Hippautonomous Team

Advance Mechatronics Final Project Report

By: Laura Jones, James Hand, Mohammed Alajaji

May 2nd, 2021

Table of Contents

A. Introduction
B. Division of responsibilities
C. Member’s contributions
- James Hand
- Mohammed Alajaji
- Laura Jones

D. Analysis of the project
E. Guidance
F. Bill of materials (BOM)
G. Appendix
- Link to Code
- Requirements Document link
- Budget
- Schedule

H. References

A. Introduction

The aim of the project was to design and create a mechatronic system that coverts the Hungry Hungry Hippos game to a robotic game. The project involved creating an autonomous car that carries a hippo facsimile, a vision system, and a collection system. The system was designed in such a way that it is best operated by people with an interest in STEM, computer vision, or simply like autonomous systems. We suggest that operation is done by anyone twelve years or older with access to a computer due to the occasional setup requirements. This product would also be a great teaching tool for teachers trying to introduce students into vision systems or programing.

B. Division of responsibilities

Table 1: Division of Responsibility.

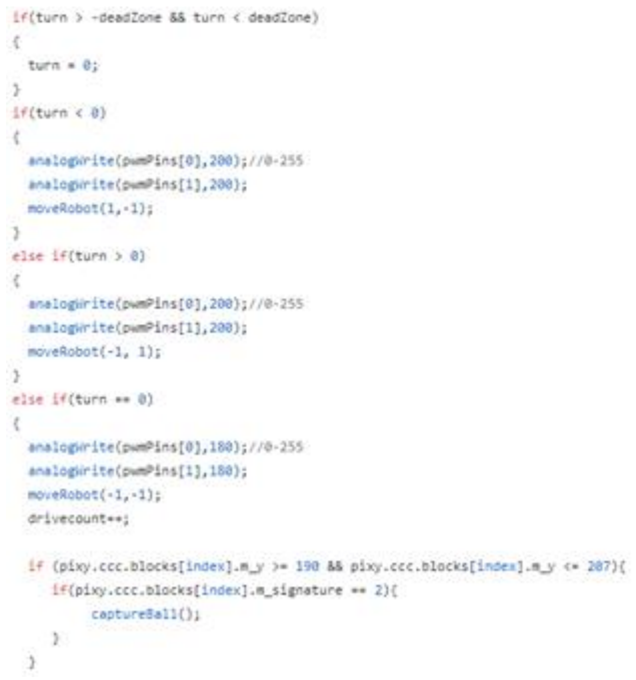
Name	Main Responsibilities	Tasks	Comments
Laura	Electrical System	Wiring	
		Servos	
		Powering	
	Collection Mechanism	3D Printing	
James	Capture/Motion Algorithm	Motor Controls	
		Image Processing	
		LED Movement	
	Play Area	Materials	
		Design	
	Original Directional Code and Distance Algorithm		This code does not have a bearing on the final product.
Mohammed	Vision System	Train the PIXY2 Camera.	
		Fine tune the PIXY2 camera	
		Lighting	
		Ball's size and color	
		Noise reduction	
	PIXY2 and Microcontroller communications	Connection	

C. Member's contributions

- James Hand

<u>Capture/Motion Algorithm:</u>

This program is what runs on the Elegoo UNO microcontroller. It handles incoming data from the Pixy2 camera, separates it into usable bounding boxes, and derives motion controls based on that information. To do this I created the program so that it creates a "pixy" object from information sent over a serial connection from the Pixy2 camera using an official API. This object contains an array of bounding boxes and their relevant information, such as height, width, xy coordinates, etc. From here I simply found the closest, aka the largest, bounding box and created a system that determines where in the frame that point is. Given the xy coordinates of the bounding box are for the center of the bounding box I used this to find whether it was in the left, right, or center of the frame based on its X coordinate value. Given how precise this system is in determining direction I added a small dead zone area that is counted as center. Once the direction is decided using this logic, I give the car motors a command by changing left and right-side motor polarities to turn the car the desired direction. This is highlighted through the example of Figure 1.



```
if(turn > -deadZone && turn < deadZone)
{
  turn = 0;
}
if(turn < 0)
{
  analogWrite(pwmPins[0],200);//0-255
  analogWrite(pwmPins[1],200);
  moveRobot(1,-1);
}
else if(turn > 0)
{
  analogWrite(pwmPins[0],200);//0-255
  analogWrite(pwmPins[1],200);
  moveRobot(-1, 1);
}
else if(turn == 0)
{
  analogWrite(pwmPins[0],180);//0-255
  analogWrite(pwmPins[1],180);
  moveRobot(-1,-1);
  drivecount++;

  if (pixy.ccc.blocks[index].m_y >= 190 && pixy.ccc.blocks[index].m_y <= 207){
     if(pixy.ccc.blocks[index].m_signature == 2){
          captureBall();
     }
  }
```

Figure 1: Turning Logic

The capturing code, where the system realizes it is close enough and begins a capture sequence, was created by Mohammed. This happened while I was away for a week so I merely provided guidance during this period on how the system might work, as well as pseudo code for the basics of this program. This also includes the use of PWM signals to change the speed of the car as it approaches the ball, a collaborative effort between Laura and Mohammed. However, I did design and write the code following ball capture that allows the car to drive back to its estimated starting position. This involves keeping track of how many times the system has "driven forward" and using that number in a reverse manner to back the car up to the starting position.

