Roborodentia Senior Project

Bryan Hendricks
bdhendri@calpoly.edu

Adviser: Dr. John Seng
jseng@calpoly.edu

June 2018

Table of Contents

Figures Index

Project Description

This project is an autonomous robot, designed to perform a series of basic tasks without any human input. It’s based on the 2018 Roborodentia competition, in which teams of students design and build a small (roughly 1 square foot) robot that collects small foam spheres from vertical tubes on the edges of a table-sized arena, and shoot them into goals across the field. The more foam spheres the robot makes into the goals after a 3 minute time period, the more points they get. The challenge is doing so autonomously, without any human input after the initial timer for the start of the match.

Several assistive elements are laid out on the field to allow a robot to detect its surroundings without human input. One such element, found on the floor of the field, is lines of black electrical tape against the white background that lead to all four vertical loading stations, as well as a separate horizontal loading station. The robot I built detects these lines using a series of infrared sensors, and navigates from loading station to loading station using these lines as guides, shooting the foam spheres into goals along the way.

Design

Hardware Design

The design for the robot was done primarily in SolidWorks, a 3D engineering design program. SolidWorks was chosen based on its ability to accurately lay out 3D components in order to test fits, as well as its ability to export cross-sections of 3D components as DXF files, which can be used to laser-cut components.

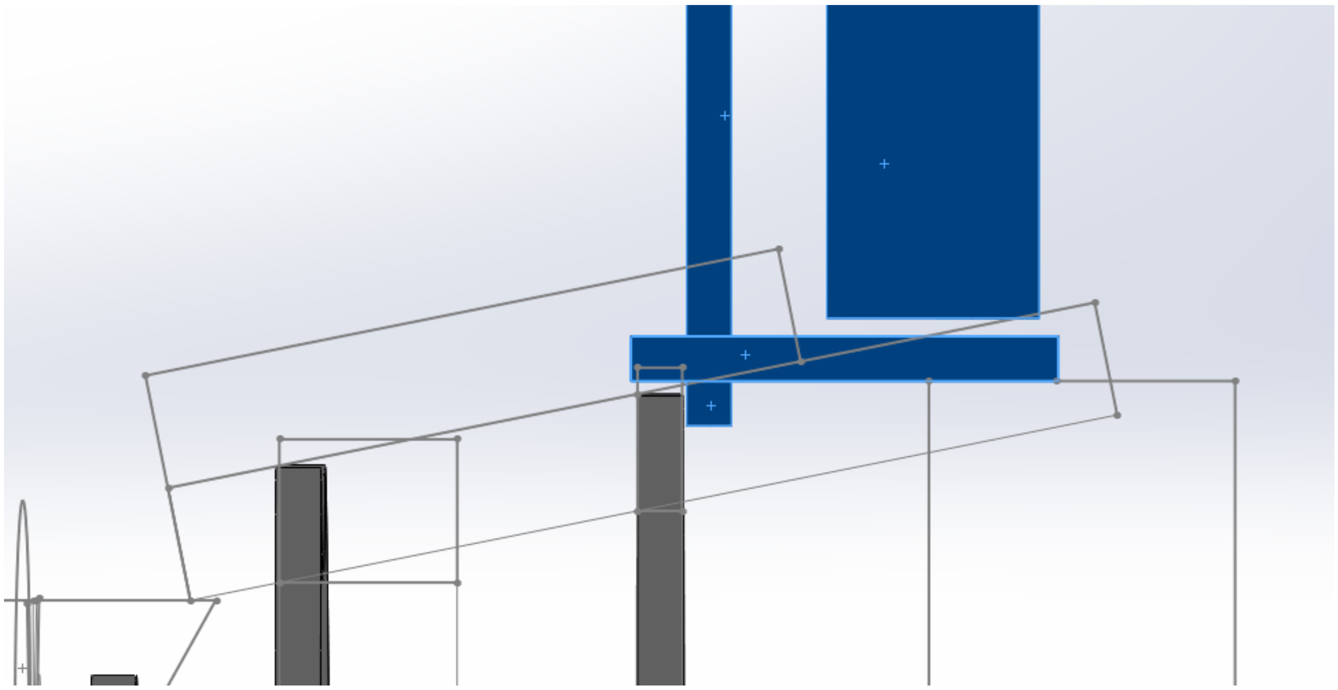
Figure 1, SolidWorks Design of Loading Mechanism

The first step of the design was to set up the contact points between the robot and elements of the field, namely the loading mechanism and the drive train (wheels and base). In *Figure 1*, the blue components are the loading mechanisms on the edge of the field, the gray lines are the outline of a PVC pipe contacting the loading mechanism, and the gray components are fins holding up the PVC pipe.

In *Figure 2*, the blue component is the base of the robot, and the gray components are the robot's wheels and the floor of the arena.

