and placed onto the PCB we were able to make a Bill of Materials (BOM), to keep track of the prices of the different components used. Finally, we were able to create a board outline for the PCB mounting holes and drilling.

Before taking and saving our BRD file, we had our PCB design go through a design rule check and electrical rule check. The DRC and ERC check allows us

to see what is wrong with our design if there is any. These checks will notify us if we violated a design parameter or electrical parameter. These could be instances such as having parts too close to one another or not having a ground for your battery.

After our BRD files were checked off by the DRC and the ERC the next step is to generate our schematic file in EAGLE. This was done instantaneously by pressing the schematic button. Once done all we had to do is wire the traces and leave no air-wires on the schematic file. When drawing the air-wire we wanted to make sure that we were not causing any of the wires to cross and we wanted to make sure there aren't any stray air-wires that can cause any electrical components to malfunction. Once we have all the electrical components air-wired and there aren't any crossed wires then we have to check if the layers on the PCB schematic are in order and the board outline is on the bottom layer of the PCB. Finally, we wanted to do another DRC and ERC check to make sure we don't have any air-wires, crosses, or small shorts on your PCB layout. Once we pass the ERC and DRC checks then we are free to save and download our ".sch" schematic PCB layout. Below is an example of a good PCB schematic layout without any errors or cross wires.

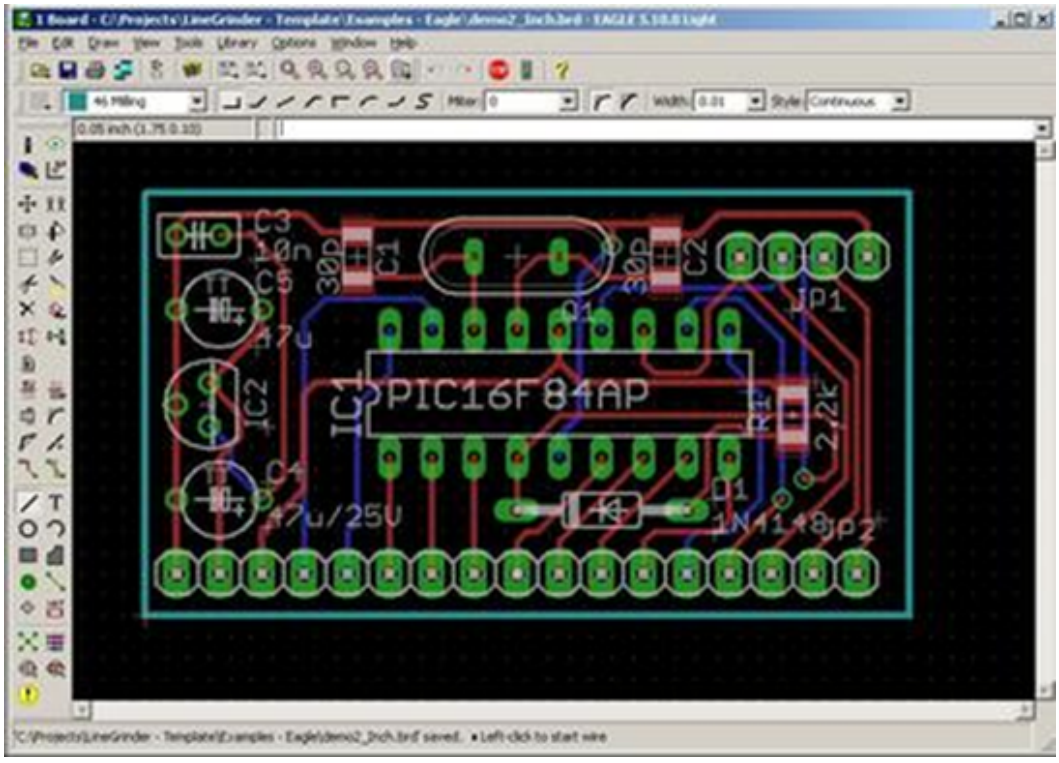


Figure 39: PCB Schematic Layout

7.4 PCB Vendors

Once our PCB is completed as both a BRD file and a SCH file and they have both been processed, the next step is to choose a manufacturer to print out our PCB. For our case we wanted to choose a vendor that has a low cost and a quick manufacturing rate because we wanted our PCB to be here as fast as possible and to be as cheap as possible. You can choose to have your PCB added with all the components already on it for .10 cents to .15 cents per component. Or you could receive your board with all the components but you would have to place all the components on your PCB on your own. We didn't have too many parts on our PCB to make it so expensive where we can just have the manufacturer send us our PCB with all the components already on it and ready to go. This would have helped us in time and money, as placing and figuring out the correct configuration of the PCB would take some time. We also would have to get a soldering iron or go to the lab to solder all the components onto the PCB. Soldering is the process of combining two pieces of metal while using a metal filler that has a low melting point, such as paper clips. This process would have to be done with extreme care, because one mistake where the connections aren't correct then the entire PCB components would fail and not draw power.

Due to the limitations of manufacturing and parts sourcing the PCB vendors we contacted to manufacture and solder our boards required a premium fee to solder the boards for us. We opted to obtain the components and PCB separate as a set of three boards along with components to match would cost half as much as to get a single board manufactured and soldered.

When choosing a vendor to construct our PCB we needd to have one that can get us a return on our investment. They have to be reliable, have a good reputation, and have good and quality products/components. So, while looking for a good PCB vendor, we have decided to look at three different companies that manufacture PCBs. We are looking at Sierra Circuits, RUSHPCB, and Advanced Assembly to meet our needs for our PCB manufacturing.

Specifications and Return Time for Sierra Circuits are as follows:

Standard Two Layer Order; $5/square inch	**Standard Four Layer Order; $10/square inch**
Includes Two Copies of the design; can order these in multiples of 2	Includes Two Copies of the design; can order these in multiples of 2
Ships within 5-10 Days	Ships within 5-10 Days
Material	Material

FR-4 lead-free (370HR material)	FR-4 lead-free (370HR material)
Thickness & Thickness Tolerance 0.062", 0.031" or 0.093" with thickness tolerance of +/- 10%.	Thickness & Thickness Tolerance 0.062", 0.031" or 0.093" with thickness tolerance of +/- 10%.
Maximum Number of Holes Per Board 80 holes / square inch	Maximum Number of Holes Per Board 80 holes / square inch

Table 16: Sierra Circuits Specifications and Return Time

Specifications and Return Time for RUSHPCB are as follows:

Standard Two Layer Order; $7/square inch	**Standard Four Layer Order; $12/square inch**
Includes Three Copies of the design; can order these in multiples of 3	Includes Three Copies of the design; can order these in multiples of 3
Ships within 7 Days	Ships within 7-8 Days
Material Polyimide 2 Layer Flex (370HR material)	Material Polyimide 4 Layer Flex (370HR material)
Thickness & Thickness Tolerance 0.060", 0.030" or 0.090" with thickness tolerance of +/- 12%.	Thickness & Thickness Tolerance 0.063", 0.033" or 0.093" with thickness tolerance of +/- 12%.
Maximum Number of Holes Per Board 90 holes / square inch	Maximum Number of Holes Per Board 100 holes / square inch

Table 17: RUSHPCB Specifications and Return Time

Specifications and Return Time for Advanced Assembly are as follows:

Standard Two Layer Order; $5/square inch	**Standard Four Layer Order; $9/square inch**
Includes Three Copies of the design; can order these in multiples of 3	Includes Three Copies of the design; can order these in multiples of 3
Ships within 7 Days	Ships within 7 Days
Material FR-4 lead-free (370HR material)	Material FR-4 lead-free (370HR material)
Thickness & Thickness Tolerance 0.063", 0.035" or 0.092" with thickness tolerance of +/- 10%.	Thickness & Thickness Tolerance 0.063", 0.035" or 0.092" with thickness tolerance of +/- 10%.
Maximum Number of Holes Per Board 80 holes / square inch	Maximum Number of Holes Per Board 80 holes / square inch

Table 18: Advanced Assembly Specifications and Return Time

While comparing all the different specifications for each of the different manufacturing companies; we saw that they are all very close to one another and each of the companies have a good return time and low-cost ratio. In that case we would have liked to go with the Advanced Assembly as they have the lowest cost per layer of board as well as having one of the fastest return times on the PCB. We are able to have the best of both worlds where we can get the lowest cost on a PCB and get it as fast as possible.

This was changed to OSHPark for PCB manufacturing as they were a cheaper option with our final design. We did come across some manufacturing defects with our board and proceeded to switch manufacturing to EasyPCBUSA. With the board from this new manufacturer the boards cost marginally increased but

added in a silver contact layer for easy soldering, the only drawback was that there were no options for faster manufacturing and shipping.

After getting the PCB delivered back to us, we were required to solder all the electrical components onto the board that we designed in the schematic diagram. We pieced together the PCB since some of us have experience using a soldering iron and it would cost more for us to have all the electrical components already placed on the PCB board. We followed the PCB schematic diagram that we created to see where we needed to place all the electrical components and do so as needed. After we have all the components on the PCB and they are all soldered and working we connect the PCB using connector cables to attach to the chassis of the iRobot system.

7.5 PCB Design

Below is our final PCB design schematic for our entire project with the MSP430. In this we created the subsystem where the MSP430 MCU is connected to the multiple Ultrasonic sensors along with the liquid level sensor and the relay. This is a schematic of the PCB design for the MSP430 that was connected to the Raspberry Pi with a serial connection. Together all of these parts work together so they can act as the eyes and ears of the iRobot system.

