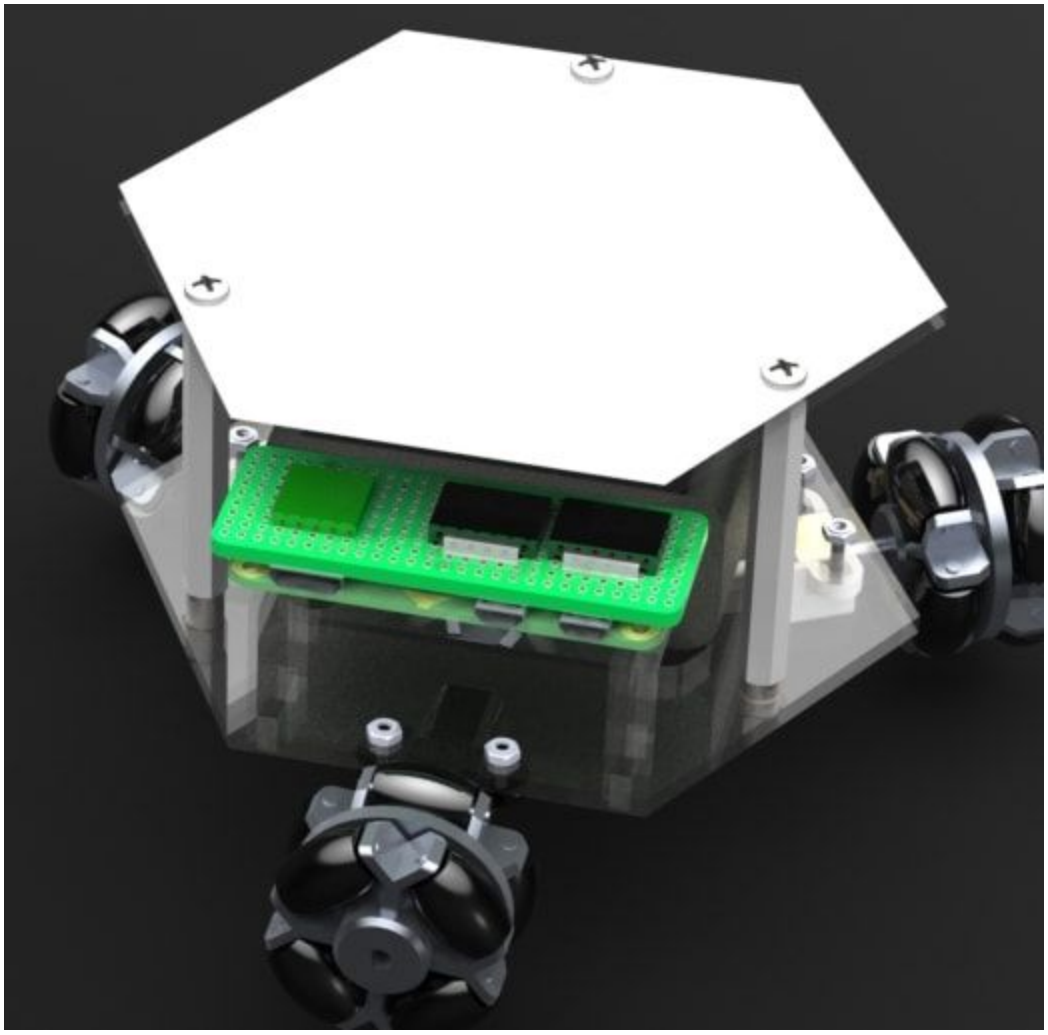


Team 9 Robo-Squad: Final Report

Alex Colton, Julia Di, Bailey Fryer, Chuck Poklikuha, Connie Zhang

Table of Contents

Executive Summary

Robo-squad is a swarm of four small mobile holonomic robots that execute dynamic coordinated movement within a controlled arena through an open feedback loop. With computer vision, an overhead camera records the robots' locations in realtime, and sends them to a computer. The computer acts as a centralized processing node to calculate the swarm controls, and wirelessly sends each robot velocity commands. This project explores the motion and applications of a small robotic swarm. These robots successfully work together to locate themselves and move in formation, which can be used for tasks like navigating obstacles during terrain exploration or moving and picking up objects — tasks that one robot cannot accomplish alone.

Introduction

Robotic swarms is a growing field of research in response to traditional robotics. Traditional robotics focus heavily on developing a specialized robot for a particular task, meaning long lead times and robots that can only accomplish a very narrow set of tasks, but are typically optimized to perform them well. Swarm robots is focused on developing small, homogenous robots that use collaborative systems to work together and accomplish tasks. This creates a very extensible system that allows robots to be added and subtracted with very little effect on the swarm as a whole, similar to an ant colony.

This type of robot is perfect for unknown environments. While a standard mobile robot would be ruined by losing a part, a swarm of robots has a higher chance of returning as the swarm is still able to function with lost members. Such environments would be search and rescue, disaster relief, mining, space exploration and more. Robots involved in unknown environments typically have onboard sensors, which do make them more expensive. The alternative is having an overhead camera that tracks the robots in a master-slave relationship. An exterior computer is responsible for the movement of the robots, which is used in indoor and structured environments such as warehouses and hospitals.

We opted to build the overhead camera framework because it offers the cheapest version of robots, a structured project which to design, and the opportunity to learn several new skills including ROS, Computer Vision, and networking. Our goal was to provide a proof of concept for swarm robots using completely open source materials and within our budget. To this end, we planned to create four robots, use a webcam, and a desktop computer to create our swarm dynamics. Our code was written to be extensible, and account for missing robots, to prove that such a system would be flexible enough for a wide variety of real-world applications.

Background Information

Our concept began as a robot swarm wherein several robots work together to solve a problem. We considered several ideas as to what that problem would be, including crossing a chasm, climbing stairs, and moving objects around. We decided to work on picking up and transporting an object, which informed our design choices, though later we simplified our task to coordinated motion.

We first had to decide the mechanism by which the robots will be picking up the object: active or passive? We decided the robots would pick up the object passively in order to reduce the number of motors and moving parts we needed to cram in a small robot chassis. The robots will be wedge-shaped, like a door stopper, and ram themselves into the object and force the object up the wedge's slope.

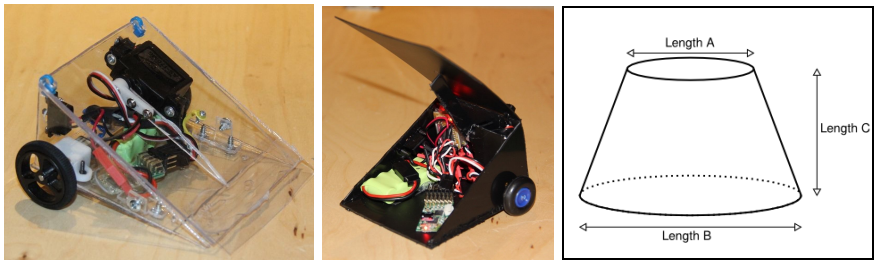


Fig. 1: Previous design ideas. The left two show a wedge-shaped robot that actively lifts the wedge. The rightmost shows a saucer shape that would passively pick up objects without regard for orientation of the robot.

Our design presented in Design Review I was a rectangular wedge shape, shown below. Each wheel had its own motor for turning, by having one wheel rotate while the other was still.

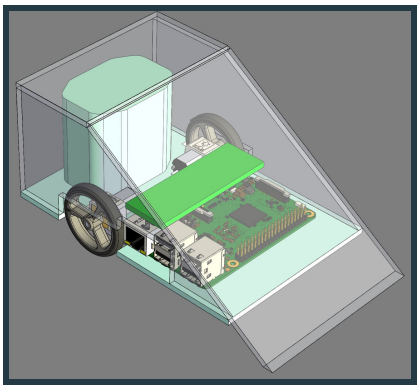


Fig. 2: Initial design with two powered wheels in back and two caster wheels in front.

However, we realized that we would have a controls issue when the robots picked up an object to move it. Without holonomic drive, rotating the robots underneath an object in order to navigate

would be nearly impossible. With holonomic drive, rotating will no longer be necessary since the robots will be able to move in any direction regardless of their orientation, making navigation with the object much simpler.

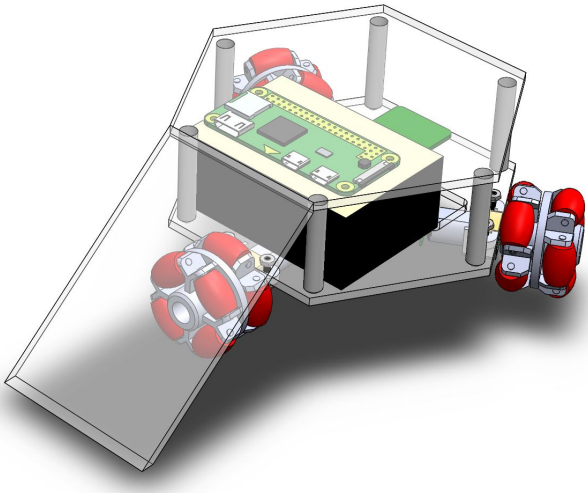


Fig. 3: Previous design with three-wheeled holonomic drive.

However, due to complexity, we eventually scrapped the idea of picking up an object entirely. Instead, we settled on the idea of having four robots that can move independently and together. An overhead camera, attached to the arena that the robots will drive in, feeds location data to the remote computer through computer vision. The robots communicate through onboard WiFi chips in the Raspberry Pi Zero W to the remote computer. With the swarming capabilities, our robots are a platform for several multi-agent robot algorithms (e.g. collision avoidance). Furthermore, with our swarm robot platform, we can eventually push around objects to achieve our original task of transportation.