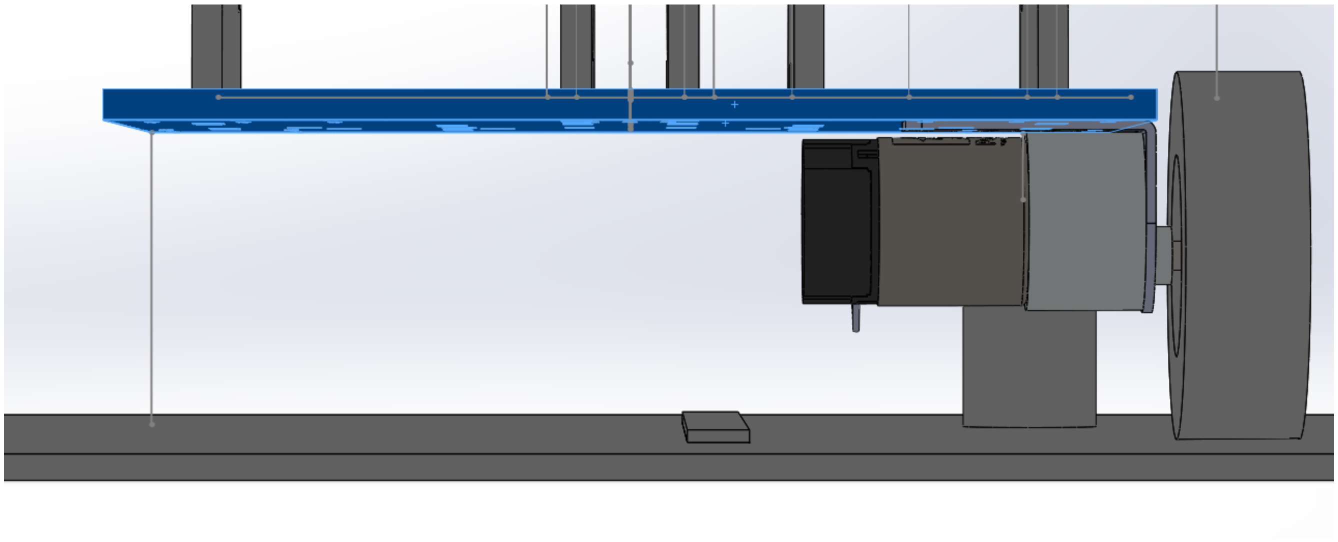

Figure 2, SolidWorks Design of Robot Base

After these contact points were positioned, the attachments between them were added. In *Figure 3*, the blue components are the fins that hold up the PVC pipes and the ramp that guides the foam spheres to the goal.

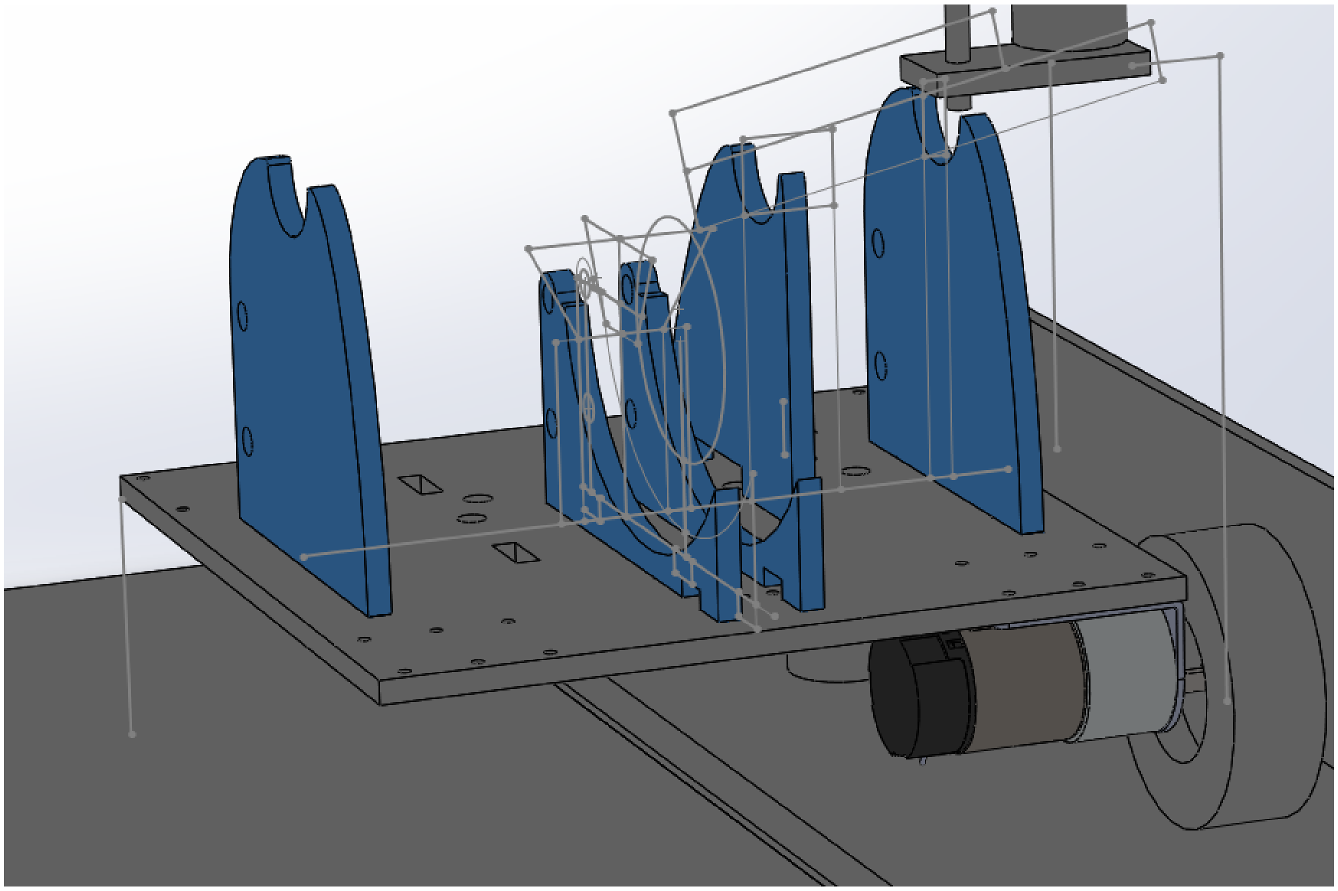

Figure 3, SolidWorks Design of Supporting Structures

The final component, the firing mechanism, wasn’t designed to be made out of laser-cut material, and thus wasn’t modeled. However, the mechanism still had to be fitted, and a sketch was used to align it with the rest of the 3D components. The sketch is highlighted in orange in *Figure 4*.