In addition, we began to implement a program in the last week of testing to allow for control over the onboard LEDs based on car movement. This code was mostly written by me, but Laura contributed a lot to the planning, and testing phases of this code. We spent a lot of time at Mohammed's place trying to troubleshoot why the program was not changing the LEDs at the right time. We came to the conclusion that there was some sort of fault in how the Arduino Nano interpreted digital signals from the Elegoo Uno for timed events. A practice program, created by Laura, worked just fine, but in the high speed and ever changing events that our driving code created all signals would almost always default to a single LOW state value, creating only one type of LED response. I created a system that stored received LED signals to get around this hurtle, but its results were spotty at best for some undetermined reason. Because of this we defaulted the LEDs to a single type of response due to time constraints and lack of impact on the overall performance of the car in its required tasks.

Play Area:

The play area discussion was one we had for a sizeable part of our early planning stages. Our original ideas included things like a walled in arena, an open space on carpet with tape boundaries, and many others. As part of this planning I went to various hardware stores and located suitable materials for constructing a play area. However, as our car design changed so did our play area requirements. After deliberation we decided, at first, that it would be more cost effective and reliable to continue without a constructed play area. So, the research I did was put on the back burner and stored away. However, while I was gone Laura and Mohammed determined that given the car's weight, its construction, and stability it would not be able to properly function on the classroom's carpet. This brought back the research that I did into a play area and while I was in Arkansas, I looked into materials there.

Upon returning to Florida, I met with Mohammed and Laura to discuss the issues they ran into. Once I fully understood the issue I looked back at my research and found a suitable material for our purposes that was light weight, didn't create Pixy noise, relatively cheap, and provided very little friction which helped with the car issues. We bought the tempered wood in two sheets of 4'x8' and cut them in two for ease of transport, creating four 4'x4' slabs. With this material I labeled them with identifying numbers, arrows corresponding to their neighbor boards and orientation, and I traced the ball play area markings that Laura and Mohammed measured. I was also in charge of all transport for the boards given that they would not fit in Mohammed or Laura's cars, and it barely fit in mine. Below is an image of one of the four boards.

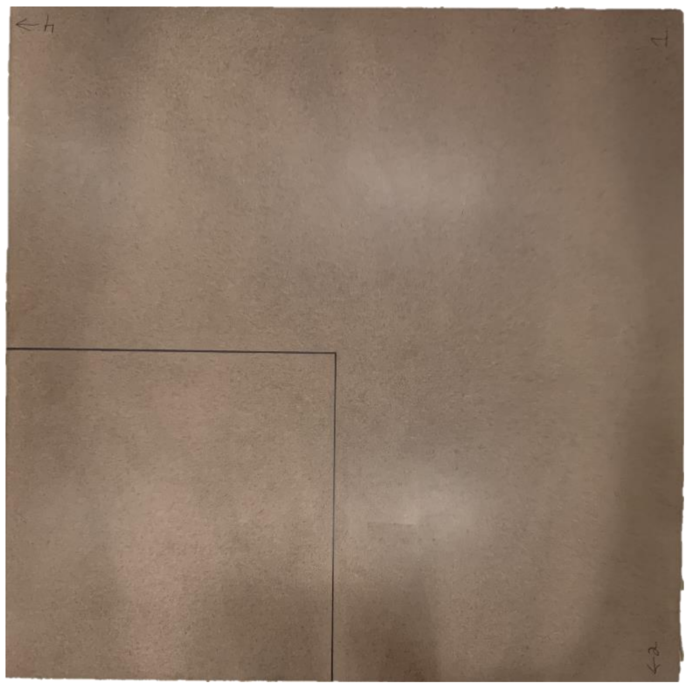


Figure 2: One Section of Play Area

Original Directional Code and Distance Algorithm:

Originally, back when we wanted to do a top-down design, I had created a program to determine the closest ball in a given set of xy values. This was when we had the idea that a Pixy would be above the play area and send signals to a microcontroller onboard the car to direct it. I was in charge and designed most of the program for this early model, including a directional value that would tell the system at what angle the ball was from the car. The tasks that remained for this early version included integrating motor commands to handle moving the car and determining the direction of the car. I had started to do this code with the onboard accelerometer/gyroscope from the Arduino Nano 3 IOT, but this model was scrapped once the group decided to switch to an onboard camera model instead of a top-down model.

As we moved away from this model our idea with the onboard camera was that we would spin around first to gather all ball data points into the system and go to the closest one. Using this idea, we needed a system that could tell how far away a ball was from the car. This led me into research on optical measurements, focal length studies, and related materials. I spent a few weeks studying this material, calculating the Pixy2's focal length, and confirming the results before we decided to simply go for the first ball the car sees. Equation 1 is the fruits of this labor and research, as I was able to determine the true focal length of the camera as it was not a specification that I could find in the documentation for the Pixy2. This work has no bearing on the current project, but it does show where I spent some of my time.

$$Distance = \frac{focal\ length\ x\ real\ height}{pixel\ height}$$

$$Focal\ Length = \frac{Distance * Pixel\ Height}{real\ height}$$

$$Focal\ Length = \frac{152.4mm * 2pixels}{10.16mm} = 30mm$$

Equation 1: Focal Length measurement.