The project has the following four levels:

1. Robots placed in arena and moving in a formation.
2. Robots can split apart and come back together again.
3. Robots can implement a multi-agent robot algorithm (collision avoidance).
4. Robots can successfully push an object from one corner to another.

Literature Search Results

Our literature search was based on our previous concept of robots picking up objects.

1. Internet Search

We found many hobbyist websites describing how to construct simple robot swarms using off-the-shelf electronics and other cheap components. The two most useful were the Bristlebot

Swarm tutorial [1] and the Leader/Follower Arduino Robot tutorial [2]. This defends the feasibility of our communication systems, as well as the feasibility of swarm robots on a budget.

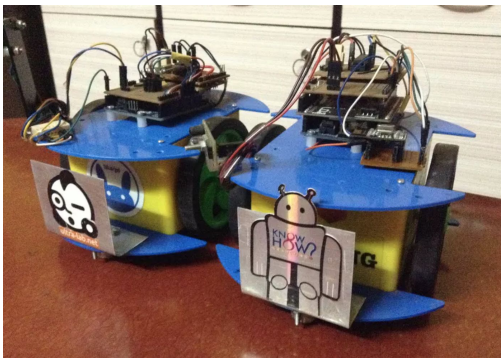


Fig. 4: A hobbyist multirobot system as pictured in tutorial [2].

2. Patent Search

We researched patents on omniwheels and holonomic drivetrains, and we found several in order to defend our triangular holonomic drivetrain [3] [4] [5].

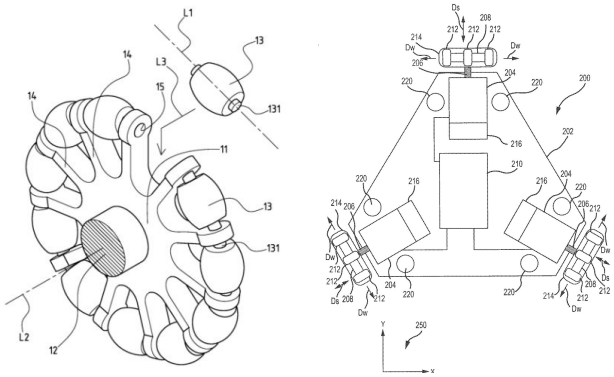


Fig. 5: Pictures from the relevant patents [3] [4].

3. Journal Search

We looked into scientific literature and the most relevant journal paper described a wedge-shaped robot swarm that could lift up objects and transport them [6]. This justified our wedge-shaped design choice, as well as the idea itself.

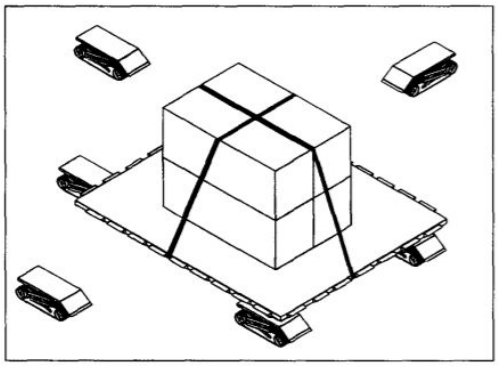


Fig. 6: Concept sketch from "Design of the "army-ant" cooperative lifting robot" [6].

Design Details

Analysis

Much of our engineering analysis was done with the goal of having a robot that was capable of lifting an object. For comprehensiveness, that analysis is included in this report, since many of our initial design decisions were based on that goal. However, it should be noted that our goals shifted mid-way through the semester once it became clear how complex our original project was. **Though our goal eventually shifted, there were many vestiges of our early design choices that were kept because they worked well, a testimony to the flexibility and mechanical stability of our original design for swarm robotics.**

1. Materials Analysis

We decided on the materials for the chassis by prioritizing being lightweight, easily accessible, easily manufacturable, and cheap. Therefore we choose to make our robot chassis and wedge out of acrylic instead of sheet metal. Originally, we also chose acrylic in order to fashion our wedge so that we could bulk buy acrylic material for the entire robot. The wedge should have had a low coefficient of friction so the object can easily slide up the slope, and acrylic has a static coefficient of friction of 0.20.

2. Shape Analysis

We chose to do a holonomic drive in order to solve our original controls issue of movement with an object atop the robot, as stated in the Concept Selection section. Most holonomic drives have four wheels, but one can do three-wheeled holonomic drive if all three wheels are arranged in an equilateral triangle. We chose to do this in order to save space and cost (one fewer wheel and motor).

We considered two shapes, an equilateral triangle and a hexagon, to accommodate the three-wheeled holonomic drive. To determine which shape would be optimal we did an area analysis. If we take an equilateral triangle and hexagon both with perimeters equaling 12x we can prove that the hexagon base will have more area for electronics. First, we solve for the area of the triangle. Perimeter = 12x means each side is 4x and therefore the height of the triangle is $\sqrt{(4x)^2 - (2x)^2}$. From this we know the area of the triangle is $(\frac{1}{2}) * 4x * \sqrt{(4x)^2 - (2x)^2} = 4x^2\sqrt{3}$. Next we calculate the hexagon's area using the same perimeter of 12x, meaning each side is 2x. If we break up the hexagon

into six equilateral triangles we can compute the area of each triangle and simply multiply by six to get the area. Using the same process we just did we get that the height of each triangle inside the hexagon is $\sqrt{(2x)^2-(x)^2}$. From this we get the area of each triangle is $(\frac{1}{2}) * 2x * \sqrt{(2x)^2-(x)^2} = x^2\sqrt{3}$ and therefore the area of the hexagon is $6x^2\sqrt{3}$. Comparing the areas we can see that the area of the hexagon is 50% more, $\frac{6x^2\sqrt{3}}{4x^2\sqrt{3}} = \frac{3}{2}$. Therefore, we chose the hexagonal shape in order to maximize the internal area that we could use to store other electronics.

3. Force Analysis on Wedge Angle

This analysis is no longer relevant to our final design, but is included in the final report for the sake of comprehensiveness.

We performed a force analysis test to help us determine the ideal angle to used for the wedge. Using the equation $F = m * g * sin(\theta) + \mu * m * g * cos(\theta)$, we created a MATLAB plot to determine which angle would be ideal for our ramp. The plot is shown below. We wanted to keep the angle steep, in order to shorten the overall length of our robot. After testing multiple weights and angles, we decided that 40° would be optimal because the plot is near linear until about 40°, meaning we only marginally reduce force by reducing the angle. In the below graph we are assuming an object weight of 1kg.

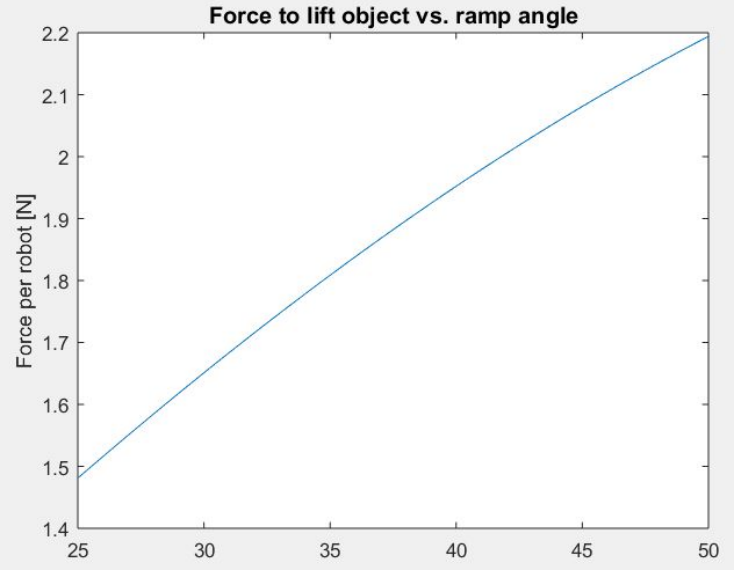


Fig. 7: Plot of force required to vs wedge incline (degrees).

4. Motor/Torque Analysis

To determine which motor we needed for our wheels we began with a torque analysis with the robot driving on a flat surface. This is going to be the amount of torque the motor is under during most of our demo. Using $T = (\frac{100}{e}) * \frac{(a+g*sin(\theta))*M*R}{N}$ where e is efficiency, a is acceleration, g is gravity, theta is the surface angle, M is total mass, R is radius of our wheels, and N is the number of wheels, we get that the required torque on each motor needed to drive around on a flat surface is 5.56 oz-in. From this torque value, we needed a motor that had a stall torque of at least 4 times this, because we did not want a DC motor running continuously at more than 25% of its stall torque (common robotics wisdom). We also wanted to make sure our robots were moving at a decent speed throughout so we decided on an rpm of at least 100 while the motors were under this torque. As seen in Fig. 7, at 5.56 oz-in of torque the DC motor we have chosen will spinn at roughly 110 rpm which converts to 0.72 ft/s, for a wheel with a radius of 0.75 in.