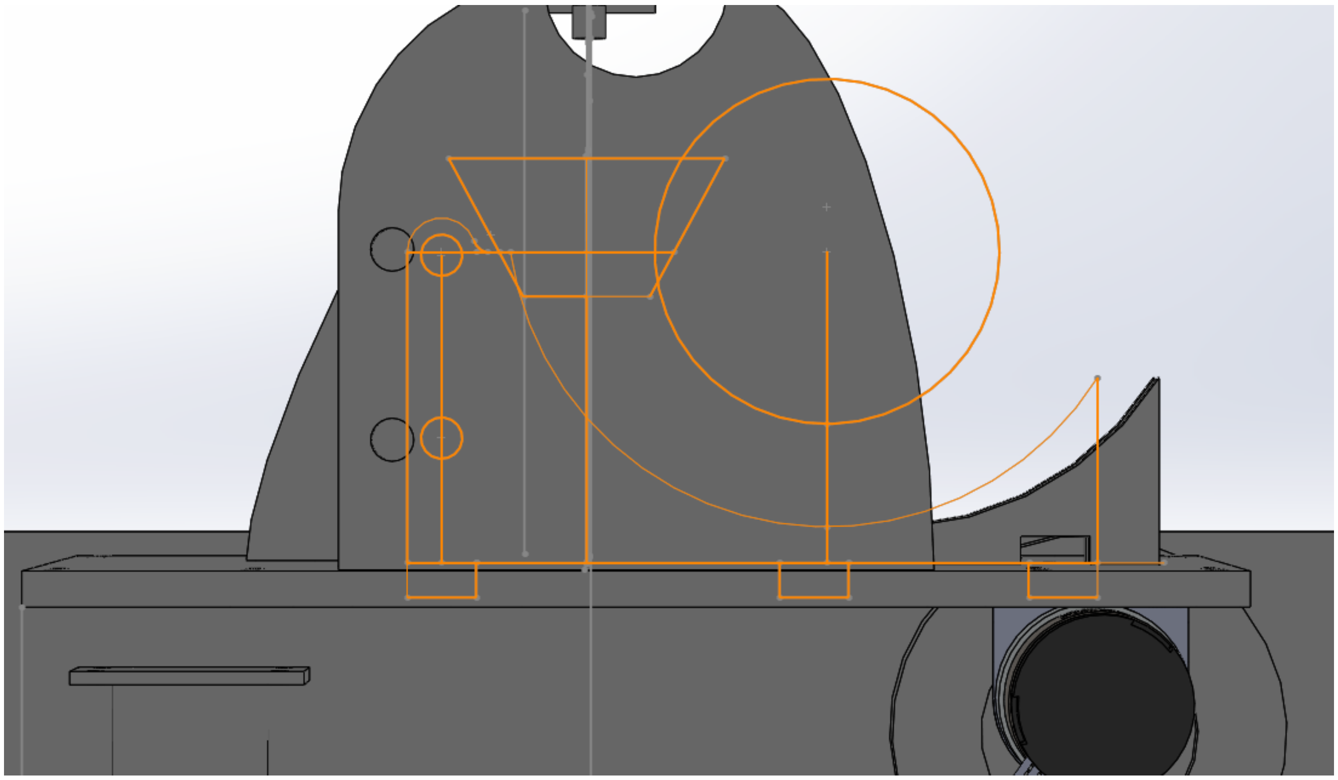

Figure 4, SolidWorks Design of Firing Mechanism

Also located in *Figure 4* is the outline of the container that holds the foam spheres. His is a 2D cross-section of what would become a conical shape, allowing the foam spheres to funnel down into the firing mechanism based on the servo motor’s movements.

Software Design

The software design began with the robot’s navigation system. I decided to use a line-following system because it had a balance of being simple to implement and tune, while still being more accurate than something like static timing. After some research and testing, I found that I had 4 control systems available to choose from for the line following system:

- Finite State Machine/P Controller: Manually create logical transitions from state to state of the set of 3 infrared sensors, power the motors based on how far away from the center of the line the robot is.
- PID Controller: Turn the positional value from the infrared sensors into a linear scalar, and run a PID algorithm on it, which would take into account previous positions and predicted future positions.
- Cascade PID Control: Split up the PID controller into 3 components – PID controllers for the speed of each wheel, and one PID controller to control the desired speeds of each wheel.
- Feedforward Neural Network: A multi-layer perceptron neural network that reads in the state of the robot’s current and past positions relative to the line, and outputs the desired motor speeds based on training data.

After more research and testing of these 4 systems, I decided to use the simplest one, the Finite State Machine/P Controller, as that would allow me the greatest control over the power of the motors at each stage. Given there are very few of these stages obtainable from 3 infrared sensors, it was feasible to manually tune the power at each stage.

The basics of this state machine/P controller method are fairly simple. Given 3 infrared sensors, which can differentiate between black electrical tape and the rest of the white floor, that have been spaced apart at specific intervals (roughly 0.5”, based on 0.75” wide electrical tape), by examining the color of the field underneath then, a rough position of the robot relative to the lines of tape can be achieved. If the robot is centered on the tape, only the middle sensor detects the tape. If it begins to veer left or right, both the middle sensor and one of the outer sensors will detect the black tape. In this case, the robot lightly steers back towards the middle of the tape. If the robot is unable to compensate for the curve of the tape at this rate, the middle sensor continues off of the tape, leaving only an outer sensor detecting the tape. In this case, the robot steers more forcefully back towards the middle of the tape. If the robot is again unable to compensate, the last remaining sensor moves off the tape, leaving the robot effectively blind. However, because of the use of a state machine, the software can identify which side of the line it was on most recently, and can move back towards where the tape is expected to be much more forcefully. Here is an example of one such state used in the Arduino code:

```
case POS_LEFT_1:
  // No sensors can see the line
  if (!left_sensor_line && !middle_sensor_line && !right_sensor_line) {
    pos_state = POS_LEFT_3;
  }
  // Only the right sensor can see the line
  else if (!left_sensor_line && !middle_sensor_line && right_sensor_line) {
    pos_state = POS_LEFT_2;
  }
  // Only the middle sensor can see the line
  else if (!left_sensor_line && middle_sensor_line && !right_sensor_line) {
    pos_state = POS_MID;
  }
  // The left and middle sensors can see the line
  else if (left_sensor_line && middle_sensor_line && !right_sensor_line) {
    pos_state = POS_RIGHT_1;
  }
  // All sensors can see the line
  else if (left_sensor_line && middle_sensor_line && right_sensor_line) {
    pos_state = POS_T;
  }
  break;
```

Figure 5, State Machine Example

In this example, you'll see an extra case built in for when all 3 sensors can see the line. This occurs when the robot encounters a T-intersection of tape from the bottom, and is utilized to signal the robot to turn left or right to follow one line direction or another.