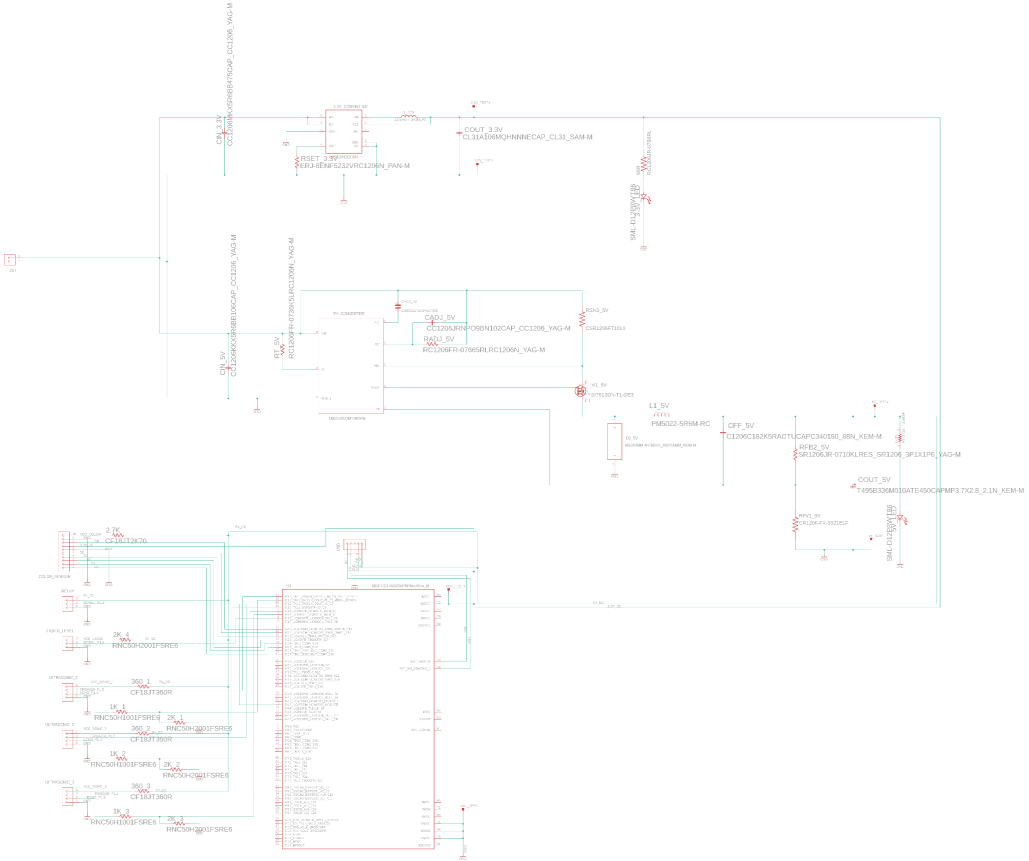


Figure 40: PCB Design Schematic

When we created the PCB design schematic we were able to accurately check if all the parts that we needed are available and if they would be compatible with one another using the Texas Instruments Webench Website. This allowed us to compare prices, accuracy and component compatibility with one another. We then took all the parts that we needed to make the following PCB design and made a library of all the parts and created the schematic shown above. After importing the library of the parts we added them to the design layout and connected the PIN's as needed. We then connected PIN's to the Raspberry Pi module and the Raspberry Pi camera to the PINs of the MSP430FR6989 PIN's. After our PIN connections were successful we then began our testing of the design. We ran DRC and ERC checks in EAGLECAD and made it so our group had an accurate design that worked appropriately.

After creating the schematic for the brains of the Alert system we were able to do basic design and electrical checks in EAGLE to make sure the PIN layouts were accurate. We were able to notice that we were able to run a power source that can sufficiently power all of the components that make this subsystem function. We were also able to notice that the number of groups and I/O pins were sufficient for the iRobot project. If we had to make changes to the system as a whole we would be able to make those changes without any drawbacks and the PIN layout would not hinder us in any way. All in all we as a group were able to make a PCB design schematic design that would be able to evolve to our iRobot system. In that we could easily make changes to the PCB design without changing any of the main components. This was a choice that we made in the testing phase of the PCB design, so we chose a PIN layout that would act as the cleanest and most efficient.

8.0 Testing

The following section introduces how the testing for all of the components related to our project were examined. Each test includes the purpose, environment, procedure, and the conclusion to our experiments. These tests are meant to be used to make sure these parts are successful for the next phase in building and combining all of the components.

8.1 Software Testing

Testing anything related to software was important for this project to make sure the robot was autonomous and could function properly without any problems. The IRobot needs to be able to properly avoid obstacles while communicating with the Raspberry Pi installed within to then communicate with the microcontroller up above so that it can control the spray system. The sensors all over the robot need to communicate with both the Raspberry Pi and microcontroller so that the features can be implemented correctly. These components must be able to trigger LCDs and responses from the IRobot that can be implemented with no hitches.

There are many different types of testing for software.

1. Unit Testing
2. Integration Testing
3. Regression Testing
4. Smoke Testing
5. Alpha Testing
6. Beta Testing
7. System Testing
8. Stress Testing
9. Performance Testing
10. Object-oriented Testing

Unit testing is very popular. The computer engineer of the team tested individual functions before testing the whole thing. Testing functions related to a feature is a good process to be in so that the entire project is not overwhelming. For the Autonomous Sanitation Robot, we created a function for a feature and tested it before moving on. For example, we implemented our logic for the robot to correctly evade objects while maintaining its autonomous functionality. This has to be tested before going onto detecting a certain color that triggers the spray mechanism. That was also tested, flushing out all the bugs before moving on. The programmer then tested the logic for the ultrasonic sensor which similarly also has a say on whether or not the spray mechanism is triggered. While the two sensors just mentioned have the same idea of controlling the spray system, it is important to test each one individually in case there is a problem with one of the functions. It is harder to determine a problem that has a lot of changes in a row while it is easier to test every new idea of logic that is written.

Integration testing is similar to unit testing, except integration is for multiple functions. Integration testing was used for our product. This can be seen when multiple functions are needed to see one feature implemented. Basically, one function is meaningless. An example of this can be seen when the disinfectant liquid is low. A function can be written for the IRobot to go back to its charging home or to turn on an LED which is pointless if it isn't included with the function to detect when the liquid is low. This is when integration testing was implemented because multiple functions were created in a row to implement the feature of what to do when the sensor detects the liquid is almost empty at 5%.

Stress testing is vital for making sure that a product can withstand all conditions. It is necessary that our autonomous robot goes through various tests to make sure it can get through every condition. Our robot was tested to make sure it can withstand a certain amount of weight on top of it and still be able to move without a problem. This factor is important because the robot needs to still stay nimble to avoid obstacles and function like it should. We also charge it all the way up and run it until the battery is drained to make sure it can last for as long as it says it can. We also checked to make sure it isn't overheating with the added weight on top. The sensors that are added with everything might also make the insides of the system run hotter than usual. The sensors were also stress tested to insure the color detector sensor can correctly spot a color over and over for hours. This is also similar for the ultrasonic sensor that needs to be able to detect distances for hours. The color detector sensor was also put through trials to make sure that if the color red is used for no spraying, that different shades and lighting can also reproduce it to detect the color correctly still. The lighting can be important because where the product is tested is different than when the actual graded tests come for the robot. We have stress tested our spray system battery life. The pump needs to be able to last for hours spraying every 30 seconds which makes it important it can last. The sensor that sits in the liquid dispenser also was tested to make sure that it can withstand conditions inside liquid for long periods of time and can still register accurate values. These stress tests apply to software testing because these components need to last to be able to accurately record data to then ensure the features are implemented properly.

Object-oriented testing is beneficial for the whole picture. It focuses on testing the software and hardware at the same time. It is mostly for making sure that the requirements have been met. These requirements are based on what we have set as features and what have been promised that can be accomplished. These goals were realistic and these tests were able to prove if they are possible. This form of testing was combined and implemented at the same time as the other tests.

We are considering trying beta testing. To basically give the finished product to someone that might need the product in a real workplace. This is done at a

customer's site. We discuss more about trying this if we can get everything put together in time next semester.

Performance testing is one of the final processes testing the software of the product. This is utilized by our group by putting our autonomous robot with all of its functions and features implemented to stress test it. It runs in an environment first to make sure it is working as it should and then to make sure it works in all scenarios, we made it tougher. We added objects to the field and different colors to make sure the sensors can detect everything properly. This final test is to make sure we can replicate and prepare for the exact kind of test we were presenting. We want to be able to expect the same results.

Name of Test	Advantages	Disadvantages

Unit Testing	Can ensure success of every feature required.	It can be more time consuming than any kind of test because the process goes through every function with testing before moving on.
Integration Testing	It could help a group collaborate by implementing many functions as a team and test after the fact.	It could lead to problems if something goes wrong, it is harder to determine what is the problem because everything was put in at once.
Stress Testing	Helps prepare for any situation possible.	It can be extremely time consuming and costly depending on components breaking because of the stress put on everything.
Object-oriented Testing	This is best implemented with the other forms of testing so that everything is tested thoroughly.	There isn't really a downside because this process needs to be done at the end regardless to ensure we have met our requirements.
Beta Testing	Can give real-world results to analyze if our device can be used in the workforce	It could be a problem if the device was broken while beta testing before the actual graded test.
Performance Testing	This gives us what we should expect for when it is put through the same environment as the graded test.	The downside is that we could focus too heavily on certain conditions when we should be testing everything possible that could happen.