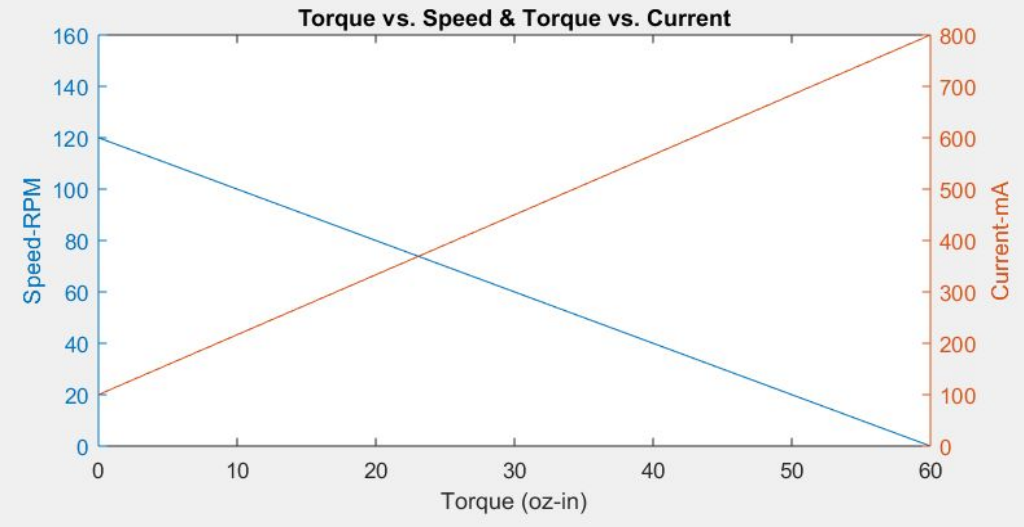


Fig. 8: Torque vs. Speed and Torque vs. Current draw

Lastly, we also wanted to be operating our motor at a torque where our motor was efficient. Using the following table we determined the motor we selected would allow us to operate in a range where the motor was efficient.

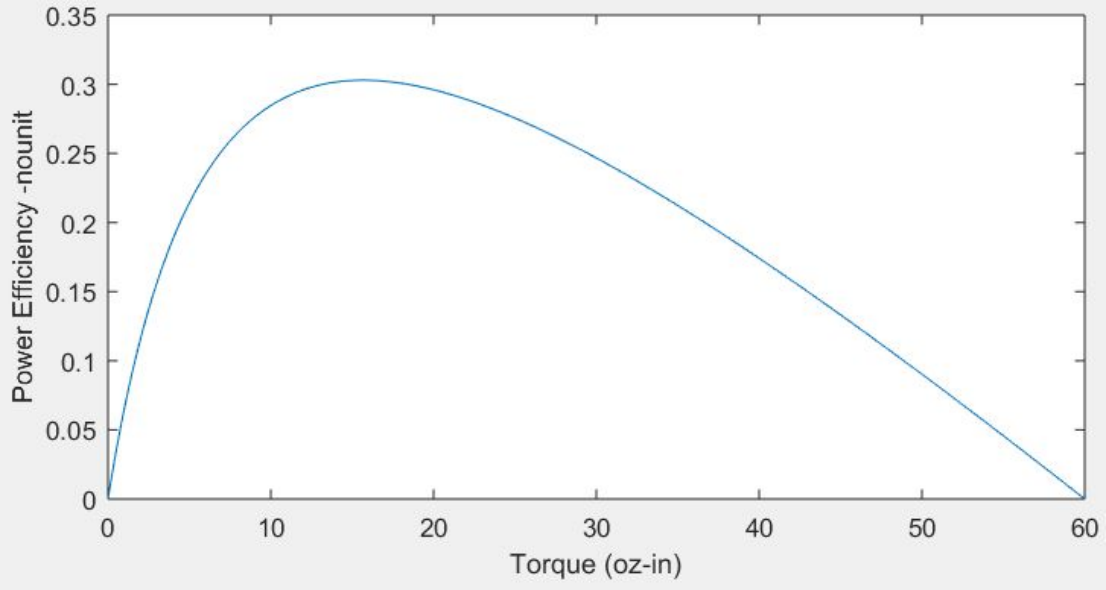


Fig. 9 : Efficiency vs. Torque

Using these requirements, we choose the smallest motor possible to fit our demands. The motor we selected gives us the amount of torque we need, moves fast enough, and when operating at the given torque of 5.56 oz-in, has an efficiency of 22% which is near its peak efficiency of 30%.

5a. Power Analysis

Having decided upon the motors and the required torque output, we looked for the smallest possible battery that could provide us enough mAh to run our motors for 2 hours of runtime. We chose a small battery that fits in our robot dimensions, providing 2000mAh. Below is our calculation of the maximum power draws of our different electrical components.

Item	**Power Consumption (mA)**
DC Motor:	150
Raspberry Pi Zero:	160
Step-Down Voltage Regulator:	2
Motor Driver Chip:	84
Total usage (hours):	**2.873563218**

Table 1: Power consumption analysis.

5b. Dimensioning the robot

Likewise, we dimensioned our robot based on two factors: a) what was subjectively small (no greater than 6" or so), and b) what was physically feasible given our parts. The battery was the largest component that we had to accommodate. Since we had already set an angle for the wedge ramp (see 2. Force Analysis on Wedge Angle), we could then calculate how large our base needed to be to accommodate both the battery and motors, and the desired wedge angle.

6. Swarm Analysis

In order to manage multiple robots in one arena, we turned to ROS, the robot operating system. This is a middleware, i.e. a collection of frameworks, that uses a subscriber/publisher method of communication. Using ROS frameworks, we are able to simulate, and control, all of our robots. We are using an existing framework developed in 2016 called the micros_swarm_framework, a package that is specifically made to manage swarms of robot. Although not perfect for our needs (as our robots are slightly smarter than they anticipated with this framework), it is a proof of concept that we can, in fact, manage multiple robots with relative ease. Preliminary experiments are shown below.

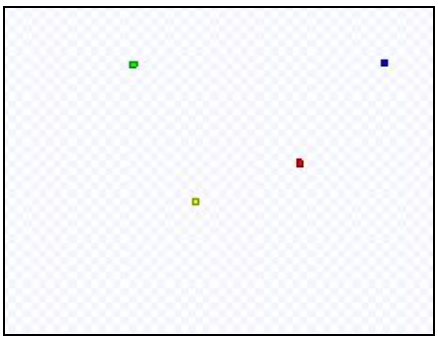


Fig. 9a: Movement as a group.

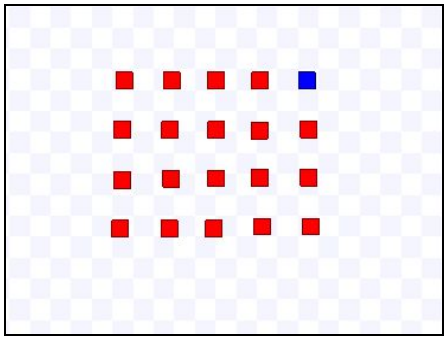


Fig. 9b: Movement in response to neighbors.

In implementation we eventually decided to custom-write our own control software for our robots. The available micros_swarm_framework was, in addition to being poorly documented and hard to use, also intended for drones which have slightly different dynamics than on-ground robots. Our custom software will be detailed in the Experiment & Test section.

However, our analysis in ROS did prove useful by providing us insights into swarming behavior.

7. Camera Analysis

Our arena is 48" x 48". We need a camera that has a high enough resolution such that each pixel on the camera maps to a certain projected square on the arena. For good resolution of our robots, we assumed that a good resolution would be 0.25"/pixel at a maximum. Using that assumption, we

then calculated the projection area based on possible field of view (FOV) and pixel resolutions of different cameras.

Our calculation is as follows:
From a side view, the camera projection is an isosceles triangle, where h is the height of the camera and theta is the horizontal FOV as listed in camera lens spec sheets.

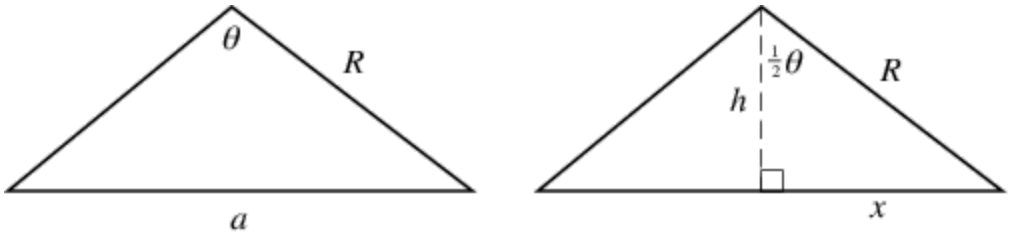


Fig. 10: Diagram of resolution calculation.

We want to solve for a, the horizontal projection of the camera in inches at a given height. Our hypotenuse $R = \frac{h}{cos(\frac{\theta}{2})}$, and $x = R * sin(\frac{\theta}{2})$. Therefore, $a = 2 * h * tan(\frac{\theta}{2})$. From there, we can divide by the number of pixels horizontally to get the distance per pixel the camera sees.

With the camera we choose, we have 0.11 in/pixel at a height of 4ft. The height was chosen because that satisfies our resolution requirements while also minimizing the amount of extra material needed to construct the mount, but still being far enough above the robots to clearly see them without obstructing the judges' view of the demo.

Design Overview

Below are some illustrations of our control architecture, arena design, and robot design. See the Appendix for detailed drawings of the robot and arena, and see the Notes subsection for our cost analysis.

Control Architecture: Open Feedback Loop

The four robots communicate via the Raspberry Pi Zero W microcontroller's WiFi chip to a remote laptop communicating over the same WiFi network. The remote laptop runs computer vision algorithms in realtime on frames from the overhead camera. We then localize and coordinate the robots through the realtime open feedback loop.

Fig. 11: Simple block diagram describing the architecture of our feedback loop.

This control scheme has to be implemented in software, making our project extremely interdisciplinary. We describe each link in detail in the Experiment & Test Results section.

Arena

We created a controlled environment for our robots to interact in which we call the arena. In the arena, the camera is mounted overhead. The floor is also created with tiled mats of uniform black color so that the computer vision can easily distinguish robot from floor. Finally, we have a black railing (it was painted black so that the computer vision does not confuse rail with robot) of aluminum as a border for our arena and to mount the camera to.

Figure 12: CAD model of robots in arena.

Robot

We tried to emphasize **modularity** and **simplicity** in our robot design. All electronics were easy to access because they were each laid in their own compartment. The top lid of the robot was magnetically attached and could be snapped off when anyone needed to access the internal electronics.