While developing for the Raspberry Pi 3, I had to re-work the navigation system due to the fact that the Raspberry Pi has no analog input pins. The infrared sensors output an analog voltage, ranging from 0V to 5V, and without an Analog to Digital Converter, the Raspberry Pi wouldn't be able to read from them. As a result, the Raspberry Pi version of the software utilizes raw timing to run the course. It follows the same path as the original line follower in order for the user to closely monitor the robot's path, and pick up and adjust the robot (incurring a small penalty) if it gets too far off course.

Because this project was developed for both an Arduino Uno and a Raspberry Pi 3, there are 2 versions of the code – one in C++, and one in Python. They can both be found at https://github.com/bryanhendricks/senior_project_2018.

Build

After the 3D modeled components were laser-cut out of wood and assembled, the rest of the components were added. The drive motors were mounted to the front of the base using some included motor mounts and some hardware, and free-spinning casters were mounted to the back of the base using other hardware. A pair of cut PVC pipes were attached to the top of the robot, in order to intersect with the loading mechanisms and acquire the foam spheres.

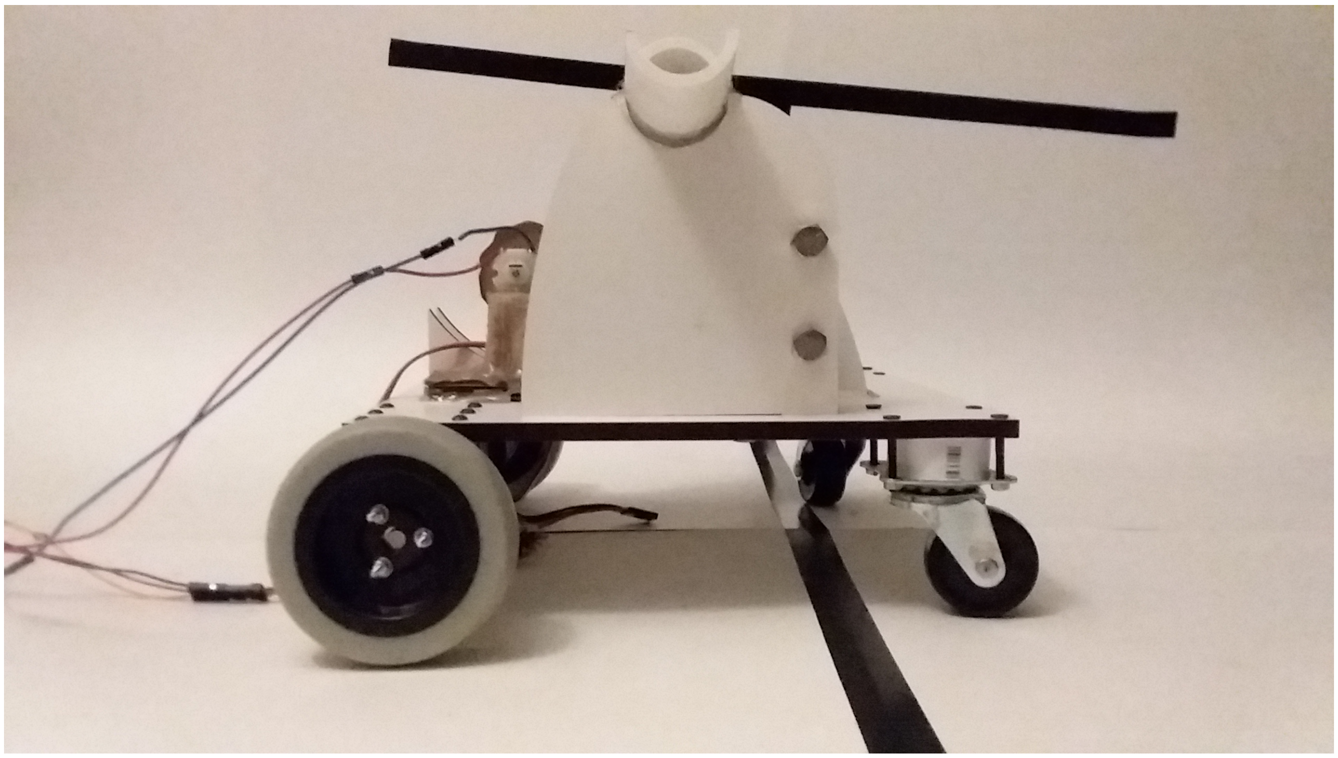

Figure 6, Completed Robot

A small fly wheel was made using a DC motor, cardboard, tape, and glue. This was the main component that wasn't designed in SolidWorks, as it had no laser-cut components, and was built by hand. By stacking circles of cardboard together with glue, and attaching it to the motor shaft, a wheel with enough speed and mass to propel the foam spheres was made. Duct tape was wrapped around the outside of the wheel, as it was discovered that duct tape gripped the foam spheres much better than cardboard, and allowed for easier propulsion.

The firing mechanism was made out of a small servo motor with a series of attachments on the end. The servo motor's arm was programmed to stay in the way of the foam spheres dropping into the curved firing chute, and move out of the way on command. Because the holding tray for the foam spheres was just the right shape, on occasion two would get stuck against each other near the top, and no spheres would be able to fall into the fly wheel. Attaching a cardboard wing to the firing trigger servo motor allowed it to jostle the holding tray, and guaranteed the release of all stored foam spheres after a handful of open-close cycles.