Table 19: Software Testing Advantages and Disadvantages

The above table shows the advantages and disadvantages that can give a good summary for the tests that we utilized.

We were looking at all of these to implement how we test the progress of the software related to everything we need to do. Adopting one of these types leads to fewer bugs, problems, and essentially creates a quality control system to make sure we are making progress.

These tests were done mostly individually due ot Covid. We did attempt to meet when it was safe to do so. These tests could also be applied to general testing of

the hardware; they focus on making sure that as long as the hardware can last that long, the software will still be functioning correctly the entire time.

Objective: The point of this test is to make sure that every component that deals with software can correctly respond. It is also to make sure that the IRobot, microcontrollers, and Raspberry Pi are not defective.

Environment: Because of Covid, as a group, we tested the components at our houses mostly. We communicated through discord and zoom reporting our results. We met to build the physical parts in the senior design lab. We were able to use our own computers and runn the correct software IDEs that are needed for our autonomous robot.

Procedure: The following steps were used to test the Software of the Autonomous Sanitation Robot:

1. Charge and power on the IRobot Chassis making sure it turns on.
2. Connect the Raspberry Pi to a computer and then inside the IRobot chassis.
3. Test the IRobot in a secure area with no logic. This is to just test it roaming around to make sure no motors are defective.
4. Check that all wiring can be connected properly to a computer including the IRobot chassis.
5. Test each sensor with the microcontroller to make sure data can be received with one another.
6. The ultrasonic sensor readings on the computer should detect distances correctly.

Conclusion: The readings and values should make sense and be close to what we expect. If the values are determined to be ok, the components can move forward and start to be utilized together. If anything is incorrect or if a component is malfunctioning, that part was be returned to the store because it was just recently purchased.

8.2 Hardware Testing

8.2.1 TCS3200 Color Sensor and Raspberry Pi 3

The TCS3200 Color Sensor is a key component to our project's success. It helped us demonstrate and show that our autonomous robot is able to recognize commands that are telling it whether to spray or not to spray and act accordingly.

The color sensor we used is one that has 10 pin outs compared to some models that have 8 pins. Most typical models have pin outs of S3,S2,Out, Vcc/Vdd, S0, S1, 0E, GND, however the model we have has two additional pinouts of LED and another GND. Looking at the manufacturer notes and details it was stated that

these two variations just allow for more usable changes in the current model available to us. All features of both versions are identical and should operate the same.

To begin testing we utilized the UCF senior design lab, and the equipment given in the Analog 2 Discovery Kit. To start testing components there was a quick visible check of the color sensor to ensure that no pins had been damaged. Once this check was complete the color sensor was placed on the breadboard. The wiring of the color sensor went according to figure 31. Using the bread board initially helped us with connecting certain switches that we tied together. In the connection diagram it is shown that Vcc, S0, S1 are all tied together so that was the starting point. Along with those pins being tied together there was also the 0E and GND pin tied together. Once the appropriate pins were tied together a DC power supply was used to power the color sensor. In the actual implementation of our project the color sensor was powered by our 5 volt converter through its specific GPIO pins.

With the DC power supply the pins going to the associated 5V power were bridged to the power and the pins associated with the ground were bridged as well. The initial testing showed no visible results on the four Photodiodes present on the color sensor. The first troubleshooting step that was taken was to ensure that the power supply was giving the right number of volts needed to power the color sensor. By using a Digital Multimeter, a series connection was made with the power supply and checked to see if the proper number of volts were being given off. The check showed that the supply was giving the accurate number of volts to operate the color sensor. This was an obvious step that was more a check to ensure that we as tester were accurately enabling the output of the DC power supply before going into deeper testing

The next steps were to probe the areas where the power supply was going into. This is specifically the Vcc, S0, and S1 pins tied to the 5 volts and the 0E and GND that are tied to the ground. With the power being supplied to these pins the digital multimeter was probed at the pin's connection site, specifically the pin into the breadboard. This test was done to the VCC pin side, and it was shown that the color sensor was indeed receiving the power being supplied. The same was true with the grounded pins which ultimately means that there was another issue besides that of power.

With the power not being an issue, we looked to the troubleshooting section of the datasheet. Our biggest issue was that we thought because the photodiodes were not loading or shining immediately that our component was not being powered, however it was the fact that code must be uploaded into the color sensor to begin having the diodes light up so they can read the associated color.

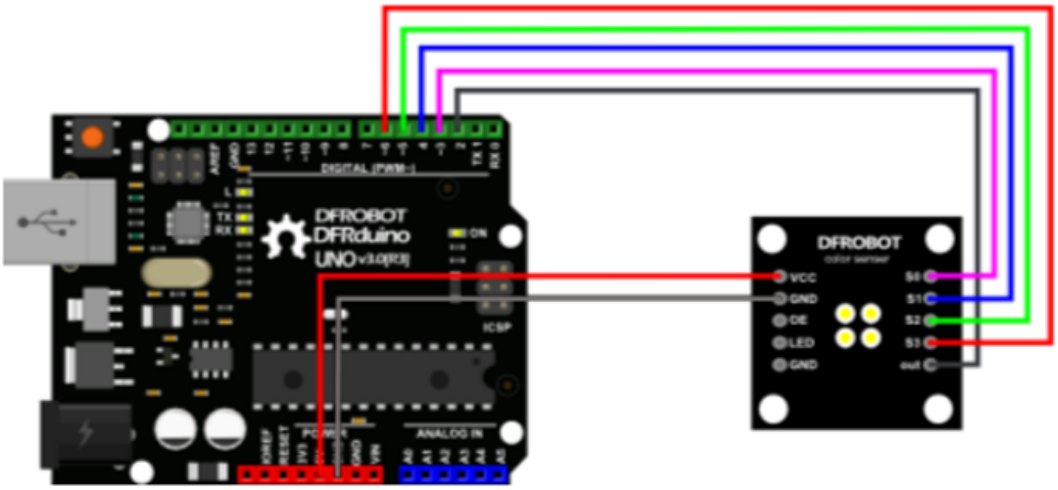


Figure 41: Color Sensor and Arduino Board Connection

Once the issue of the power was figured out the testing went on to the connection between the board and the color sensor as that is how the sensor receives its power. The breadboard testing was helpful in ensuring that the color sensor was working and receiving power to the necessary pins. The first step before connecting the color sensor to the raspberry pi sensor was to test out the connection recommended in the data sheet with an Arduino board. Now since the raspberry pi board is the main component this is just an additional step of testing to ensure connections are working properly so we may be able to isolate an issue that may arise in any integration steps further along the road.

The available Arduino board was one that was available to our group through former classes so we decided since there was no disadvantage in terms of costing, we could use this board to help test the color sensor. By using the wiring diagram shown in figure 40, the color sensor was set up and powered through the Arduino board that received its power from a laptop through a micro usb to usb connection from the board to the laptop.

The connections in the Arduino board were wired and connected directly based on the direction given by the datasheet. This followed that power was being supplied to the appropriate pins as well as the ground going to its respective pin on the Arduino board. For the other pins they were connected to the general-purpose input and output pins on the Arduino board. Once this connection was set up there was a sample code given that helps activate the color sensor. The color sensor is set to read red, white, blue, and green through its 4 photodiodes.

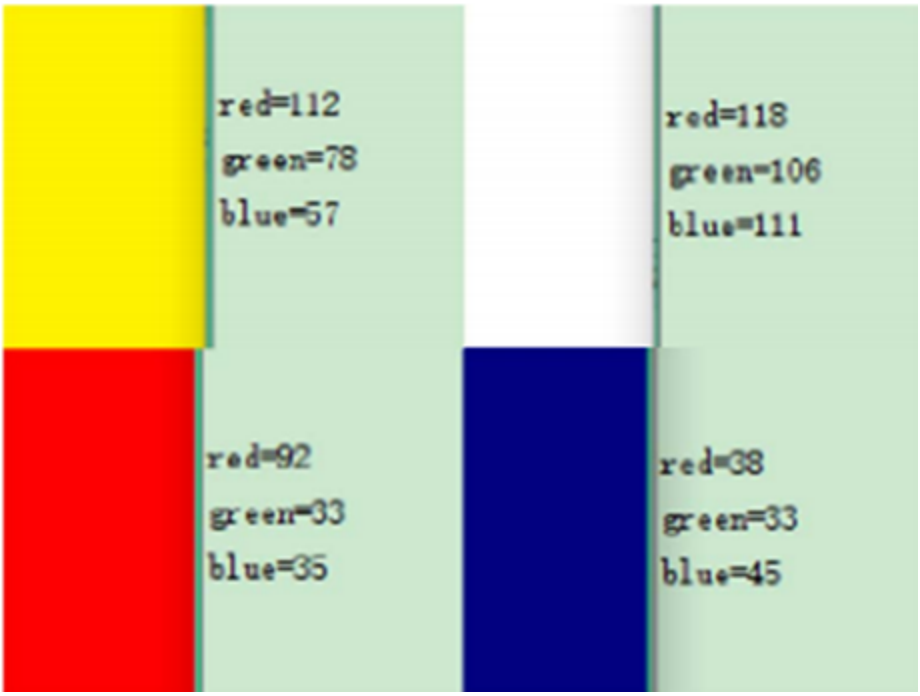


Figure 42: Example of Color Sensor Output

Figure 41 shows an example of the reading that the color sensor gives when wired with an Arduino board. The readings show the given color registered by how much of the core 3 colors of the photodiode is detected. So, for the first color detected the color was yellow and the color sensor reading gave off strong number correlation to the red and green which are the two colors associated with making yellow. This same logic can be used to see how the other colors are detected and read.