After finalizing the design on paper as detailed in the Background Information section and the Analysis section, we built the first version of our robot.

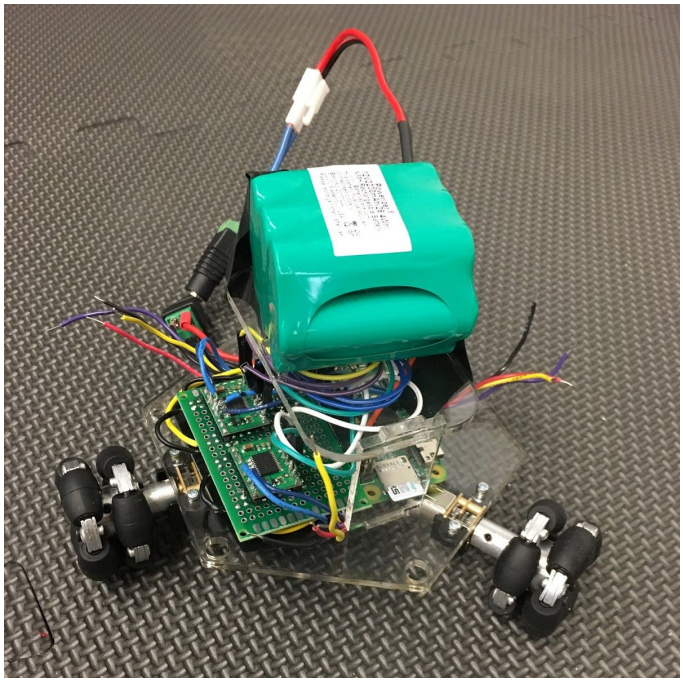


Figure 13: First prototype robot, which was presented at the Assembly Review. Not shown in this image was the lid with magnetic attachment clasps.

This prototype robot drove well. However, for the electronics we quickly realized the need for proper wire management. Soldering the custom protoboard connections for the electrical backend took over 6 hours due to the number of connections that had to be made. Furthermore, despite all attempts to make wiring as simple as possible, we still had dozens of wires crisscrossing everywhere onto the Pi. This would not be scalable since we had to build four robots in total.

We then made created a version 2.0 with a custom PCB to simplify the wiring dramatically.

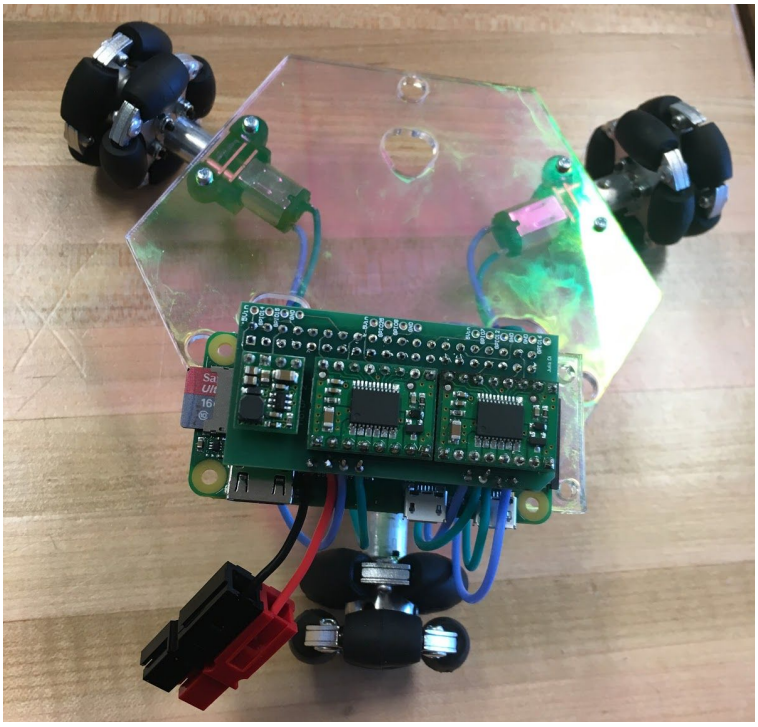


Figure 14: Version 2.0 with dramatically simple wiring.

The electrical schematics and board layout for the custom PCB is included in the appropriately named Appendix section.

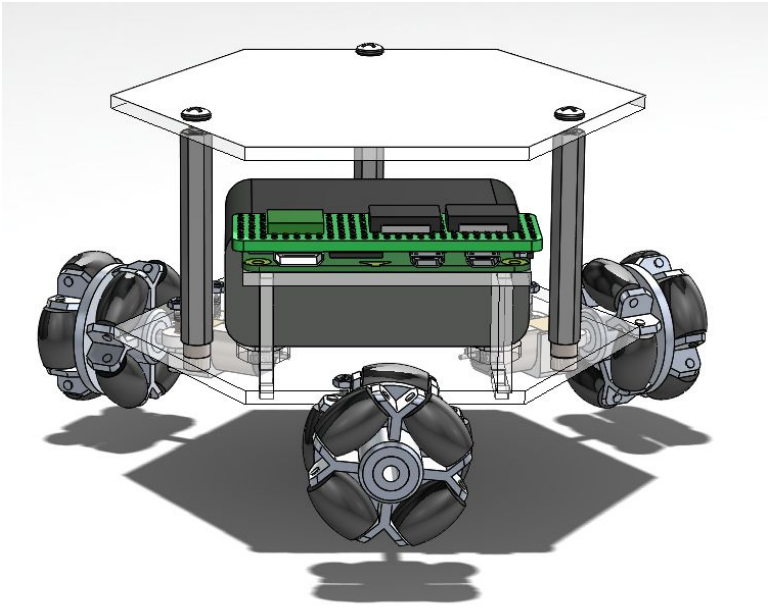


Figure 15: CAD model of Final Robot Design

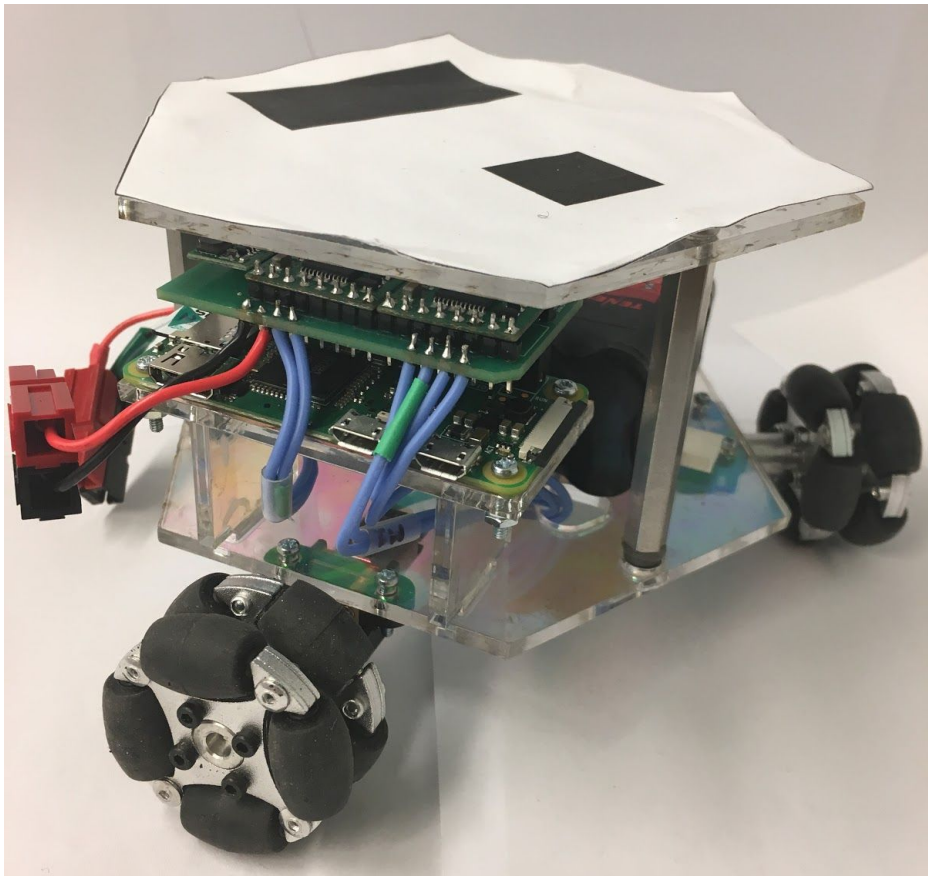


Fig. 16: Final fabricated robot

Experiments & Test Results

Our project was extremely **interdisciplinary**. In the design section we have discussed the hardware components, namely our mechanical and electrical design considerations. However, we have a substantial computer science component to our project for testing our robots as a swarm robotics platform. As discussed in the Design section, our control feedback loop is done through software, and had to be verified in software. All code can be found in the Appendix section. Videos of results can be found on our website.

Please note that we did not have the time to run all the controlled experiments we desired. This will be further discussed in the Conclusion section as part of Future Work, but the metrics we aimed to gather included items like robot driving accuracy.

Control Loop: Camera to Computer Link

The first link in our control loop architecture is the computer vision link. In plain terms, we need to make sure that our camera can accurately see all four robots, distinguish what is robot from what is not robot, compute each robot's location, and send that information to the computer.

Calibration of Camera to Arena. To measure whether our camera could accurately see all four robots, we simply printed the realtime output of the camera to the screen and visually verified whether the camera field of view was aligned with the arena.

Identification of Robots. Each robot is lid has a white piece of paper, and teh arena mat is a dark black color. To distinguish robot from non-robot, we took the camera output and binarized it according to a threshold value.The resulting binarized image a black-and-white version of the camera output.