- Mohammed Alajaji

Vision System:

The vision system is very important as it identifies and transmits information about objects in its field of view. This information can then be translated into motions and actions to capture targeted objects. The project without the vision system would be blind and not capable of working.

At the very beginning of the course, the idea of having vision system was to have a camera that covered the whole play area. The camera was required to determine the location of the balls and the RC car. Later, we realized this idea was going to cost us time and money because it needed two microcontrollers one in to control the RC car to capture the balls. The other one was needed to be connected to the camera and sent the data via Bluetooth from the camera to the microcontroller that controlled the RC car. Therefore, we decided to mount the camera to the Hippautonomous car and connect it to the microcontroller that in the Hippautonomous car.

My task at the begging of the project was to look up for a camera. There were many types of cameras such as webcams and CCTV cameras that could have done the job. However, these kinds of camera needed to be manipulated and programed to be able to detect and track an object. As well as the cost was a little bit high for the high-resolution ones. After surfing the internet for a cheap and easy camera to configured, the Pixy2 camera costed only 59.90, and it designed to detect and track objects. The Pixy2 camera was chosen to be the source of the vision system for this project.

Pixy2 Camera:

The Pixy2 camera (Figure 3) is useful for all kinds of vision-based projects. It can provide information needed for making navigation decisions, tracking, and sorting objects, identifying patterns, avoiding obstacles, and detecting motion.

Figure 3: Pixy2 Camera

The Pixy2 camera provides ample data to the microcontroller to accomplish the needed tasks. For instance, the Pixy2 sends the object's xy-coordinates, height, width, angle, color, and tracking index number when it detects the object. The Pixy2 camera can determine the position of the object, the type of the object, and the distance between the object and the vehicle.

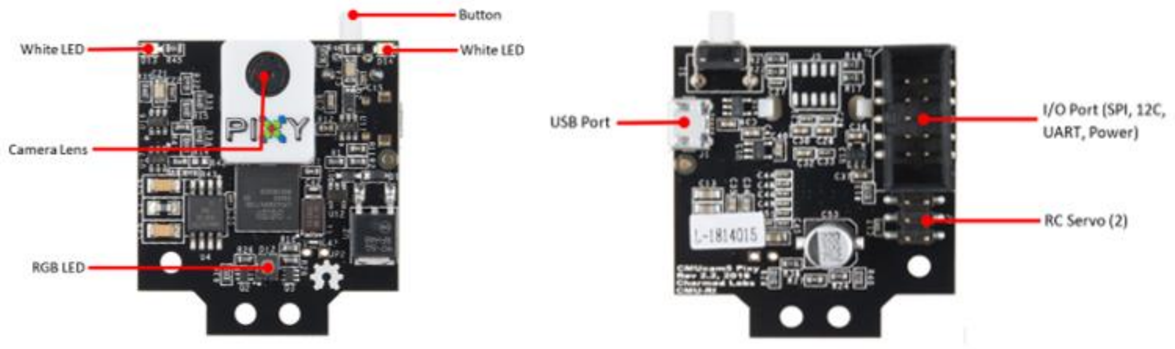
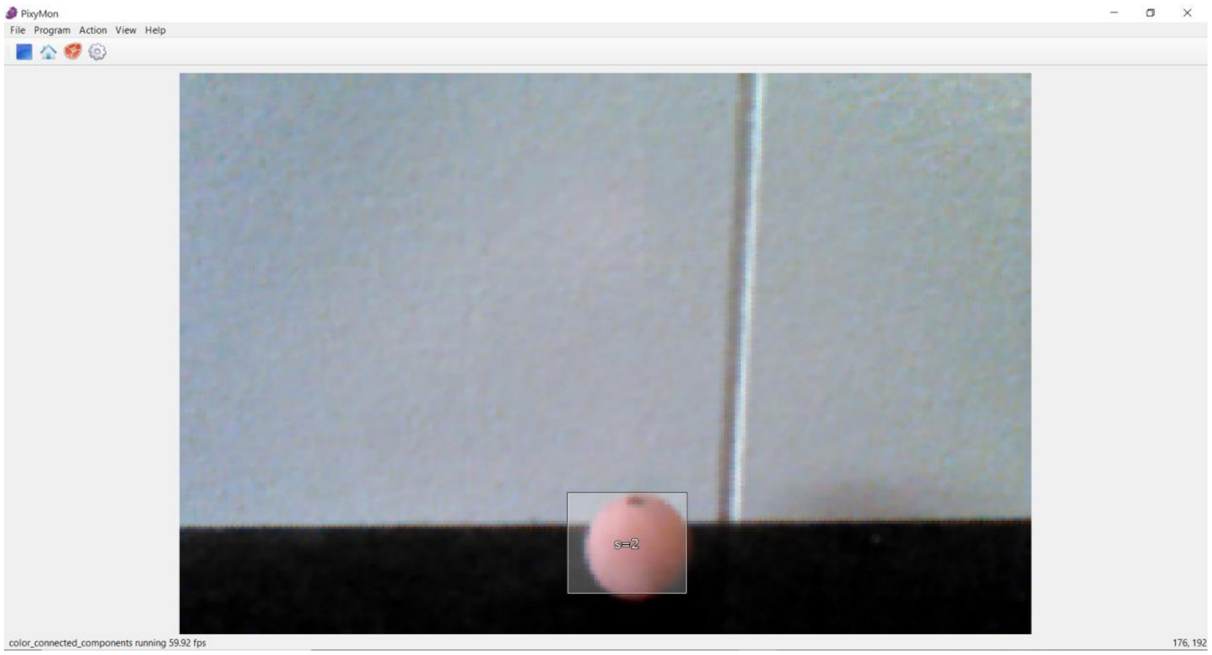