Once this connection is done it can show the ability that the color sensor has and how versatile and interchangeable this sensor can be, however for our application we are gearing to use the Raspberry Pi 3 model B+. This testing of the Arduino board and color sensor does have some benefits for our group as it shows the possibility that our design may be shifting to using the Arduino board rather than the Raspberry Pi. With the ordered raspberry pi, we have a set time we can return it so there is a possibility that we can shift our design to use the Arduino board if at all necessary.

Shifting focus back to the original setup of the raspberry pi controller and the color sensor. Through testing on the breadboard and the Arduino board there are fluid examples and stats of the color sensor and its capability. Now comes the integration of the Raspberry Pi and the color sensor which is one of the key subsystems we want to work with our raspberry pi module. The raspberry pi has 40 GPIO pins that were utilized to connect with the pins of the color sensor. The color sensor requires 5-volt power so all necessary pins that need this are tied

together on the color sensor and can be paired to any of the 5V power on the raspberry pi's GPIOs. The raspberry pi similarly to the Arduino board is powered through the computer through a micro usb to usb connection from the laptop to the board.

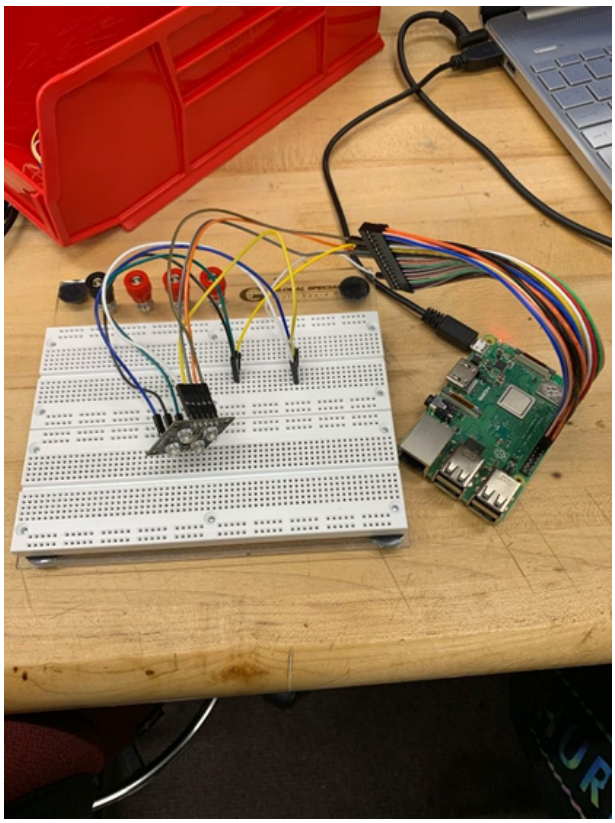


Figure 43: Raspberry Pi and Color Sensor Integration Testing

To integrate the color sensor, we utilized the breadboard to help tie together certain pins that needed power together. So, in figure 42 it shows this connection as the breadboard is used to tie together Vdd, S1, S0 and connect them to pin 2 on the raspberry pi's GPIO. This pin is a 5-Volt power pin and similarly 0E and GND were tied to pin 14 which is a ground pin. Since this is the integration of the Raspberry Pi and not the Arduino there is no code given to test it out, but due to prior testing we know that the color sensor is working well. It was a key goal and objective to make sure we can code up a script that lets the color sensor work accurately.

8.2.2 Ultrasonic Sensor and TI MSP430FR6989

The Ultrasonic Sensor was used to detect objects that are in close proximity to then alert the user through LEDs and audio speakers that the sprayer is near them. This was used to control the spray system from triggering. The sprayer will be put on hold if an object/person is detected in front.

To test, we confirmed that the sensor was able to have power sent to it. We then developed a system to confirm it was registering distances correctly. A breadboard and an LCD display was utilized to display the distance which would then ensure that the Ultrasonic Sensor was actually working like it should.

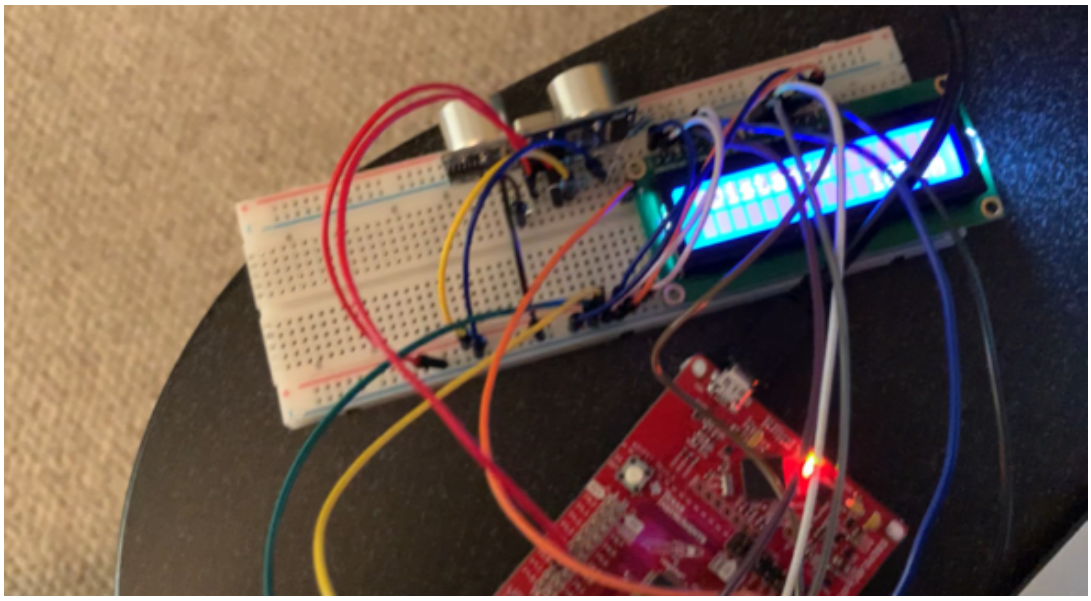


Figure 44: Ultrasonic Sensor and MSP430FR6989 Testing

The photo up above shows how our testing setup was created. We connected the MSP6989 to one of our computers for power. We then ran our script from our Code Composer Studio that connected to our Ultrasonic Sensor and LCD display. The display can output the distance of whatever object is in front of the sensor. We tested by putting our hand in front of the sensor at varying approximate distances. These distances were outputted and made sense by how close or far the hand was. The largest distance we tried was at 140cm. The Ultrasonic Sensor can detect up to 70 feet if needed, however, we only looked at distances of 3-5 feet as being important to us for this project.

We also needed to confirm a microcontroller that can be used to communicate between the custom PCB board and the Raspberry PI. We believe that the MSP6989 satisfies that requirement.

We implemented our logic in C. The thought process behind choosing this language was how close it can be to a low-level language while at the same time giving us the benefits of a high-level language.

In the figure below, you can see the distance of one of our hands being displayed as 3 centimeters away from the sensor. This test was complicated but was

important to make sure that our sensor was not broken and that it was possible for it to correctly detect objects within a reasonable distance.

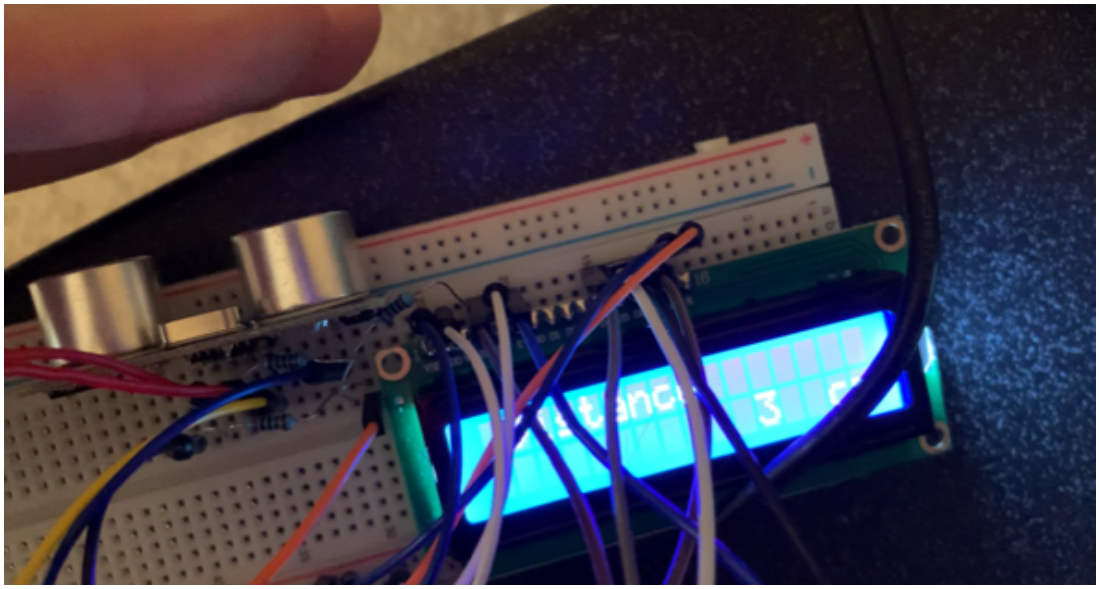


Figure 45: Ultrasonic Testing Distances Displayed

8.2.3 Create 2 Programmable iRobot Testing

The last thing that we had to test was the actual iRobot Chassis or the programmable iRobot that we purchased on the iRobot website. We plugged in the iRobot and had it charge up for a few hours before running it on the floor of one of the group members' homes. We were able to check if the iRobot could make it back to its recharge station, if it could maneuver around its environment and most importantly check if the Raspberry Pi could deliver commands to the iRobot.