We used a Harr-Cascade algorithm to classify the contours of the robot (which would be a transition from black to white).

Each robot lid has a number of black squares on top of it for identification, as well as a black rectangle to mark the front of each robot. Robot 1 has one black square, Robot 2 has two black squares, and so on. By identifying the number of shapes inside each robot lid, we could identify each robot (again by using Harr-Cascade algorithms to find the transitions from white to black).

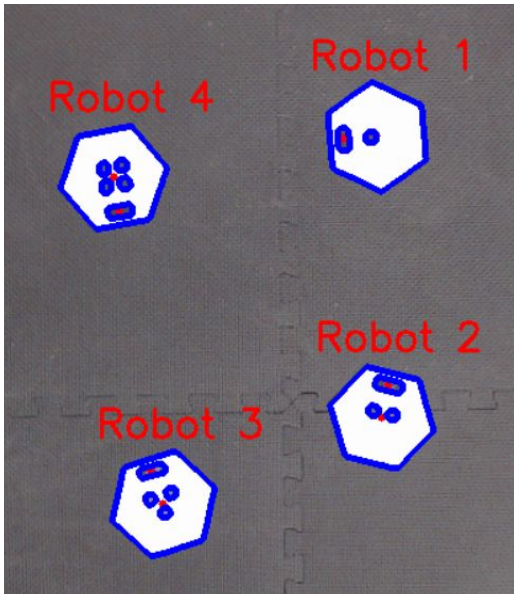


Fig. 17: Testing the camera computer vision

Identifying Robot Location and Orientation. After identifying each robot, we found the necessary information to be sent to ROS. The information sent was the centroid of the robot body, the centroid of the front contour (a black rectangle that marked the front of the robot), and the angle of the centroid of the front contour relative to the centroid of body.

The front contour was found by finding the contour inside the body with the largest area. One thing that had to be taken into account was that as the robots moved further from the center of the arena, the camera saw more of the side of the robot, since it was not looking directly overhead anymore. This meant that what the camera thought was the center of the robot (the center of the lid), was no longer accurate. We cared about the center of the robot based on the center of the body, and called this camera perspective offset between the center of the top lid and the bottom chassis the "translation problem". Since we cared about the centroids at the bottom, but we were finding the centroids at the top, we used a gradient to account for this offset and give us the revised accurate centroid regardless of position in the arena.

Once all the centroids were found, we calculated their locations based on which pixel they were in the image. We then printed those locations to the screen and verified in-person whether they were accurate. (Since we know the arena is 3' x 4' and that corresponded to a ratio-ed 600 x 800 px image, we could easily calculate the location in real life of the robot vs. the camera.)

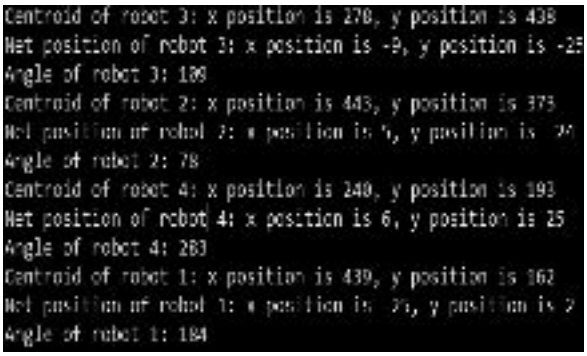


Fig. 18: We verified whether the camera accurately detected all four robots, and printed their locations correctly to the screen.

Control Loop: Computer to Robot Link

The second link in our control loop architecture is the communication from computer to robot. This is done through socket programming. We had each robot create a server that would constantly listen on port 5020. The computer created a client that would make a connection request to each robot's IP address and bind to the specific port number. There were several steps that we had to individually test in order to make this connection work.

Robot-side Communication. The first step was to set up each robot with a unique a static IP address so that the connection request could be hardcoded (there are thousands of IP addresses otherwise, and it would take too long to look through all of them, if the IP address weren't hardcoded).

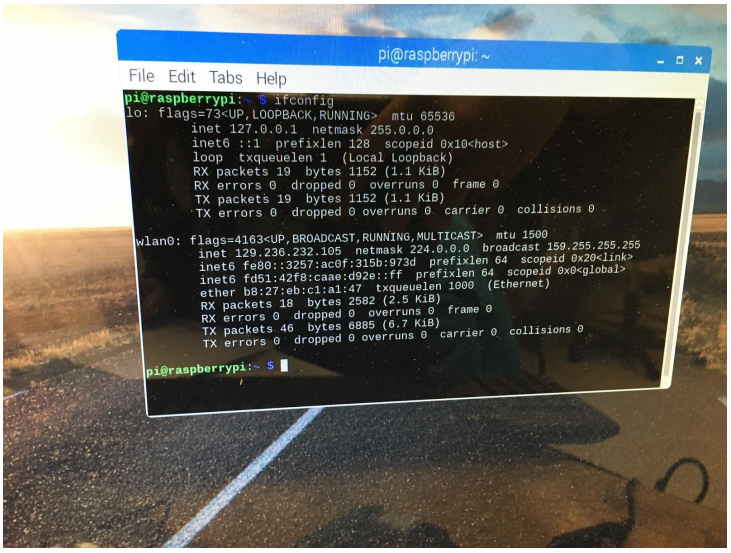


Fig.19: Image from initial testing of hardcoded IP address.

Our testing setup of the hardcoded IP address was to see if, with the hardcoded IP address, the Raspberry Pi could connect to the Internet. It could not.

The reason why, we figured out, was because there are several other factors that need to be hardcoded as well, with each. For example, the name_server and the domain_server both had to have the correct mask values. (Separately, we also programmed the robot to launch the server code upon startup).

Communication Setup. Based on the issues we had found when trying to set up the robot communication, we decided to create our own local area network. We obtained our own private router, which would allow us to statically set those additional parameters such as name_server and domain_server as well.

Simultaneously, we coded client software on the computer that would make a connection to the robot.

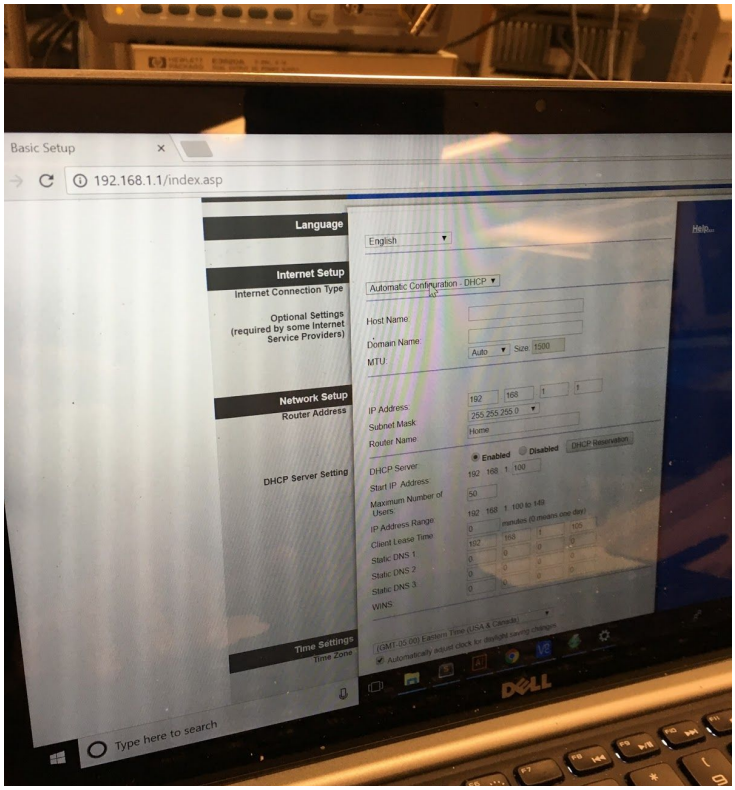


Fig. 20: Image from setting up private router.

Verification. We tested our communications by pinging 64 byte packets of data from the computer to the robot to verify that the socket connection had been made. A solid connection meant that there were no dropped packets of data.

Software Experiments

Our robots serve as a swarm robot hardware platform that a multitude of software experiments can run on. For the purpose of our senior design project, the software implementation we choose was collision avoidance and autonomous holonomic drive via centralized control. Because our project is mainly a hardware platform for software, the experimental testing of the hardware has to be done through the software implementation.

Holonomic Drive

ROS Framework. At first we tried to use the ROS micros_swarm_framework to control our robots. After much time spent, we realized that there were a few fundamental issues with the ROS framework that would prevent us from being easily able to use it. The first was that the code itself

was messy and undocumented; it was difficult to read and even after contacting the original author we still were making very slow progress. The second reason was that the framework was written specifically with drones in mind, and a lot of its functionality was in the 3D space. This is fine, except we had chosen to work with holonomic drive, which requires some non-intuitive controls to drive (when compared to a simple two-wheeled driving robot).

Custom Software. We then switched to creating our own custom holonomic drive software based on vector math. Our robots may move in any direction while simultaneously controlling rotational speed. This is true because the omni-wheels have rollers that allow them to freely roll sideways but control the motion in the direction the wheel is pointing. The following diagram and analysis illustrates how a desired motion vector and rotational velocity can be resolved into the wheel velocities that will create the desired motion.