Figure 4: Front and Back Views.

PixyMon:

PixyMon is the configuration utility for Pixy2 that runs on computer [1]. On PixyMon, I could configure the Pixy2 and to train it to the balls and fine-tune it for better detection and recognition. Furthermore, PixyMon provides an image video from Pixy2 camera that I used it to check if the Pixy2 identifies the balls as shown in Figure [5] and if there are noises around. The Pixy2 camera has three different modes: video mode, line tracking mode, and color connected components (CCC) mode. This project requires only the color connected components (CCC) mode, and it can be selected on the PixyMon.



color_connected_components running 59.92 fps

Figure 5: PixyMon Application.

<u>Pixy2 Mounting:</u>

Since the beginning of the project, placing the camera was a concern because of the lighting around, the camera's resolution, and noises. I went through many ideas to mount the Pixy2 camera to the Hippautonomous car in order to get a better vision and less noises. One of the ideas was a pan/tilt design (as shown in Figure 6) in front of the car, but it was not a good idea because of the weight and power. It had at least one servo and it weighted more than 50 grams.

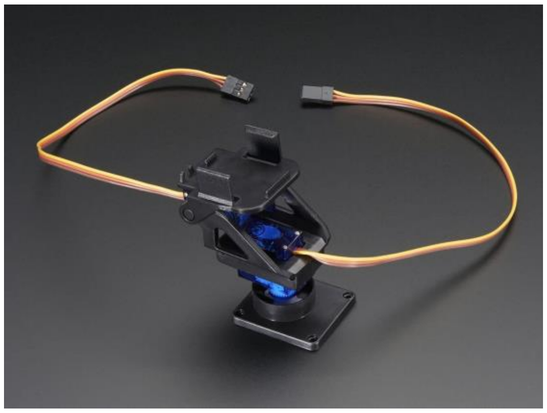


Figure 6: Pan/Tilt Mount

Therefore, I ended up using a wooden stick, and I attached half of it to the front of the Hippautonomous car under the chassis and installed the Pixy2 camera in the top of the stick. However, Laura designed the final mounting stick because it was interfering with the front gate mechanism design that Laura had designed. In the final assembly, I helped Laura in attaching the stick and installing the Pixy2 camera (Figure 7).

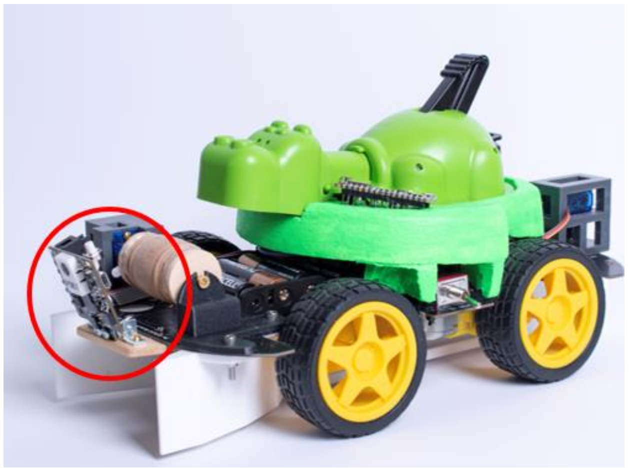


Figure 7: Pixy2 Camera Mount

Noise Reduction:

The main two issues in the project were faced noises and lighting, they impacted the Pixy2 camera detection and tracking. Minimizing the noise to a limit that not affecting the Hippautonomous performance was a challenge. And it was hard to be done because of the resolution of the Pixy2 camera and the size of the balls. The noise reduction was done in too many ways, but in this report, I will highlight only the two most effective ones. Both were done code wise. The code uses the *Pixy2* library in Arduino IDE which provides "pixy.ccc.getBlocks()" function. The function gets all detected blocks' (objects') data (such as xy-coordinates, signature no, index, … etc.) in the most recent frame [1X]. It arranges the detected objects by their size and the first version of the code was designed by James to select the first object (the largest). Most likely, the object that has the largest size is a noise, it might be a table, a chair, or a T-shirt of an audience. I suggested to select the closest detected object instead of the largest. James added a

For Loop in the code (see Figure 8) to select the nearest detected block (object) to the Hippautonomous car based on the y-coordinate.

```
  blocks = pixy.ccc.getBlocks();
index = 0;
float besty = pixy.ccc.blocks[0].m_y;
  if(blocks){
for (int i =0; i< sizeof(blocks)-1; i++)
{
  //search for largest y
  curry = pixy.ccc.blocks[i].m_y;
  if(curry > besty){
  index = i;
  besty = curry;
  }
```

Figure 8: For Loop code

The second feature that I recommended to add was an if condition into the code to determine the object that inside the play area and ignore the ones that out of the play area. The detected object will be excluded when its y-axis less than or equal 55, and it will be considered as a noise see Figure 9.