Figure 46: Create 2 Programmable Robot at Docking Station

In figure 46, the robot is connected to a charging station that charges through magnetic clips on the bottom of the chassis. It can find its way back to the charging station when it is out in the field by bluetooth. When it was out in the same room roaming around, it took about four minutes to figure its way back to the station doding chairs. The code needs to be fine tuned so that it can avoid some obstacles without actually bumping into the obstacle. At the moment, the rubber padding acts as a buffer whenever it makes a mistake.

We were able to see that the iRobot was able to make it back to its rechange station without any issue and was able to maneuver around basic obstacles that stood in its way. We were able to see that the basic functionality of the iRobot was functional but we still had to test the main feature of the iRobot being fully programmable. After a few hours of opening the system up and hooking up the communication cable we were able to see that the iRobot was capable of taking commands and executing them correctly.

We were able to run a basic script on the Raspberry Pi and connected the two together so they could communicate with one another. After a few trials and

errors we were able to make it so the iRobot would respond to our commands and follow the instructions that we programmed onto the Raspberry Pi. We used basic C logic and were able to make this function after only a few hours of testing. This was actually a little tricky as we had no prior knowledge to the system but can now accurately set up and program the iRobot. This was a huge breakthrough in our testing and we were able to mark off the fact that the iRobot can take directions from the brains of our system.

The iRobot system testing was a huge success and we as a group can now accurately and efficiently make the system run a command, dock and maneuver around any environment it may encounter.

In conclusion we were able to test these products and got useful information that would help us make discoveries in the future. Through this testing we hope to take this information into senior design 2 and if necessary, make proper adjustments and replacements to equipment. We were able to make sure all the parts and components would accurately work with one another and are compatible with all the subsystems that would be implemented into the system as a whole. Also through this testing, we gained the knowledge we needed to make all the components work together and how all the parts would work with one another. This information came in handy when we entered senior design 2 as all the testing, compatibility testing and component testing greatly impacted how our iRobot system functioned. If necessary we can make changes on the fly to our design and still know how all the parts would function with one another.

8.2.4 Sprayer Testing

In this section we are testing the effectiveness of the sprayer unit that was selected for this application and the way we were tapping into the spraying mechanism.

With the sprayer's reservoir filled with water and the included AA batteries installed the sprayer operation was tested. The sprayer operated nominally and within the parameters stated by the manufacturer. The spray pattern adjustments work as advertised and the sprayer head can be taken apart easily.

Figure 47 : Disassembled Sprayer Head

As shown in the figure above the sprayer head is rather simple on the inside. The top of the unit houses the power source, in this test the unit is powered by two AA batteries. There is a tube that moves from the nozzle to a leak blocker that is actuated by the trigger when squeezed. This leak blocker was removed as the squeeze trigger mechanism is going to be replaced with a relay. After the leak stopper the tube moves to the pump that has a black tub attached to it that goes to the reservoir. With the insides of the unit unveiled it shows that if needed the enclosure can be discarded and the sprayer unit can be fitted to the robot as is without any major modifications required.

On the reservoir end of the unit there is a one liter bottle to serve as the reservoir and the black tube that protrudes out of the sprayer head leads into the cap of the reservoir bottle. The end of the tube is weighted to allow the tube not to float and be able to draw all the liquid from the bottom of the bottle.

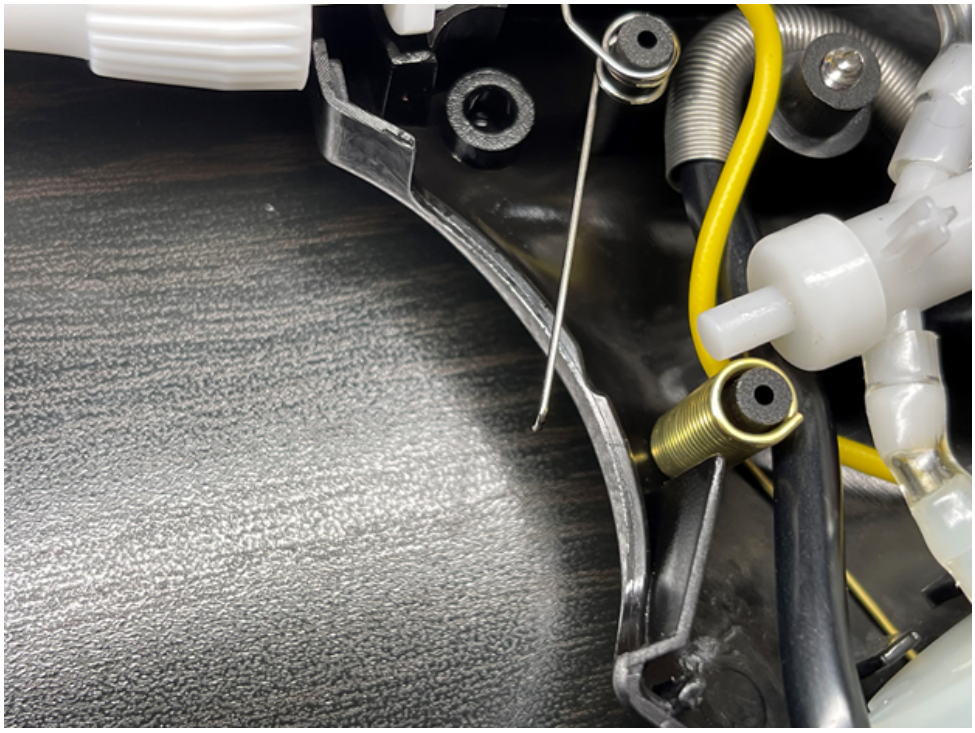


Figure 48: Trigger Mechanism with enclosure removed

With the trigger enclosure removed as shown in the figure above this gives us a view of the way the pump is activated. One end of the battery is connected to the pump while the other end of the battery is connected to the protruding wire, as can be seen in the figure above. The golden coil seen in the figure above is connected to the other terminal on the pump. When the trigger is squeezed the wire makes contact with the golden coil, thus completing the circuit and activating the sprayer.

For us to be able to activate the sprayer without the need to squeeze the trigger every time we plan on hardwiring it to the mainboard. This means that we wired the two contacts of the sprayer to a small relay, this acts as our switch. This relay was then wired to our main board. When the relay receives power it closes the switch allowing the circuit for the sprayer to complete which causes the sprayer head to activate.

As we can see in *Figure 47* the battery compartment is located at the top of the sprayer. In testing we checked if we would be able to supply power to the sprayer without the need of extra batteries in this device. To test this we hooked the positive and negative leads of the battery housing in the sprayer to a function generator. With this done we attempted to use the sprayer at various voltages. While doing this we checked what the average voltage of a AA battery was, which is between 1.2 and 3.6 volts. We found that as long as the voltage inputted into the sprayer was within this range then the sprayer would act nominally.

When we increased voltage out of the normal specification then the pump would increase in speed. This is a good addition to the function, but applying too much voltage could cause the sprayer to degrade faster. This additional voltage could also cause the sprayer to burn out and cease to function.

8.2.5 Battery Testing

Although the power draw of the entire system is minimal, we have done testing on the batteries we used for this application. That would be Nickel Metal Hydride (Ni-MH) batteries. These come in AA formfactor and are able to be combined to form a larger cell array. With that in mind we want to stray away from using an enclosure and simply removing the batteries from the enclosure to charge. Depending on the number of batteries being used this could be cumbersome.

With this we tested if we can build our own custom battery pack for this application. By connecting up the batteries in series we are able to achieve a large voltage. This gives us the flexibility to build a battery pack as large or small that we need. The only part of this test that is still unclear is the type of connector that was used to connect the battery pack to the main board. This was chosen to be a common JST connection that includes a charging cable that plugs into a usb port.

8.2.6 Speaker Testing

When we received the speakers we were worried they were not doing to be loud enough for this application through our initial testing this was proven wrong. The speaker kit included two three watt speakers along with a miniature class D amplifier, this simplified the testing process as the amplifier only needed an audio source via the 3.5mm jack and power from an external source.

Utilizing the function generator we connected the amplifier to the power source and used the desktop computer provided in the Senior Design Lab as an audio source. This setup worked flawlessly and the addition of the volume knob on the amplifier makes adjusting the volume of the speakers very easy.

What has yet to be tested on these speakers is the power draw. Depending on the power draw from these speakers we would have to reduce the volume or even change out the units.

In testing these speakers showed to have a reasonable size and are easily mountable to the robot. Unfortunately the wire length is lacking this would mean either we wire the speakers close together or we can extend the speaker wires to allow for more flexible mounting on the robot.

Figure 49: JBL Speaker

We opted to use the JBL speaker insead, because we were unable to get the other speaker to work on our PCB design. One of our group members had a JBL speaker and we decided to use that as the specifications for the two speakers are the same and would have the same outlook.