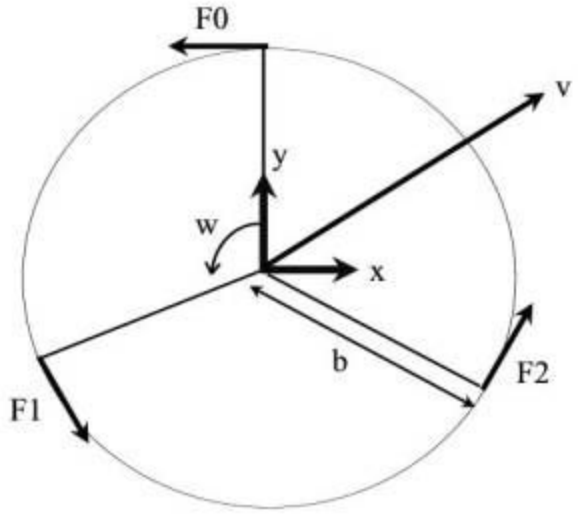


Fig. 21: Diagram detailing force vectors on each wheel for kiwi holonomic drive.

r -wheel radius
F0, F1, F2 -unit direction vectors
v - desired body velocity expressed in body coordinate frame
w - angular velocity
b - wheel baseline
v0, v1, v2 - wheel linear velocities
w0, w1, w2 - wheel angular velocities
n - wheel number
pn - velocity of the body at a given wheel n
F0 = [-1, 0]
F1 = [1/2, -sqrt(3)/2]

F2 = [1/2, sqrt(3)/2]

In our control program, this analysis is incorporated by creating a drive function which drives the robot in a direction specified by a given vector V, while rotating the robot at an angular speed w.

Verification. To verify that our holonomic drive calculations were correct, we wrote software that would take, as input, all the data from the computer vision portion of our control loop and interpret that data into a simulation of each robot. We then constructed a simulation GUI on the computer to display what the camera detected (each robot's location in the arena and their orientation). The user can click on a point in the simulation area to direct the robots to drive to it.

We verified accuracy by determining whether the robot drove to the desired location accurately.

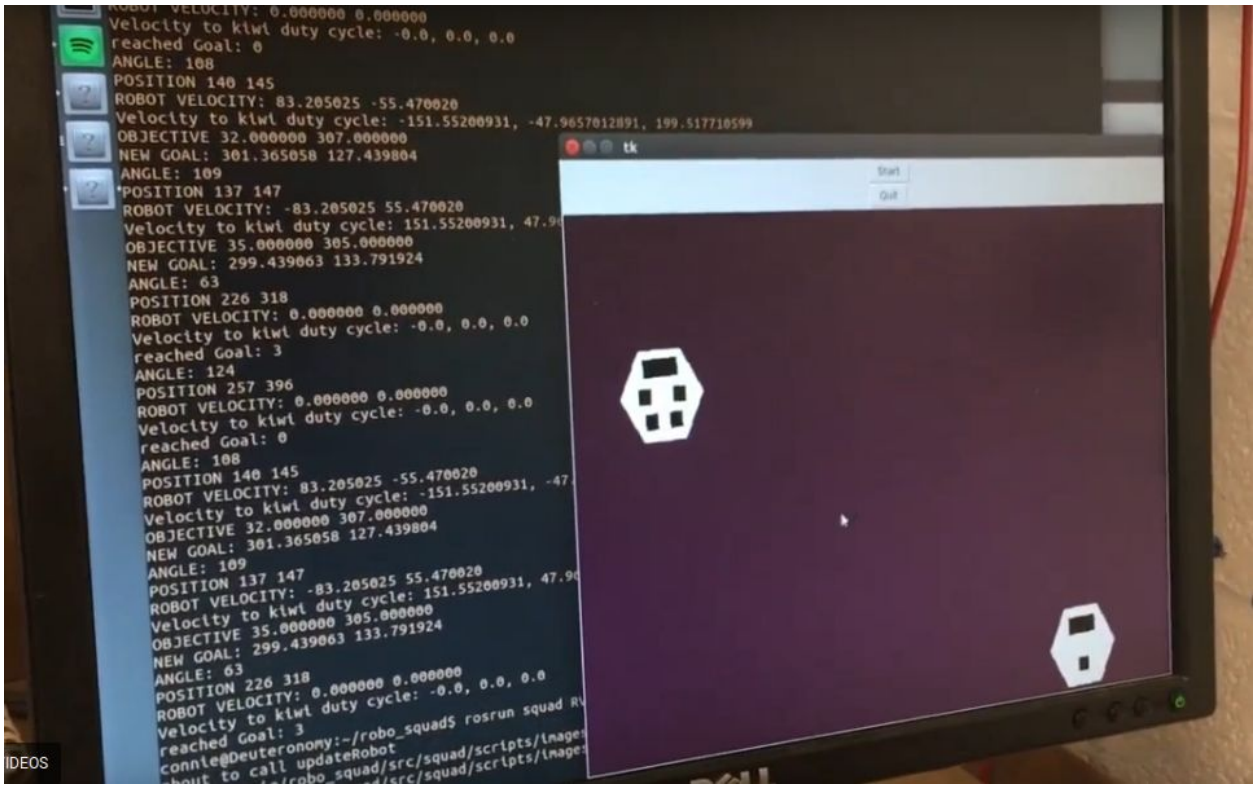


Fig. 22: Screenshot of the simulation parsing in computer vision data, outputting those locations to the screen, and then calculating the correct velocity vector calculations.

Collision Avoidance

Collision avoidance is a common problem found in nearly every type of robot. The simplest type is a stationary obstacle that the robot navigates around. In swarms, however, the obstacles are the other

robots in the swarm. Moreover, the other robots are also moving, becoming what is known as "velocity obstacles", making the collision avoidance problem much harder.

It is a fairly complicated math problem, so we turned to existing libraries that have already done much of the heavy lifting. One called Optimal Reciprocal Collision Avoidance (ORCA) fit our needs well, and is made specifically to simulate the movement and collision avoidance of large numbers of moving objects, i.e. perfect for swarms. Every step of the simulation the velocity vectors of the objects, and a number of neighbors (that the user sets), is calculated. Through a linear transformation, it can calculate whether a certain velocity will collide in a given amount of time. From that, it generates an area of vectors that would cause a collision, and an area of vectors that will guarantee no collisions within a certain amount of time. It picks the closest velocity to its desired velocity, and adjusts the robot's course. For each step, it recalculates this simulation for each robot. This guarantees a path for each robot that will cause no collision with any other robot. This works well on our small scale of four robots in particular, keeping the calculations fast and easy. The math is covered in more detail in their paper.

To implement this, we used their python wrapper class to use the simulation, then translated the results of the simulation onto our Graphic User Interface (GUI). This allowed us to track the simulated path of the robots on a custom designed screen. Later, it also allowed us to track the robots in real time onscreen, visualizing our feedback loop.

Conclusion

Our robot design is optimized for our goal of coordinated motion. We have implemented the simplest possible holonomic drive system in order to have translation in all directions without constraining ourselves to a certain turn radius or starting orientation. We have created a modular robot design that allows for components to be easily separated and swapped when broken (e.g. a magnetic snap-on top). We have custom-designed our electronics in order to simplify the electrical wiring (no more wires everywhere!). Our design is well within the budget constraints, and we have mathematical proof to defend how our robot works and why we have chosen our design.

Future Work. However, if we were to start over again, the biggest change we would make is to have started earlier. We underestimated the lead time for shipping items, which meant that we did not receive the parts for our first order until mid-March. This severely pushed back our timeline, and in some ways, irrecoverably pushed back our timeline.

Because the hardware development finished very late (mid-April), we had less time than desired to finish the software. Since the software is what makes our robots special (since we were building a

swarm platform, we wanted to show a variety of use cases), this was less than ideal. The little time we had to polish our software meant that we had little time to do controlled experiments on robot accuracy, as mentioned in our Experiment Results section. However, we were able to qualitatively verify the accuracy of our robots through software implementation (as discussed in the Experiments section).

We are quite proud of the progress we made this semester. Robo-Squad is a non-trivial project with many moving parts, requiring specialty in math/physics (holonomic drive control), mechanical engineering (custom robot design), electrical engineering (custom electronics), and computer science (custom software and computer vision). To see a video of our results, please check out our website.