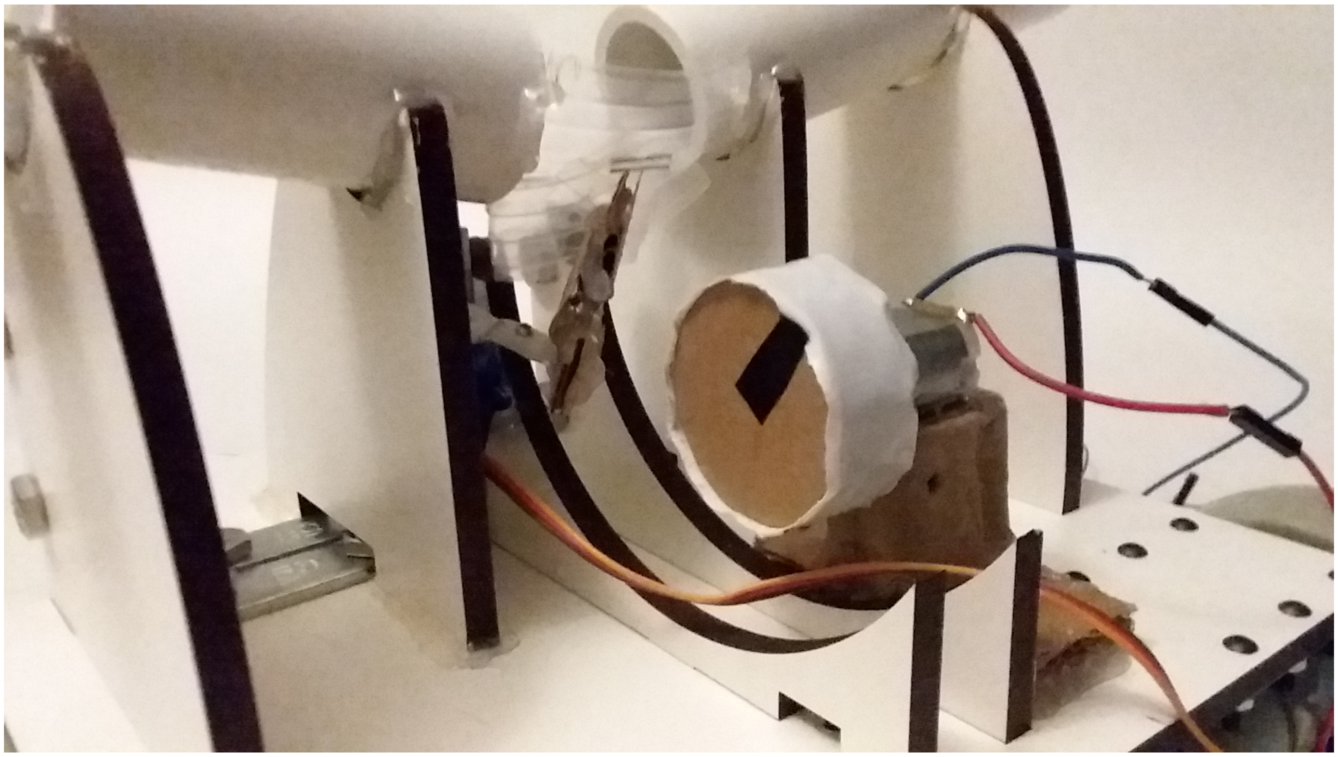
Figure 7, Completed Fly Wheel and Firing Mechanism

Results

During the testing phase, while calibrating the movement timings on the Arduino control system, the Arduino I was using stopped working. I eventually diagnosed it to down to bad firmware, but without another specific external device to flash new firmware onto the Arduino, the device was effectively bricked. This led to the utilization of another device, a Raspberry Pi 3 – however, much of the software had to be adjusted to match this, as it had to be translated from C++ to Python. The Raspberry Pi 3 was able to effectively run the firing mechanism, including the servo motor and the small fly wheel motor, but was unable to power the larger drive motors. The digital GPIO pins on the Raspberry Pi 3 ran at 3.3V, which is enough to power the drive motors effectively from an external power supply, but the GPIO pins were unable to provide the required amperage to do so.

I researched other alternatives, including using the Raspberry Pi 3 as an external device to flash the Arduino firmware or to manually control the Arduino motor shield, but the lack of any online resources for these two processes meant that doing either wasn't feasible given the short time I had left. Overall, each individual component of the robot has been proven to work, but without another Arduino or another motor controller, they aren't able to be controlled in tandem.