8.3 Ordered Parts

We ordered all of the components required except for our custom PCB board. We knew it was important to order everything as fast as we could because of the electronic shortages from Covid. We looked at different sights to order all of our parts and were able to find the best prices for the parts that we needed. We also were able to compare the different prices of the microcontrollers and some of the more miscellaneous parts that we needed by using the Texas Instruments

website. We could price match the exact same parts and order the ones that had the lowest prices.

Some of our group members already had the ultrasonic sensors with other important small parts that we need. We got it from a kit bundle on amazon.This kit bundle included ultrasonic sensors, male to male and male to female jumper wires, LEDs and resistors. This was a good head start on the amount of parts that we needed to order and with this kit we were able to knock off some of the more challenging parts that we needed for the iRobot system. We ordered the IRobot Create 2 Autonomous Robot. We purchased sensors including the liquid level sensor, and the color sensor detector sensor. The Raspberry Pi 3 has already been received as well. We have purchased the automatic sprayer mechanism as well.

These components were all tested to make sure nothing was defective. We want to make sure everything is working and that what we researched is confirmed to be true to work with our project. We do not want to be ordering any parts next semester and finding out what we need happens to be out of stock. We do not know how the future will be with Covid.

As seen in the photo below, everything has arrived except the IRobot chassis and custom PCB board. We used our Raspberry PI and MSP6989 to test our components. We have looked at our audio speakers, ultrasonic sensor, color sensor, and water depth sensor. We confirmed each component is able to work and are functioning properly.

Components as seen in the photo:
- Ultrasonic Sensor
- Liquid Depth Sensor
- Color Sensor
- Raspberry Pi
- Spray System
- Audio Speakers
- MSP430FR6989
- LED Display
- Breadboard

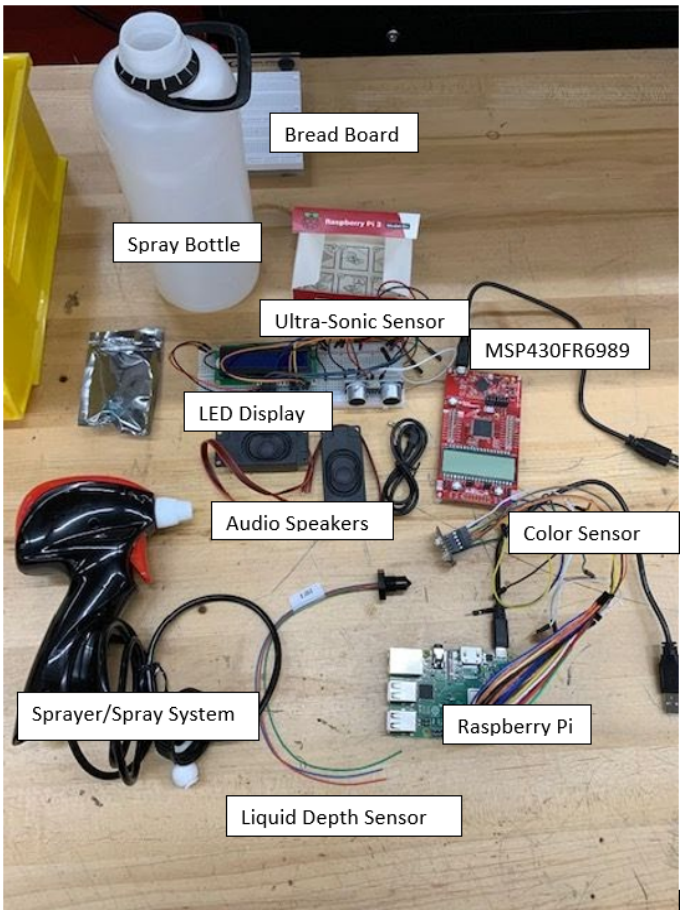


Figure 50: Picture of Components

8.4 Prototype Demo

Once all the parts are assembled together and appear to be in working order we assembled a test environment. In the test environment we build a scaled down hallway most likely built out of cardboard that connects to form two loops. These hallways form a figure 8 type of shape. With this built the robot along with its base station was placed in one corner of the simulated hallway.

The interior of the simulated hallway wasbe lined with white paper to simulate clean white walls. Along certain areas of these simulated hallways colored squares were placed to simulate a dirty area on the wall. And some cubes were placed to simulate a user in the hallway.

When the robot is activated then it should start to patrol the hallways, when it finds that there is a simulated dirty area it proceeds to clean the area by spraying it with the solution. When the robot comes across a simulated user it should flash its LEDs and play the chime to draw the user to its presence.

When the reservoir runs low on solution the robot should begin to flash its “Low Solution” LEDs and play its “Low Solution” chime to notify the user that the reservoir needs to be refilled. The robot should also begin to make its way back to the base station for the user to refill the solution.

This demo should be run multiple times to ensure all of the robots systems are fully operational and have no bugs or flaws. Everytime the robot makes a mistake the situation should be logged and the team began the troubleshooting process. This should happen a couple of times with trial and error to properly dial in the settings for the robot to operate optimally. This would be in the final stages of the iRobot system testing phase. This is a process where we can make slight adjustments to the coding of the system or slight electrical design changes where maybe a sensor is facing a direction where it may not be most efficient. When making all changes in this testing environment we want the environment to stay the same and make slight variations of the system as whole and make note of what we changed and how much better or worse it did in the same environment.

By doing this type of trial and error testing we can conclude that settings on the iRobot system would work best to most accurately show off our deliverables. We as a group want our system to have the best showcase of the different deliverables and making these slight tweaks to the system as a whole can show which change or tweak made the system better. All in all, we want to run the system in the same environment for as long as possible so we can gather a large amount of data and changes, then log that data and pick the best possible variation of the iRobot system that most accurately describes our specific deliverables.

We are looking at a list of different features displayed in the table below.

Features proposed	How each feature should run
Spray System	Correctly releases a mist every 30 seconds
Ultrasonic Sensor	Detects objects/people within 3-5 feet to then turn on/off the spray system.
Color Sensor	If red is detected, the sprayed stops releasing spray for 30 seconds. If Green is detected, the spray system is reinstated back to normal.
Audio Speakers	When the Ultrasonic Sensor detects an object/person, an audio queue is alerted on the chassis.
LEDs	LEDs turn on correctly with correct colors based on the current phase of the system Green LED for spraying and red LED for disengaged.

Table 20: List of Features Proposed

The changing of all parts, tweaks, and component switches had to be accurately recorded in order to find the best combination of system settings to most efficiently execute said deliverables. The table above shows what each part does and how it affects a certain part of the iRobot system as a whole. This was useful when testing our iRobot prototype and finding the best possible iRobot system.

These list of features would be some of the different components that we would have to adjust and record when testing the prototype. We want to make note of all the slight adjustments that we make to the prototype and find the best variation of the iRobot system that makes it through the environment while executing all the intending deliverables. For example we may want to move or add another ultrasonic sensor if we observe that the iRobot system is failing to maneuver through the testing environment. Then after making the necessary adjustments we can test the system again and record our results. If the iRobot system was successful in completing the testing environment then we would be able to keep that new prototype and run testing again till we find another failure. We repeated this process till the iRobot system prototype has a close to no failures and successfully executed all the required deliverables that we set out for.

After going through all the features and parts of our system, we decided to only go for the sprayer head and bottle, color sensor, ultrasonic sensor and an audio speaker. As for the LEDS, we had to use the LEDS on the MSP430 Development board to get those to flash at the appropriate times for our iRobot system as a whole. These were the parts we were able to get working due to COVID-19 and were the parts we had the least amount of problems with as a whole. We repeated the process of testing all these parts in unison for the iRobot system till we had executed all the required deliverables that we had set out to do.

9.0 Administrative Content

This section of the document shows the breakdown of how the team managed the time and how we monitored and managed the set budget. The budget has been decided by the team members based on the average cost associated with senior design projects with similar specifications. The milestones cover the range of time from basic idea conception to the final presentation.

9.1 Project Milestones

This section show the milestones for this project along with their projected completion date. It should start from the initial conception stage that started in the Summer 2021 semester of Senior Design 1 and be finalized in the Fall 2021 semester in Senior Design 2. The milestones set for Senior Design 1 have been set by the faculty and replicated in this document. We were able to look at the Senior Design 1 Webcourses to determine when each of the milestones should be due and then mark each date that a certain section is due.

The milestones for Senior Design 2 are totally compiled list by the team members as when they believe these tasks are to be finished. The Senior Design 2 milestones are what we chose for this Fall 2021 semester. We as a group were able to make a list of when we are able to complete each task for Senior Design 2 next fall. We know that all of the dates could be subject to change and all of the work that we put into Senior Design 1 directly impacts that amount of workload we had in the following semester. We understood this statement the moment we started this paper and wanted to make sure that we could be as accurate and efficient as possible when making a time table of when we plan to finish and unveil our iRobot system. We planned each of the Senior Design 2 milestones by the amount of weeks rather than an exact day. We did this so we can more accurately gage when we have a milestone done and give full amounts of days to a certain task for that week or weeks. With every week our PCB was unfinished we had less and less time for testing so we had to allocate time and resources to other unfinished parts of our project.

Each of us, as a group, was able to work together to meet each of these deadlines and make sure all of the work is completely accurately and efficiently. We made this semester in Senior Design 2 flow as smoothly as possible by planning and following our milestone path that we set forth in this semester. All in all we were able to make it so that all the work was able to go with the least amount of speed bumps or errors in the way.