Acknowledgements

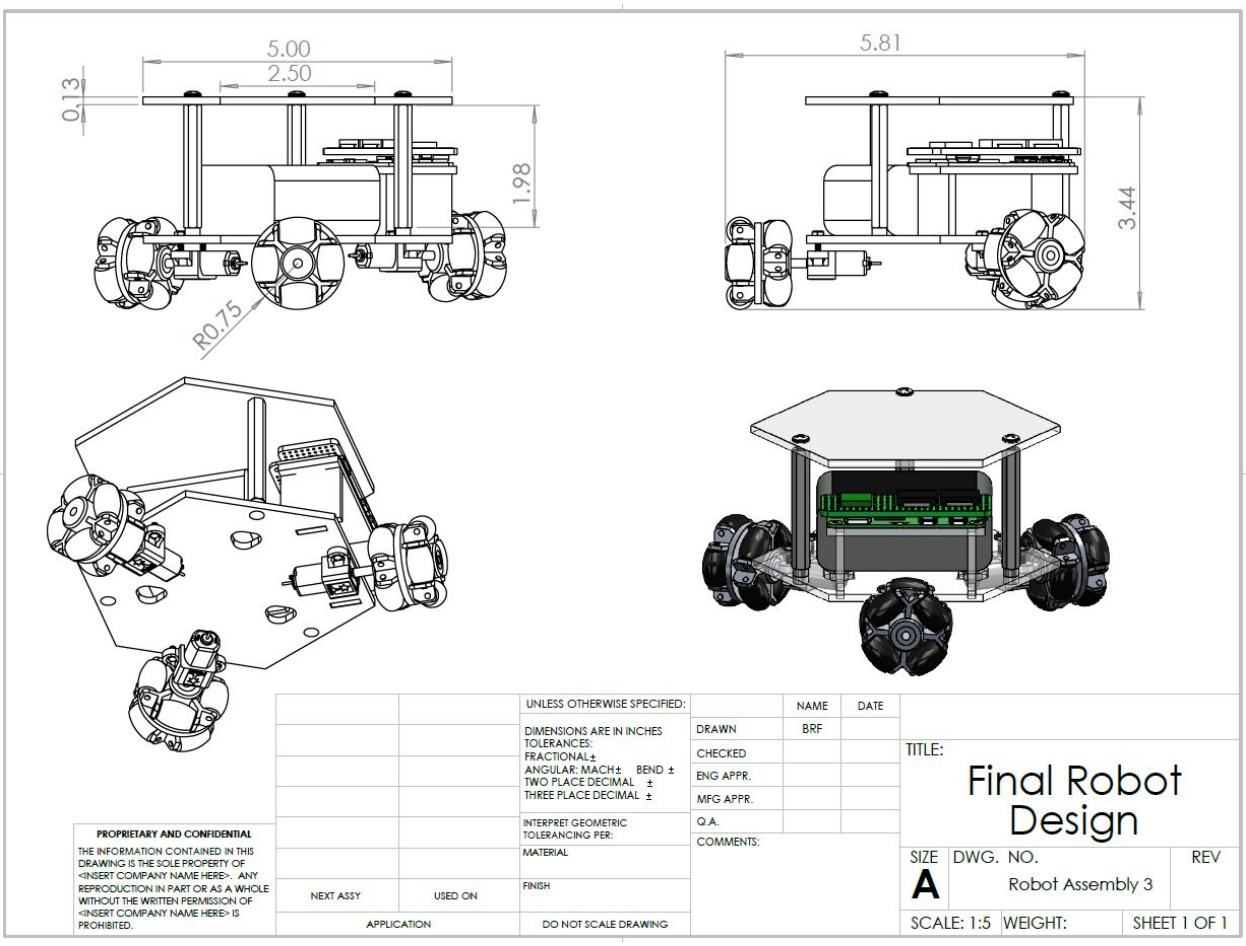
We would like to thank Professor Stolfi, Richa, and JB for all their help this semester. We would also like to thank Andrei Shylo and Bob Stark for their assistance during our countless hours in the MechE shop. Similarly, thank you to Professor Massimino for stepping in during Stolfi's medical leave. Finally, we would like to thank Amol Kapoor for letting us borrow his personal router and for advising us with the complex computer science portions of our project. We could not have finished Robo-Squad without the support of many.

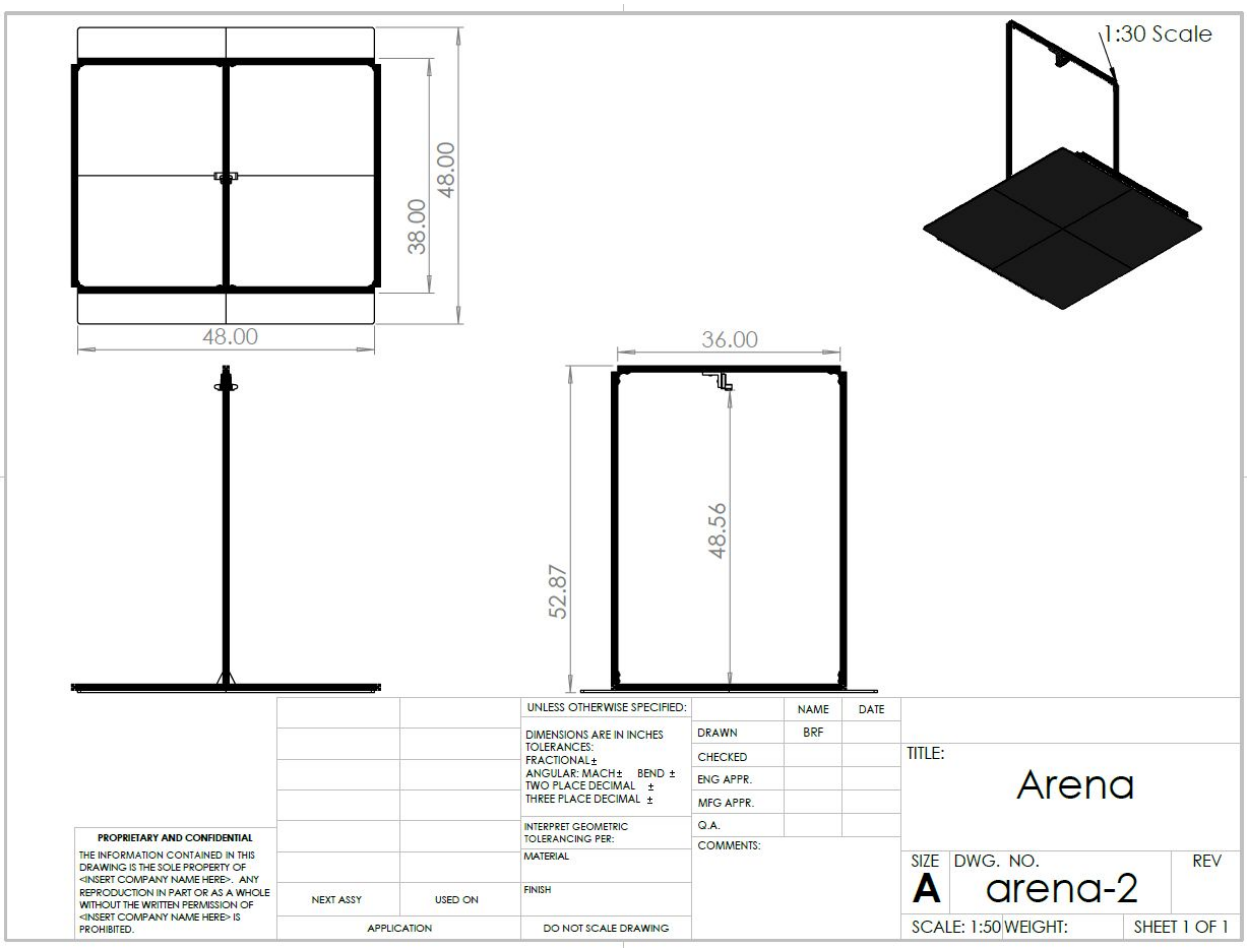
Reference List

[1] Bristlebot Swarm - https://learn.adafruit.com/simple-raspberry-pi-robot/overview
[2] Leader/Follower Arduino Robot - https://create.arduino.cc/projecthub/team-hermes/swarm-bots-assembly-and-co-operative-transport-fedbe0
[3] Multi-directional driving mechanism of a spheric wheel, US Patent 20130257138 A1 https://www.google.com/patents/US20130257138
[4] Magnetic spherical balancing robot drive, US Patent 20110231013 A1 https://www.google.com/patents/US20110231013
[5] Omni-directional, holonomic drive mechanism, US Patent 6810976 B2 https://www.google.com/patents/US6810976
[6] J.S. Bay, "Design of the "army-ant" cooperative lifting robot", IEEE Robotics & Automation Magazine pp. 36 - 43 vol. 2, 1995. http://ieeexplore.ieee.org/abstract/document/388293/

Appendix

CAD





		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	BRF	
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

PROPRIETARY AND CONFIDENTIAL

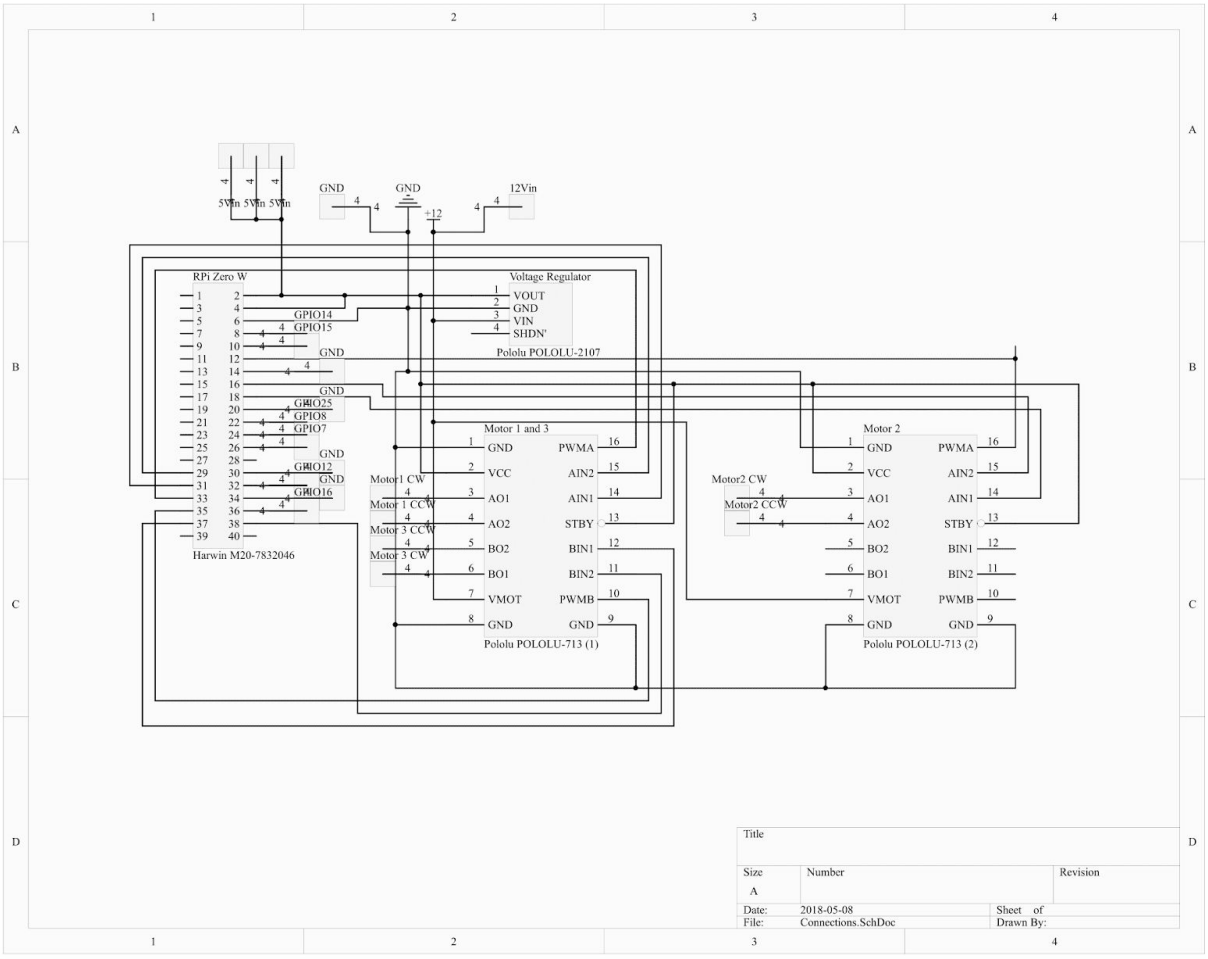
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