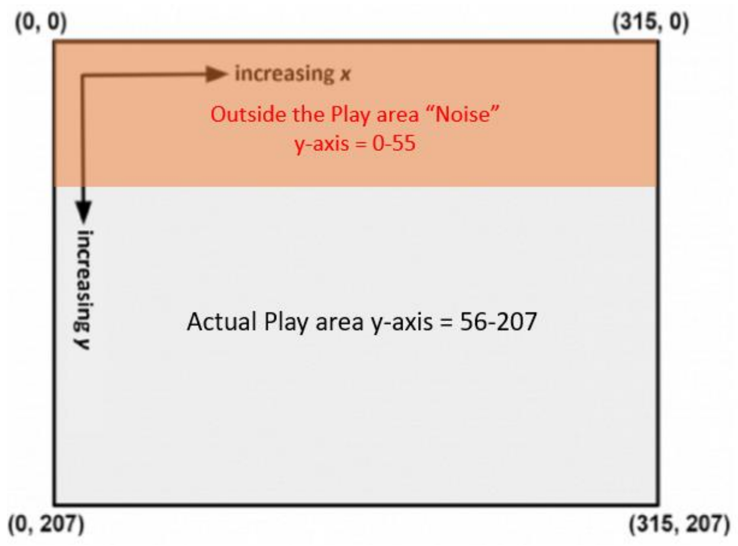


Figure 9: Noises in the image Frame of the Pixy2 camera.

Ball’s size and color:

The balls are critical component in the Hippautonomous game because they are detected and tracked by the Pixy2 camera and collected, stored, and released by the collection system. Therefore, they needed to be in the color and size that the Pixy2 camera can identify and to be in the right size and shape for the collection system. At the beginning of the project, we used the small red balls (diameter=14mm) that came with Hungry Hungry Hippos game. The Pixy2 camera could not detect and track them in a distance that more than 0.5m according to the test that I had conducted. additionally, the Pixy2 camera had more noises when I adjusted the configuration sittings to increase the distance of detection and tracking. The issue was discussed by the team and we decided to get bigger balls. Styrofoam balls (diameter= 20.32mm) were one of the options, but they were excluded because of the inconsistent shape and the light weight. The second option was to 3D print the balls, but they would cost more money and time. Laura and James found and bought wooden balls (diameter= 19.05mm) in one of the stores in Daytona beach that. The wooden balls had the proper size and weight for the game and cheaper. I painted four of them four different colors red, green, blue, and pink to test the Pixy2 camera identification see Figure 10. The pink ball (painted by a pink highlighter) had less noises, and the Pixy2 camera identified it correctly up to 0.7m. finally, the pink color was selected to be the color of the balls in the Hippautonomous game.

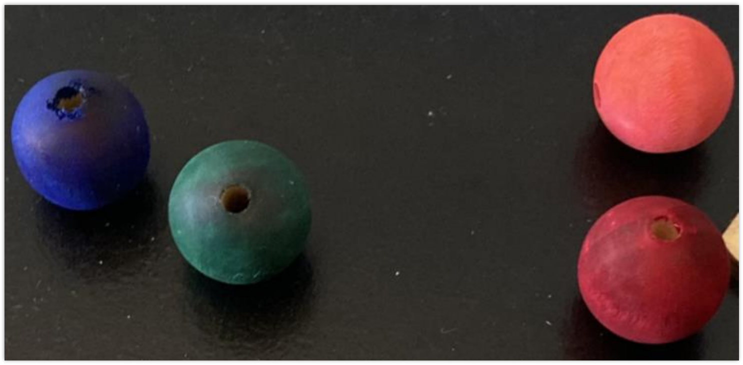


Figure 10: Red, green, blue, and pink wooden balls.

Play Area Boundaries

The Pixy2 camera has only 1296×976 resolution which makes it weak to detect the Hippautonomous’ balls from far distance (more than 0.7) and sensitive to light change and noise.