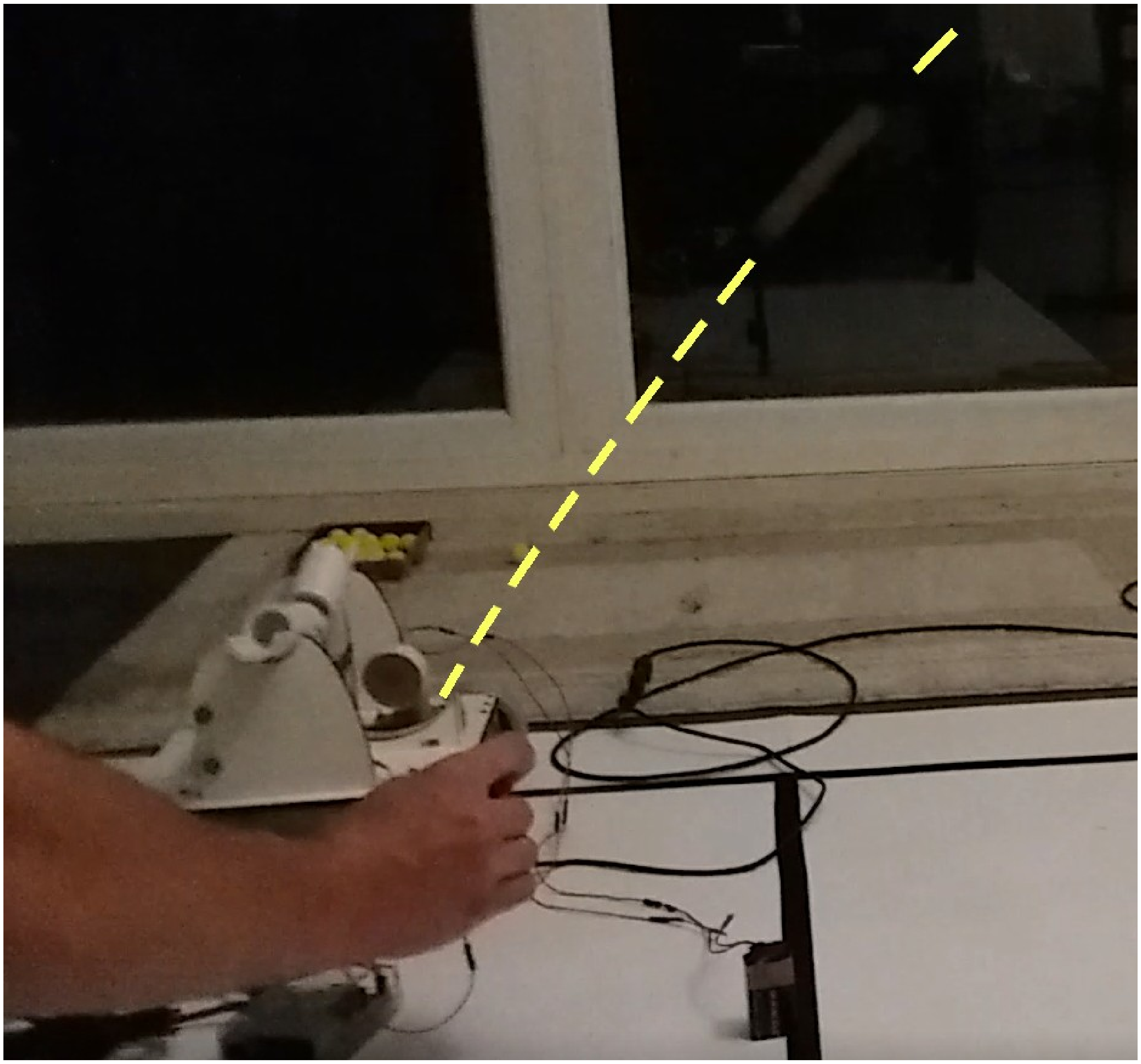

Figure 8, Trajectory of Foam Sphere Fired

In spite of this, the robot was proven to function enough to score points in a Roborodentia match. As shown in *Figure 8,* the combination of the servo motor firing mechanism and the DC motor fly wheel were able to propel the foam sphere with enough velocity to easily make it into a goal. This demonstrates the ability to successfully score points, and with a replacement Arduino, the robot is ready to effectively compete.

Future Improvements

During the testing phase, I discovered that the loading mechanism had a flaw that would occasionally cause two foam spheres to become stuck on each other, preventing them from falling down into the loading mechanism. I was able to fix this by adding the wing to the firing trigger servo motor, allowing it to jostle the holding tray enough for the foam spheres to come loose, but this system would be a good place for future improvements. These could be in the form of a more wide-open holding tray, or a dedicated actuator to jostle the foam spheres to ensure they never got stuck. The wider holding tray method could work well if designed correctly, but doesn't guarantee that foam spheres won't get stuck. A dedicated actuator to move the foam spheres would guarantee they wouldn't get stuck, but would raise the price and complexity of the robot as a whole.

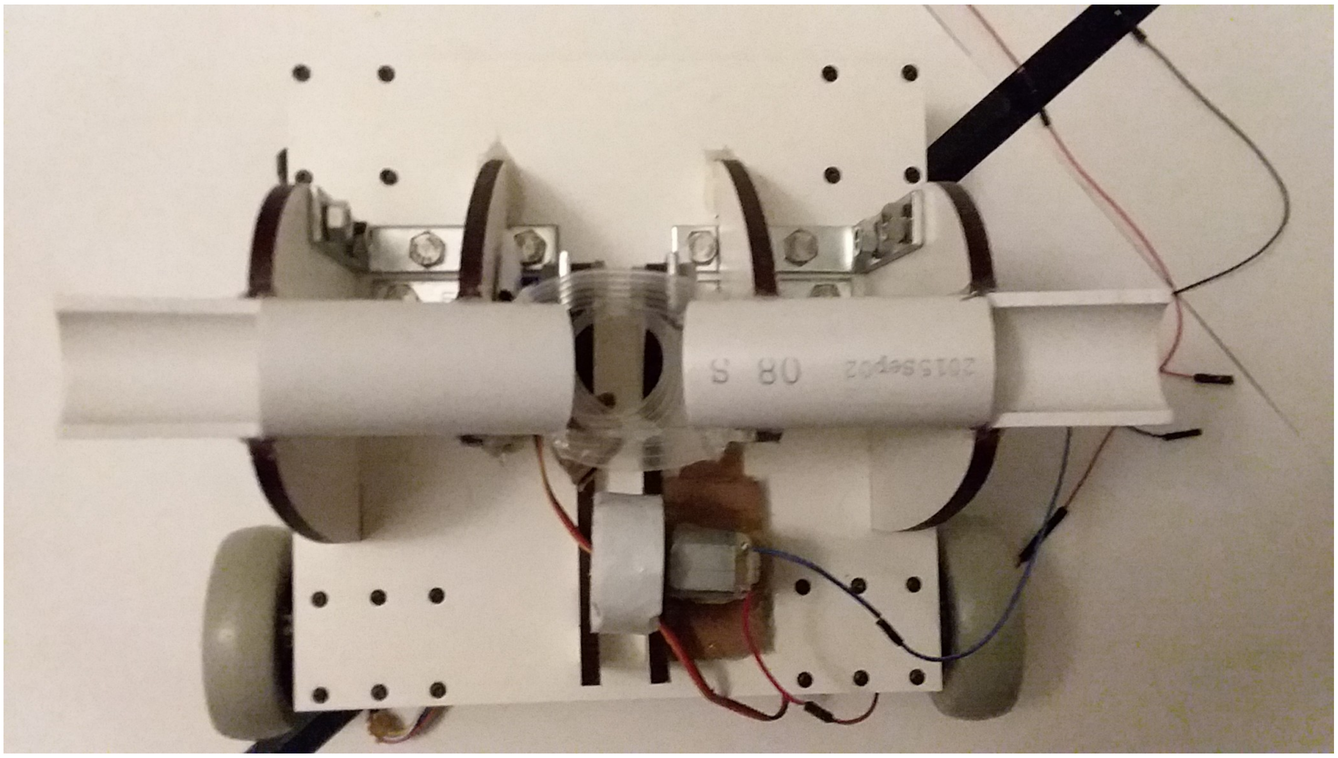
Figure 9, Robot Loading Mechanism

Another possible improvement could be, again, to the holding mechanism. As of right now, the robot has a capacity to hold up to 10 foam spheres. With a larger holding tray, more foam spheres could be held, which would decrease the time spent stopping to fire, which would increase the number of points that could be earned. In addition, the acquisition components could be expanded. The PVC pipe utilized here works well due to its ability to guide the foam spheres to a central location, but has a fairly narrow range of acquisition when interacting with the field loading mechanisms. Using a wider material would allow for a larger margin for error, as well as increase the maximum number of foam spheres able to be held.