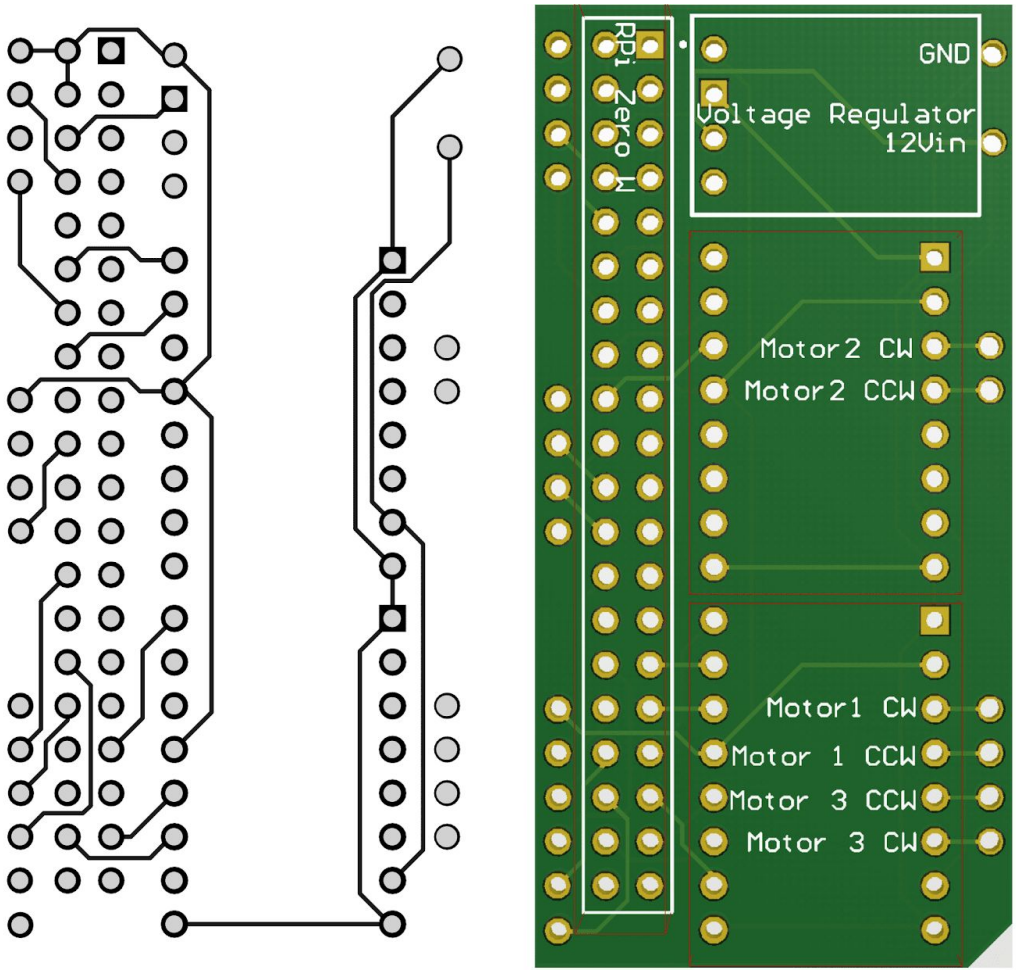
TITLE: Arena

SIZE	DWG. NO.	REV
A	arena-2	

SCALE: 1:50 WEIGHT: SHEET 1 OF 1

Electrical Schematics





Code

All our code is openly available on GitHub. We have separated the code onto two different GitHub repositories. Connie's GitHub repository (https://github.com/Catosis/RoboSquad) has our swarm and holonomic drive algorithms.

Meanwhile, Julia's GitHub repository (https://github.com/JuliaDi/Robot-Squad) has all our OpenCV (camera computer vision) code, as well as any communication (socket-client) code.

Notes (Budget)

We have prepared two budgets, the cost per robot and the cost for the arena.

Cost of One Robot (MECE + EE)						
Item #	**Vendor Name**	**Item Name**	**Unit Size**	**Unit Price**	**Quantity**	**Final Cost**
1	*Pololu*	Pololu Micro Metal Gearmotor Bracket Pair - Black	pack of 2	$4.99	1.5	$7.49
2	*Robot Shop*	38mm Aluminum Omni Wheel	1	$12.00	3	$36.00
3	*Amazon*	Tenergy 12V 2000mAh NiMH Battery Pack w/ Bare Leads	1	$23.92	1	$23.92
4	*Pololu*	Pololu 5V, 600mA Step-Down Voltage Regulator D24V6F5	1	$5.95	1	$5.95
5	*Pololu*	TB6612FNG Dual Motor Driver Carrier	1	$4.95	2	$9.90
6	*Advanced Circuits*	Custom PCB Fabrication	1	$33	1	$33
7	*McMaster*	High-Pull Rare Earth Magnetic Disc, Neodymium, 0.1" Thick, 1/4" Diameter	1	$2.10	3	$6.30
8	*McMaster*	Plastic-Head Thumb Screws Knurled, 1/4"-20 Thread Size, 3/8" Long	pack of 10	$8.00	1	$8.00
9	*McMaster*	Female Threaded Hex Standoff, 18-8 Stainless Steel, 1/4" Hex, 2" Long, 6-32 Thread	1	$2.73	3	$8.19
10	*Amazon*	Silicon Power 16GB x 2 MicroSDHC UHS-1 Memory Card Limited Edition-with Adapter (SP032GBSTHBU1V73TW)	pack of 2	$14.99	0.5	$7.50
11	*Polulu*	Machine Screw: #2-56, 7/16″ Length, Phillips (25-pack)	pack of 25	$0.99	1	$0.99
12	*Polulu*	Machine Hex Nut: #2-56 (25-pack)	pack of 25	$0.99	1	$0.99
13	*Polulu*	250:1 Micro Metal Gearmotor HPCB 12V	1	$15.25	3	$45.75
14	*Amazon*	15 Amp Unassembled Red/Black Anderson Powerpole Connectors Complete with Roll Pin (10 sets)	pack of 10 sets	$16.47	0.2	$3.29
15	*Adafruit*	Break-away 0.1" 2x20-pin Strip Dual Male Header	1	$0.95	`1	$0.95
					Amount Spent	$198.22

Cost of Arena Setup (incl Camera)						
Item #	**Vendor Name**	**Item Name**	**Unit Size**	**Unit Price**	**Quantity**	**Final Cost**
1	*Amazon*	Logitech HD Pro Webcam C920,	1	$54.86	1	$54.86
2	*McMaster*	T-Slotted Framing Single Rail, Silver, 1" High x 1" Wide, Solid, 10ft	10 ft	$31.59	1	$31.59
3	*Amazon*	Stalwart 6 Pack Interlocking EVA Foam Floor Mats Black 24x24x0.375	pack of 6	$14.70	1	$14.70
4	*McMaster*	T-Slotted Framing Single Rail, Silver, 1" High x 1" Wide, Solid	8ft	$23.57	1	$23.57
5	*McMaster*	T-Slotted Framing Single Rail, Silver, 1" High x 1" Wide, Solid	1	$6.07	4	$24.28
6	*McMaster*	T-Slotted Framing Diagonal Brace for 1" High Single Rail, 18" Long	1	$16.80	4	$67.20
7	*Amazon*	Rust-Oleum 249126 Painter's Touch Multi Purpose Spray Paint, 12-Ounce, Flat White	1	$3.48	1	$3.48
8	*McMaster*	T-Slotted Framing Single Rail, Silver, 1" High x 1" Wide, Solid, 8ft	8ft	$26.38	2	$52.76
9	*McMaster*	T-Slotted Framing Single Rail, Silver, 1" High x 1" Wide, Solid, 4ft	4ft	$14.20	1	$14.20
10	*McMaster*	T-Slotted Framing Corner Bracket for 1" High Single Rail, Black	1	$5.60	4	$22.40
11	*McMaster*	T-Slotted Framing Extended Corner Bracket for 1" High Rail, Silver	1	$5.85	2	$11.70
12	*McMaster*	T-Slotted Framing Tee Surface Bracket for 1" High Single Rail, Silver	1	$8.83	2	$17.66
13	*McMaster*	T-Slotted Framing Compact End-Feed Fastener, for 1" High Single Rail	4	$1.85	6	$11.10
14	*McMaster*	T-Slotted Framing Spring-Loaded Ball Fastener, for 1" High Single Rail	1	$1.12	1	$1.12
15	*McMaster*	T-Slotted Framing Compact End-Feed Fastener, for 1" High Single Rail	pack of 4	$1.85	6	$11.10
					Amount Spent	$361.72