Instead of letting the Hippautonomous car moves freely, Laura came up with an idea to return the Hippautonomous car to the center of the play area. The idea was basically the Hippautonomous car spined in the center of the play area until the Pixy2 camera identified a ball, and then the Hippautonomous car went towards the ball. When the Hippautonomous ate the ball, it turned 180° and then returned to the center. It was not going exactly to the center and it missed it many times. I came up with two ideas, the first one was to hang a big object in the center of the play area which the Pixy2 was trained to identify it. The Pixy2 camera sent the object's data to the microcontroller to direct the Hippautonomous car to the center of the play area. The Hippautonomous went almost perfect to the center. However, the Pixy2 camera had a lot of noises because the position of the Pixy2 had been changed, as well as the configuration settings had been adjusted. The second idea was green tape boundaries around the ball play area (1.2m×1.2m). Once the Pixy2 camera identified the green tape, it turned around. Although it was not a bad idea, but James had a code wise idea. The idea was saving all the movements of the car in an array and reverse the array when it caught a ball. This did not work because of the size of the array that could not fit the memory of the microcontroller. Lastly, James suggested and implemented the reverse idea. The Hippautonomous car reverses back to the center in the same amount of time that it went forward. Based on that, we limited the ball play area to be (1.2m × 1.2m) and 0.6m from the center of the play area.

- Laura Jones

I was responsible for the collection system, assembly, and electrical system. I also wrote the code for servo movement and assisted in troubleshooting/optimizing the seek and capture program and the LED program. I worked with Mohammed and James during the last two weeks in testing. Mohammed helped me assemble the parts on the car. James helped me solder wires and determine wiring connections for the motors and LEDs. I also consulted with James and Mohammed while designing the collection system.

<u>Vehicle:</u>

The original car that our team chose for the project was a very small, light, and fast RC car (Figure 11).

Pros:

- Inexpensive
- Small play area size
- Fast for fun gameplay

Cons:

- Front wheel drive: forwards, backwards, left, right, causing difficulty in controlling
- Lightweight and structurally unsound, which could cause weight and balancing challenges
- Chassis low to the ground, which is challenging for collection/storage mechanism design, since the balls cannot roll under the center of the car.
- Small, providing limited space for object placement and mounting

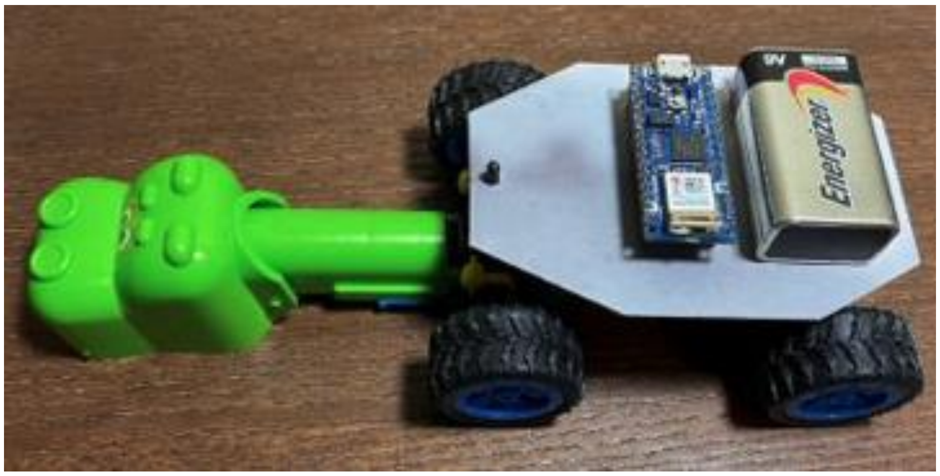


Figure 11: First car chosen to the Hippautonomous vehicle

The Hippautonomous team weighed the pros and cons of the original vehicle, listened to our professor's suggestions, and did some online research. We decided to use a different style car that was designed to be used with Arduino/Raspberry PI projects. The vehicle chosen was the smart robot car chassis kit by XIAOR GEEK (Figure 12). The vehicle provides the means of movement and transport of the entire project as the autonomous hippo moves around in the playing field.

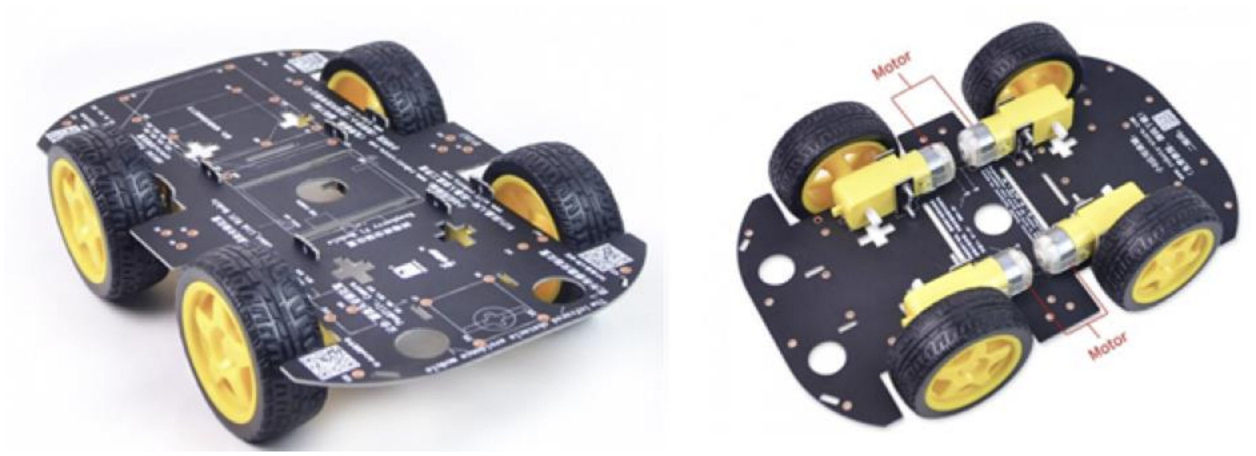


Figure 12: Car kit top side up (left) and bottom side up showing placement of motor (right) [2]