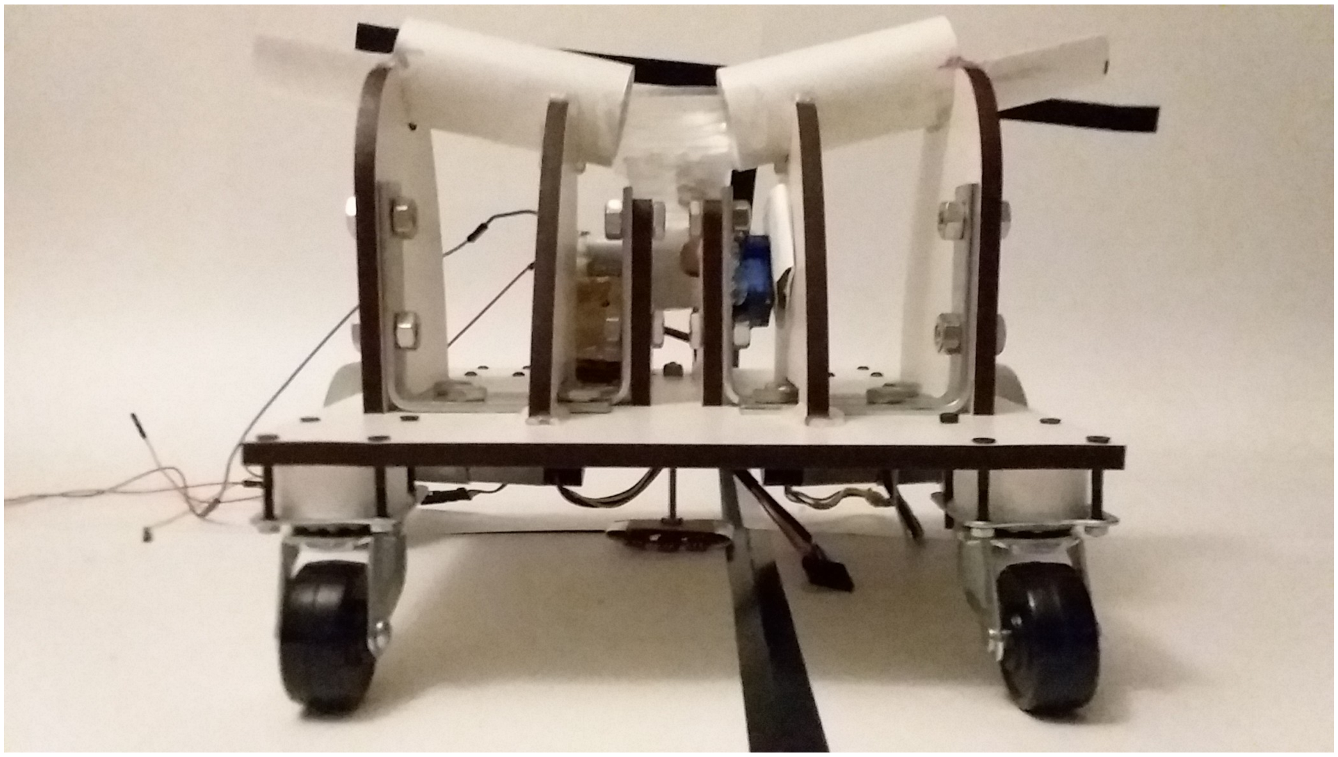
Figure 10, Completed Robot Casters

More improvements could be made to the hardware requires for majority of the robot. Most of the parts were custom, or assembled out of smaller custom parts. As shown in *Figure 9,* the mounting system for the rear casters requires custom PVC length to be cut, as well as specialized casters and bolts to attach the system to the base. This could be improved by adding more mounting points to account for multiple caster wheel systems, or by standardizing the spacing component as more laser-cut wood rather than PVC pipe.

Project Analysis

Summary of Functional Requirements

This robot is designed to complete a series of basic tasks autonomously. It was built based on the 2018 Roborodentia competition, in which teams of students design and build a small (roughly 1 square foot) robot that collects small foam spheres from vertical tubes on the edges of a table-sized arena, and shoot them into goals across the field. The more foam spheres the robot makes into the goals after a 3 minute time period, the more points they get. The challenge is doing so autonomously, without any human input after the initial timer for the start of the match.

Several assistive elements are laid out on the field to allow a robot to detect its surroundings without human input. One such element, found on the floor of the field, is lines of black electrical tape against the white background that lead to all four vertical loading stations, as well as a separate horizontal loading station. The robot I built detects these lines using a series of infrared sensors, and navigates from loading station to loading station using these lines as guides, shooting the foam spheres into goals along the way.

Primary Constraints

One of the main limiting factors in this project was monetary cost. There are several robot component kits available for purchase on the internet that accomplish the majority of the tasks that this Roborodentia competition required, but I wanted to accomplish the task with the minimum amount of purchased resources as possible. As such, I used an Arduino and an Arduino motor shield I had previously purchased for an old project, a small assortment of motors from a different previous project, and several pieces of scrap wood and PVC pipe. While the overall monetary cost was still very low, the short-term monetary cost of the project was almost zero, which achieved the goal I had set.

Another limiting factor, closely related to the first, was the lack of available electronics. While the Arduino and the associated motor shield worked well, the Arduino stopped working during the final tuning phase of the project. Another Arduino would have fixed the problem easily and quickly, but with the lack of physical resources available, several other alternatives had to be attempted. Fixing the Arduino was possible, but required another Arduino board to flash firmware to it, which looped back to the lack of resources. I did have an alternative of a Raspberry Pi 3 from a personal project, and while this did allow for some progress after re-writing the code for it, it ultimately proved unusable as the digital output pins didn’t have enough amperage to move the larger drive motors. Another alternative was attempted for using the Raspberry Pi 3 as an external device to fix the Arduino or to manually control the Arduino motor shield, but the lack of any online resources for these two processes meant that doing either wasn’t feasible given the short time I had left.

Economic Impact

At the start of the project, the estimated price came to $114, but with previously purchased items available to me, the short-term cost was roughly $30 for the hardware and the wood/PVC.