Milestone	Expected finish date
Complete Divide and Conquer 1.0	6/11/2021
Get project approval in DC meeting	6/17/2021
Improve on Project and Complete Divide and Conquer 2.0	6/25/2021
Begin Senior Design I Documentation and have 20 pages done by the end of this week	6/27/2021
40 Pages completed of Senior Design I Documentation	7/2/2021
60 Pages completed and turn in 60-page Draft Senior Design I Documentation	7/9/2021
80 Pages completed of Senior Design I Documentation	7/16/2021
100 Pages completed and turn in 100-page report submission	7/23/2021
Complete final document	8/3/2021

Table 21: Senior Design 1 Milestone Schedule

Milestone	Expected Completion Week
Place order for all parts	1
Critical design review	4
Construct a working prototype	5
Adjust and refine any constraints and or parts in prototype	7
Midterm Review	8
Adjust project constraints based on midterm review performance	10
Demo final product	13
Turn in final Senior Design II documentation	14

Table 22: Senior Design 2 Milestone Schedule

9.2 Project Budget

Below are two tables that illustrate this budget project. The two tables represent the possible budget if a programmable kit is used and the other table represents if the kit is not used and rather a robot chassis is used. Table 1 has the budget for the programmable robot and Table 2 has the chassis robot budget. We wanted to have as many options when it came to budget because, obviously, we wanted to make the same project with the lowest possible pricing. This is what led to each of the project budgets and why the iRobot chassis was the key component in each of the pricings.

The budget for this project is our final accurate budget by the team members utilizing the total cost of what our projects cost and the components required for

this specific project. Due to the COVID-19 pandemic availability of most parts were unknown and possibly have prolonged restocks. The shipping on many of the electronic parts could also have extended shipping and processing times. Due to this product prices and shipping cost could rise causing the team to go over budget. The total cost for the PCB was also driven up as when buying and creating the PCB, we ran into problems for it not correctly working. This then paired with COVID-19 and we were unable to get the PCB working exactly how we wanted it to. But through trial and error we finally got it to work and it affected our budget as well.

This project was self financed by the team members and the cost was split evenly amongst all four of the group members. Due to this the team members strive to do everything in this project as efficiently as possible without wasting any components in an attempt to keep the cost down as much as possible. The goal of this budget is to ensure that the proper parts are to be bought but to not needlessly spend money on parts the "thought about using" and never did. We don't want to sacrifice the quality of one part and the compatibility of another because of pricing. But on the other hand, we also want to budget this project so we can keep it at the lowest possible price. This is where budgeting can come in handy and you can see the drawbacks from choosing a cheap product versus buying a more expensive product. This could backfire in that a cheap product could cause problems for us in the logengiviety of the iRobot system. But maybe buying a more costly component could be just as good as the cheap variant. This is why we did extensive research and testing to make sure we're getting the most bang for our buck and not wasting our money on parts that could compromise the life of the iRobot system as a whole.

Component	*Price*	*Quantity*	*Total*
iRobot	***$199.99***	***1***	***$199.99***
Custom PCB (OshPark 1.0)	***$20***	***3***	***$60***
Custom PCB (Oshpark 2.0)	***$40***	***3***	***$120***
Custom PCB (EasyPCB)	***$65***	***4***	***$260***
Liquid Level Sensor	***$38.25***	***1***	***$38.25***
Color Sensor	***$16.48***	***1***	***$16.48***
Raspberry Pi	***$40.48***	***1***	***$40.48***
PCB components (1.0 + 2.0)	***$200***	***2***	***$100***
Raspberry Pi Voltage Convertor	***$9.95***	***1***	***$9.95***
Total			***$865.15***

Table 23: Budget with Programmable Robot

Component	*Industry Price*	*Quantity*	*Total*
Ultrasonic Distance Sensor	***$6.00***	***4***	***$24.00***
Robot Tank Chassis	***$30 - $40***	***1***	***$30 - $40***
Custom PCB	***$150 - $200***	***1***	***$150 - $200***
Resistors	***$0.01 - $0.10***	***50***	***$0.5 - $5***
Capacitors (Ceramic)	***$0.10***	***50***	***$5***
Multimeter	***$24.63***	***1***	***$24.63***
MSP430 Board	***$13.29 -$17.28***	***1***	***$13.29 - $17.28***
Raspberry Pi Camera	***$50***	***1***	***$50***
Misc. Electrical Components	***$150***		***$150***
Total			***$447.42 - $515.91***

Table 24: Budget with Robot Chassis

9.3 Conclusion

Throughout this paper we as group 9 have illustrated and detailed our autonomous sanitation robot. Starting with our executive summary and project description, we curtatied ideas and expanded on different possible projects before settling on this one. Now finalized on our project we expanded on what we got this robot to do and how this was demonstrated. These are our wants or deliverables. Along with these deliverables we also created and formed our overarching diagram that details all the necessary systems and who is responsible for it all. Essentially it is up to all four of us to hold each other accountable for work and helping each other work on things together so essentially no one person is responsible for a set thing.

The next steps were to study similar ideas and products in the market as well as research all different types of components and make a final selection. In the autonomous industry there are many products to study and understand how they operate. In our specific case our autonomous robot would resemble most closely that of a roomba, so by studying products of this line in the market was something key. We also went into extensive research into the types of sensors available and highlighted the many advantages and disadvantages of these sensors. There was also the study of the different types of battery selections needed as this is vital in powering our PCB board and running all the different components. Once all the research was done there was time to make the appropriate selections for components. Due to effects of COVID - 19 there were many delays in manufacturers shipping parts that played key roles in which components were chosen. Along with manufacturer factors there was also the performance of each component and system that needed to be further analyzed before any final decision was made.

Following the part selection and research was a quick overview of standards. Standards are an essential and overarching theme in modern day engineering. Many standards fall under the many categories of engineering such as project, design, time, financial, environmental and ethical. As each name states, the standard focuses on setting baselines and conditions given for their respective theme and ensuring an understanding between all in the field of engineering.

Lastly is the system breakdown and testing part of our paper. After breaking down roles and the essences of the project and conducting proper research and selection on components that last step is to implement a design. The design starts with the breakdown of the necessary subsystems of this autonomous robot. Each subsystem plays a crucial role that if not done properly may impact the entire robot as a whole. Once these subsystems were discussed the conversation moved into integration of these said subsystems. The integration is something that is crucial and typically occurs with the help of a PCB. It was necessary to research different PCB fabrications as well as vendors in preparations for our eventual design and ordering of a PCB. Lastly we needed to test all our individual components and systems to ensure they work as we want them to. This part is crucial as it was the first step in understanding how our project came together with physical hardware to test. It also opened the door to potential component replacement and or redesign of specific aspects if systems don’t work as originally intended.

In summary all the work that was put into the paper gave us a great blueprint in senior design 2 as well as our project as a whole. This paper has given us a great reference and is a great tool, but it is not without potential change as well. There is an understanding that changes be made and that is the understanding of this project and engineering. The discipline of engineering is always evolving as we as engineers must always be prepared for these changes and meet it with an open mind as well as an eager mind.

The Conclusion above was written after SD1 and with SD2 there were many changes that we have illustrated through edits made in this project. We learned alot about how to implement projects of this magnitude and how things do not go according to your original design. We dealt with many different experiences that hindered our progress and increased our cost.

A big aspect of design changes was our PCB design. We went through 3 boards in total in our attempts to make a working PCB. Our first board we ran into an issue of improper sized footprints. This meant that our components were not the right size for our PCB board. Once this was fixed we had our second board that went on to be soldered. When the board got soldered we learned that when power was applied we were not getting any power through our board. This was due to the lack of proper grounding and we needed to account and fix this with a grounding plane on our last design. The final board we were able to get had a proper grounding and we were able to get our PCB board to work in terms of power. However we could not upload the code because we had the wrong header for our to transfer our code from the computer to the MSP chip on the board. Along with the wrong header we did not have the right protection circrtrary applied for the MSP chip so to account for this we had created a filter to be able to upload the code onto the board.

There was this along with other design implementations like our platform. When we first designed our platform it had many flaws that affected our design and had to be fixed upon in the second platform. The second platform took away all the design restrictions that the first one had. Things like this can not be accounted for unless a physical prototype was developed

In all, SD2 really showed us that whenever design implementations are being made it is almost impossible to predict the necessary errors and restrictions that may occur. It is the part of an engineer to identify these mistakes when they occur, and figure out a solution that is both effective and cost friendly.

10.0 References

Adafruit. “Optomax Digital Liquid Level Sensor - LLC200D3SH-LLPK1.” *Optomax Digital Liquid Level Sensor - LLC200D3SH-LLPK1*, https://www.adafruit.com/product/3397?gclid=Cj0KCQjw24qHBhCnARIsAPbdtlI_zhM2_b7R-5CfZWygyeKJ6LNHFZL48zyJ7CG9Zz3CO480uxmmEKEaArsdEALw_wcB.

Adafruit. “Raspberry Pi 4 Model B - 4 GB RAM.” *Raspberry Pi 4 Model B - 4 GB RAM*, https://www.adafruit.com/product/4296#technical-details.

Adafruit. “US-100 Ultrasonic Distance Sensor - 3V or 5V Logic.” *US-100 Ultrasonic Distance Sensor*, https://www.adafruit.com/product/4019#description.

Airchr. “airchr Automatic Spray Bottle Electric Watering Can Automatically Water Sprayer Adjustable Mist.” *airchr Automatic Spray Bottle Electric Watering Can Automatically Water Sprayer Adjustable Mist*, https://www.amazon.com/dp/B088FWKTXG/?coliid=I3FWPFPTZNAJBC&colid=2OAKX940FV970&psc=1&ref_=lv_ov_lig_dp_it.