The chassis is made of a non-conductive material, so as not to cause a short between electrical pins and wires. It is large and strong enough to support all the Hippautonomous components. The chassis is equipped with multiple cutouts to allow for mounting of the Pixy2 camera, ball-collection mechanism, servos, microcontroller, batteries, and motor driver. The vehicle has four wheels to support the weight and balance of the overall system and to provide adequate clearance from the ground for the placement of a ball-collection mechanism underneath. The wheels have soft rubber treads, which are good for traction on flat, smooth surfaces. Each wheel is connected to a DC motor, which provides the option of four-wheel drive control. The four DC motors are mounted underneath the chassis adjacent to the four wheels. The motors connected to the two wheels on left side are wired in parallel. This ensures that the two left wheels rotate together at the same speed. The motors on the right side are wired the same way as the left side.

After completing the project, we had two challenges from the car chassis. The first challenge was a result of the heavy load on the car. The motors would struggle while the car was trying to spin in place on high-friction surfaces, such as industrial carpet. This made for unpredictable behavior of the car to the input commands. For instance, if a signal was sent to the motors to spin the car right for one second, the car might spin 90 deg the first time and then 45 degrees the second time. The results ended up being more consistent on smooth, low friction surfaces.

The second challenge was the PCB chassis material. It would have been much easier to design the front and back gate mechanisms if we could drill holes in the chassis. Our team was hesitant to drill into the PCB chassis because, firstly, none of us have had experience cutting into PCB material, secondly, the PCB chassis is made of fiberglass, which is harmful to breathe if using a saw or Dremel, thirdly, I had trouble finding online instructions on how to drill the large-size holes that would have been needed, and fourthly, online websites suggest not drilling holes in PCB with a hand drill, and we did not have access to a drill press at the time. For these reasons, it was decided to design the gate opening and closing mechanism to work with the pre-cut holes in the chassis of the car.

Motor Driver:

The L298N motor driver controls the speed and rotation direction of the DC motors that rotate the wheels. One motor driver is used to control the two motors simultaneously, up to 2A each. It receives low voltage control and PWM input signals from the microcontroller and applies voltage to the output pins accordingly.

There is an onboard 5V, 0.5A Enable pin that can be used to power a microprocessor [4], but we found it insufficient to power the Uno and Pixy2. The driver spins the DC motors forwards and backwards by changing the voltages across the output pins. If a positive voltage drop is applied across two pins, the wheels will turn in one direction, and if the voltage drop applied across the two pins is reversed, then the wheels will turn in the opposite direction. The A Enable and B Enable pins allow for speed control using pulse width modulation output from the microcontroller [3][4].

The goal was for the motor driver to run the DC motors at the upper end of their 3-6V operating range for fastest speed and then control the speed with PWM, if necessary. We found that the extra speed helps overcome friction when the vehicle spins in place. In order to meet the goal, the L298N motor driver was powered by five AA batteries in series (7.5V total). The approx. 2V drop between the input power pins of the motor driver and the output pins to the motors ensures the voltage across the motors is less than 6V.

Ball Collection Mechanism:

There were three original design ideas for the collection mechanism. The first idea consisted of two servos with attached arms that would move in a reverse-pinball machine motion to brush the ball up a slight ramp and into a holding area. A flap in the front would open before the ball got brushed in and then close behind the ball. For release of balls, the flap would open while the car drives backwards. The second design idea consisted of a single-wide brush, connected to a servo, that would brush the ball into a collection area using a single forward rotational movement. The ball would roll up a slight ramp and then drop into the holding area. For release of balls, the flap would open while the car drives forwards. The third idea combined elements from the first and second ideas.

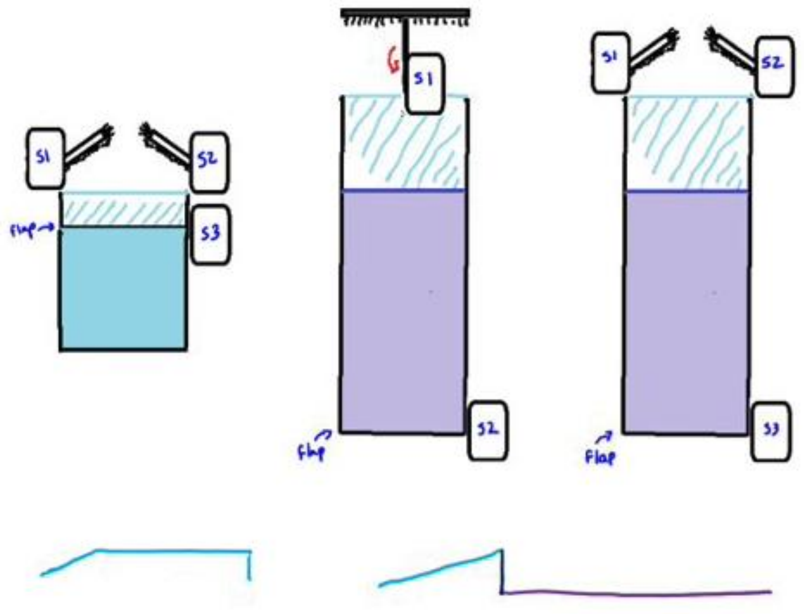


Figure 13: Original collection design ideas.