Component	*Price*
Arduino Uno	$22
Arduino motor shield	$22
Motors/Servos	$40
Wood/PVC	$10
Hardware	$20
Total Estimated Price	$114

Figure 11, Estimated Cost of Components

After the project was completed, the used materials and components had risen in price. While still relatively low, the broken Arduino has caused more components to need to be utilized.

Component	*Price*
Arduino Uno	$22
Arduino motor shield	$22
Raspberry Pi 3	$39
Motors/Servos	$40
Wood/PVC	$10
Hardware	$30
Total Price	$163

Figure 12, Final Cost of Components

At the beginning of the project, the estimated development time was roughly 2 weeks of active work. Time had to be split with other courses, so this 2 weeks of active work was scheduled to take place over the course of 4 to 5 weeks. However, other courses took up significantly more time than expected, and this 4 to 5 weeks extended to 8 to 9 weeks. In addition, the original 2 weeks of active work time was extended due to the issues with the Arduino. The final development time took place over the course of roughly 10 weeks.

Commercial Manufacturing

While this device wouldn’t be manufactured on a commercial basis for this purpose, many devices much like it are sold as un-assembled kits for people to built and learn about electronics and programming. If this project were to be re-packaged as such a kit, depending on the seller and advertising, it could sell several hundred per year. The manufacturing cost of the kit would be closer to the total price found in Figure 1, as the total price in Figure 2 is inflated due to development costs, and the final kit could be sold for roughly around $150, yielding a 31.5% profit over raw materials costs. Some percentage of that would be put towards costs other than materials, such as labor, and would bring the cost from $114 to roughly $130. This still yields a 15.4% profit, and if 500 units are sold per year, it can result in a net profit of $10,000. Not enough to base an entire business off of, but a significant amount when added to an existing online electronics store.

Environmental Impact

This robot doesn’t have any environmentally hazardous components to it, the electronic components can be sent to an electronic waste recycler rather than be thrown away, and the majority of the rest of it is made of wood. Most scrap materials from manufacturing can be recycled or is more environmentally friendly, but the addition of PVC pipes to the design does cause an impact on the environment. Parts of the pipes must be cut off in order for the rest to be used, and the buildup of extra PVC pipes as waste could prove detrimental to the environment. However, due to the very low number of manufactured items, it is unlikely that this would have any measurable effect on the environment as a whole.

Manufacturability

Due to the majority of the robot being made of laser cut wood, manufacturing a kit for this system would be far easier than cutting out pieces by hand. The largest setback when manufacturing would be cutting the PVC pipes, as using abrasive tools to saw through parts of it release PVC dust, requiring extra safety equipment to prevent machinists from breathing them in. Because it’s a kit, though, the rest of the components are already manufactured by other companies, and can be packaged in without any extra processing.

Sustainability

Because the robot uses a single 9V battery as a power supply, the battery must be replaced periodically. Beyond this, very little maintenance is required by the user; however, significant maintenance will likely be performed by the user due to the nature of the product being a learning kit. It’s designed to be experimented with, and that will result in more time from users being put into it after receiving it.

Possible upgrades to this project are extensive, and are the goal of a kit as a product. It could be expanded by adding more infrared sensors to improve line-following accuracy and speed, or by changing the software to utilize the encoders on the motors rather than the line-following sensors, or by adjusting the physical layout to improve the lading and firing. The list of possible upgrades is nearly endless.

Ethical Considerations

Not very many ethical issues can be brought up with a small Roborodentia learning kit. It could be argued that the simple act of selling a kit is unethical, as it is essentially re-packaging and re-selling the products of other companies, but this robot was designed around the existing components and adds on to it with the laser-cut parts and the provided software. The value from the design of the laser-cut parts, the software, and the work that went into figuring out which components work together should be enough to merit selling the kit as its own entity.

Health and Safety

While this robot is fairly safe, as it doesn't move very fast and utilizes relatively low electrical voltages/amperages, it does have a few sharp corners, and does shoot projectiles. The sharp corners can be rounded out in the design easily, but the projectiles can cause injury. They are just small foam spheres, but they move quite quickly, and cause damage to the eyes if hit directly. During the Roborodentia competition, participants and assistants were required to wear eye protection to mitigate this risk, but such a rule can't be reasonably enforced if this project were turned into a kit for sale.

Social and Political

If this Roborodentia robot was turned into a kit and was spread, it would help promote the learning of electronics and programming. The main goal of many similar robot kits is the same, and it proves to be very effective, particularly with younger children.

Development

During the development of this robot, I learned a significant amount about physical design and control systems, as well as alternatives to the electronic systems I had available. While I was designing the robot's physical layout on paper, I believed I had all the kinks worked out and that it would be able to function properly. As I moved this design from paper into SolidWorks, I discovered several small issues that would prove to be problematic unless I had caught them before I began making the physical components. This transition from 2-dimensional design to 2-dimensional design taught me a significant amount about how to design physical systems and components.

Another topic that I learned more about was control systems. As I was considering a control method for the driving portion of the robot, I considered reusing a simple line-following system I had implemented in a robot in the first CPE course I had at Cal Poly. I did some research first, and found that control systems for such a task ranged from the state machine I had already utilized, to PID controllers, all the way to neural networks. I did some experiments and implemented each of the control systems, but eventually settled on the simpler P controller due to the ease of calibration.

The largest topic I learned about was different electronic control systems. I knew enough about Arduinos to be comfortable using them, but when the only one I had broke, I had to start looking for alternatives. As I mentioned earlier, I researched and experimented with several alternatives. I found a way to fix the broken Arduino that I had, but it required another Arduino to flash new firmware onto it, and having a second Arduino would fix my problem regardless. After several failed attempts to revive the Arduino, I began experimenting with a Raspberry Pi 3. I researched options to flash the firmware to the Arduino from the Raspberry Pi, or to manually control the Arduino motor shield from the Raspberry Pi, but I wasn't able to find any online resources to assist in this, and implementing either method from scratch wasn't feasible given my time frame. I attempted to control the motors on the robot directly from the Raspberry Pi, but while the voltage was just enough to get them moving, the available amperage from the GPIO pins wasn't enough. Without a motor controller, or even a handful of transistors, the Raspberry Pi wasn't enough to power the drive motors. It could, however, drive the servo and small DC motor I used to fire the foam spheres, and this allowed me to test and verify the functionality of the vast majority of the robot.