Atlassian. “Scrum.” *Scrum*, https://www.atlassian.com/agile/scrum.

Components 101. “Different Types of Batteries and Their Applications.” *Different Types of Batteries and Their Applications*, https://components101.com/articles/different-types-of-batteries-and-their-uses.

Devathon. “Top 10 Best Programming Language Rankings.” *Top 10 Best Programming Language Rankings*, https://medium.com/@devathon_/top-10-best-programming-language-rankings-e49a0def796c.

Digi-Key. “RASPBERRY PI 3 MODEL B+.” *RASPBERRY PI 3 MODEL B+*, https://www.digikey.com/en/products/detail/raspberry-pi/RASPBERRY-PI-3-MODEL-B/8571724.

Digi-Key. “SEN0101.” *SEN0101*, https://www.digikey.com/en/products/detail/dfrobot/SEN0101/6588457.

Digi-Key. “SEN-14032.” *SEN-14032*, https://www.digikey.com/en/products/detail/sparkfun-electronics/SEN-14032/6204382?utm_adgroup=Optical%20Sensors%20-%20Distance%20Measuring&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Product_Sensors%2C%20Transducers&utm_term=&utm_content=Optical.

Digi-Key. “SEN-15776.” *SEN-15776*, https://www.digikey.com/en/products/detail/sparkfun-electronics/SEN-15776/10650801.

Electronics Hub. “100+ Microcontroller Based Mini Projects Ideas.” *100+ Microcontroller Based Mini Projects Ideas for Engineering Students*, 1 8 2016, https://www.electronicshub.org/microcontroller-based-mini-projects-ideas/.

Embedded School. “Different types of Microcontroller Programming used in Embedded Systems.” *Types of Microcontroller*,

https://embeddedschool.in/different-types-of-microcontroller-programming-used-in-embedded-systems/. Accessed 2 8 2021.

Encyclopedia Britannica. “C.” *C*, https://www.britannica.com/technology/C-computer-programming-language.

Geek for Geeks. “Types of Software Testing.” *Types of Software Testing*, https://www.geeksforgeeks.org/types-software-testing/.

Geeks for Geeks. “The complete History of Java Programming Language.” *The complete History of Java Programming Language*, https://www.geeksforgeeks.org/the-complete-history-of-java-programming-language/.

Geeks for Geeks. “Differences between Procedural and Object Oriented Programming.” *Differences between Procedural and Object Oriented Programming*, https://www.geeksforgeeks.org/types-software-testing/.

Geeks for Geeks. “History of Python.” *History of Python*, https://www.geeksforgeeks.org/history-of-python/.

Geeks for Geeks. “How JVM Works – JVM Architecture?” *How JVM Works – JVM Architecture?*, https://www.geeksforgeeks.org/jvm-works-jvm-architecture/.

Gravitech. “Various LED 10mm Kit.” *Various LED 10mm Kit*, https://www.robotshop.com/en/various-led-10mm-kit-25pk.html.

IEEE. “IEEE 1625-2004 - IEEE Standard for Rechargeable Batteries for Portable Computing.” *IEEE 1625 - 2004*, https://standards.ieee.org/standard/1625-2004.html.

IPC. *IPC Standards*. 2020.

Kelechava, Brad. “ANSI C18.2M: Portable Rechargeable Batteries Specifications.” *ANSI C18.2M: Portable Rechargeable Batteries Specifications*, https://blog.ansi.org/2020/06/ansi-c182m-portable-rechargeable-cell-batteries/#gref.

LG. “Introducing CLOi the Autonomous UV-C Robot.” *CLOi UV-C Robot*, https://www.lg.com/us/business/commercial-displays/discover-cloi-uv-c-robot?gclid=CjwKCAjw_o-HBhAsEiwANqYhp5W4v83mLBtTOzj8f2ksmQ01Xn_r_MdNJx80LSAA0HQG6NC3qxHyvxoCOE8QAvD_BwE.

Lotz, Mary. “Mary Lotz.” *Waterfall vs. Agile*, https://www.seguetech.com/waterfall-vs-agile-methodology/.

MakerHawk. “https://www.robotshop.com/en/single-relay-board.html.” *https://www.robotshop.com/en/single-relay-board.html*, https://www.amazon.com/dp/B07GJ4GH67/?coliid=I3KZE29NKR4IC8&colid=2OAKX940FV970&psc=1&ref_=lv_ov_lig_dp_it.

MCL PCB. “Guide to IPC Standards for PCBs.” *PIC Standards for Printed Circuit Boards*, https://www.mclpcb.com/blog/ipc-standards-for-pcbs/.

Modern Robotics. “Modern Robotics RGB Color Beacon.” *Modern Robotics RGB Color Beacon*, https://www.robotshop.com/en/modern-robotics-rgb-color-beacon.html.

Mouser Electronics. “Parallax LaserPING 2m Rangefinder.” *LaserPING 2m Rangefinder - Parallax*, https://www.mouser.com/new/parallax/parallax-laser-ping/.

Newark. “GP2Y0A41SK0F.” *GP2Y0A41SK0F*, https://www.newark.com/sharp/gp2y0a41sk0f/sensor-distance-analog/dp/14N9318?gclid=CjwKCAjwz_WGBhA1EiwAUAxIcTYKL1CekG5ZyxxVTuXGKUzDKIRWG35BlZifA-HyKwwk669m_xMTpRoC4tgQAvD_BwE&mckv=sHkYbANS6_dc|pcrid|434136793437|plid||kword||match||slid||product|14N9318|pgr.

Newark. “OPB350W062Z.” *OPB350W062Z*, https://www.newark.com/tt-electronics-optek-technology/opb350w062z/liquid-level-sensor-1-16-tubing/dp/30M5162?gclid=CjwKCAjw_o-HBhAsEiwANqYhp7twrODj6jSDZSxs0iMruKxQW_ZsjQaARzYEHAVBlh9tbk4VR7nV5BoC4cEQAvD_BwE&mckv=sHkYbANS6_dc|pcrid|434136793437|plid||kword.

Octopart. “How To Choose A Microcontroller.” *How To Choose A Microcontroller - Blog - Octopart*, 2017, https://octopart.com/blog/archives/2017/08/how-to-choose-a-microcontroller. Accessed 2 8 2021.

Parallax. “Single Relay Board.” *Single Relay Board*, https://www.robotshop.com/en/single-relay-board.html.

Petra. “PetraTools Automatic Battery Sprayer - Easy-to-Use Sprayer with Comfortable Grip for Multi-Purpose Use.” *PetraTools Automatic Battery Sprayer - Easy-to-Use Sprayer with Comfortable Grip for Multi-Purpose Use*, https://www.amazon.com/gp/product/B082BF1S6V/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1.

Profis, Sharon. “How to maximize your Roomba's battery life.” *How to maximize your Roomba's battery life*, https://www.cnet.com/home/kitchen-and-household/how-to-maximize-your-roombas-battery-life/#:~:text=iRobot%20promises%20that%20the%20battery,sooner%20than%20you'd%20like.

Python. “General Python FAQ.” *General Python FAQ*, https://docs.python.org/3/faq/general.html.

Res, Environ. “Massive use of disinfectants against COVID-19 poses potential risks to urban wildlife.” *Massive use of disinfectants against COVID-19 poses potential risks to urban wildlife*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7346835/.

REXBETI. “REXBETI Ultimate-750 Paint Sprayer, High Power HVLP Home Electric Spray Gun.” *REXBETI Ultimate-750 Paint Sprayer, High Power HVLP Home Electric Spray Gun*, https://www.amazon.com/dp/B07DLR5FK2/?coliid=IN1C8PN9Q9KS2&colid=2OAKX940FV970&psc=1&ref_=lv_ov_lig_dp_it.

Seed Studio. “Types of Distance Sensors and How to Select One?” *Types of Distance Sensors and How to Select One?*,

https://www.seeedstudio.com/blog/2019/12/23/distance-sensors-types-and-selection-guide/.
Sparkfun. “5mm Adressable RGB LED.” *5mm Adressable RGB LED*, https://www.robotshop.com/en/5mm-adressable-rgb-led-5pk.html.
Sparkfun. “PCB Basics.” *PCB Basics*, https://learn.sparkfun.com/tutorials/pcb-basics/all. Accessed 2 8 2021.
Sun Microsystems. *Java Code Conventions*. Sun Microsystems, 1997.
Texas Instruments. “TIRSLK-EVM.” *TIRSLK-EVM Evaluation Board*, https://www.ti.com/tool/TIRSLK-EVM?utm_source=google&utm_medium=cpc&utm_campaign=corp-uni-null-Microcontroller_Projects-cpc-evm-google-wwe&utm_content=Microcontroller_Projects&ds_k=%7b_dssearchterm%7d&DCM=yes&gclsrc=aw.ds&gclid=CjwKCAjwr56IBhAvEiwA1fuqGg-P.
Thomas Net. “Top USA and International PCB Manufacturers.” *Top USA and International PCB Manufacturers*, https://www.thomasnet.com/articles/top-suppliers/pcb-manufacturers-suppliers/.
Usman HM, Abdulkadir N, Gani M, et al. Bacterial pigments and its significance. MOJ Bioequiv Availab. 2017;4(3):285-288. DOI https://medcraveonline.com/MOJBB/bacterial-pigments-and-its-significance.html