After some consideration, it was decided to not use any of the original three collection system designs and to choose a simpler, more reliable design. In the new design, a front gate lifts up in a guillotine-manner, the car drives over the balls, and then the gate lowers to prevent the balls from leaving the holding area underneath the car.

The final collection mechanism is a 3D-printed custom part that is designed to collect, store, and release the balls that the Hippautonomous vehicle collects (Figure 14). All 3D printed parts were designed to be printed using PLA filament.

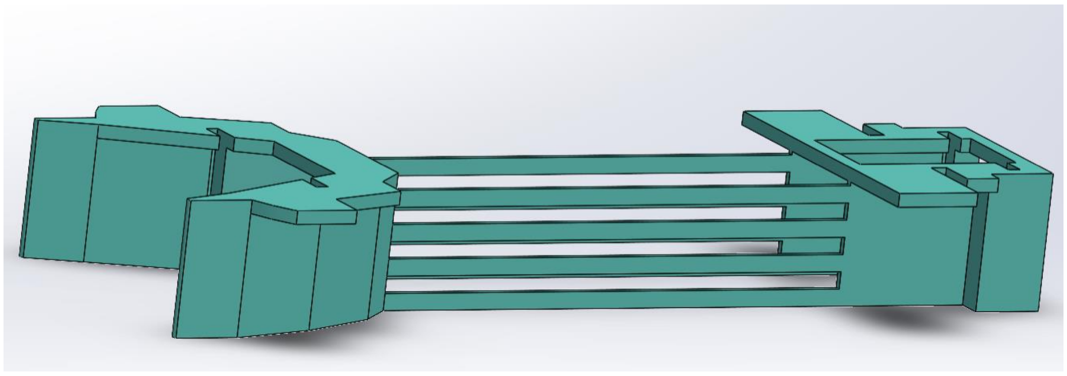


Figure 14: Model of final collection design.

The collection system was designed to provide as much room for the ball entry and storage as possible without contacting the motors that are mounted under the chassis. The collection mechanism is mounted to the underside of the chassis, and it is aligned with the mounting screw holes that were pre-drilled in the chassis. The holes were drilled into the collection system mounting platforms after everything was 3D printed, in order to ensure proper alignment with the preexisting holes in the chassis (Figure 15).

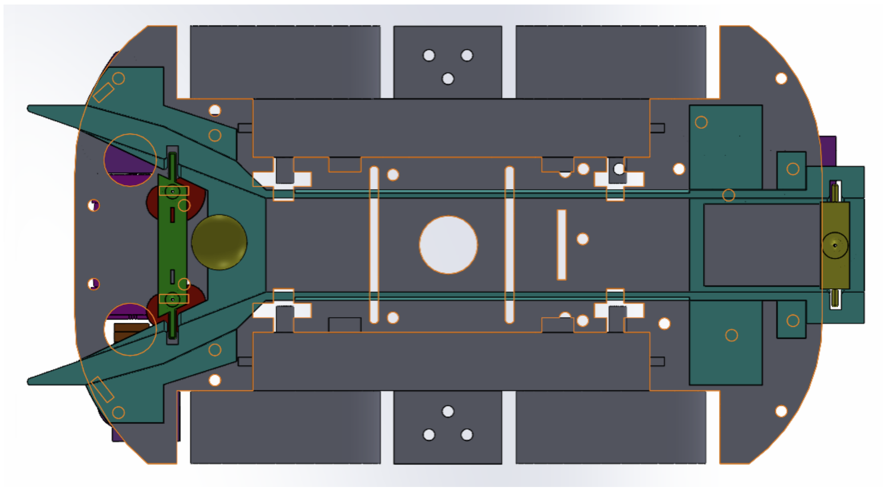


Figure 15: Alignment of collection system under the chassis.

The collection mechanism consists of a wide (92mm) opening in the front for collection. The front opening tapers down to a 34mm wide channel that runs through the remaining length of the chassis. The tapered section is the collection area, and the channel is the holding area. The collected balls freely roll with the movements of the vehicle while in the holding area during gameplay. The collection system was designed to be light in weight while maintaining structural integrity. The rectangular slits along the length of the sidewalls of the holding area reduce the weight of the overall collection mechanism and provide a means of viewing the balls in the channel. The top rail is 5mm below the underside of the chassis, to provide means for feeding the motor wires to the upper side of the chassis (Figure 16).

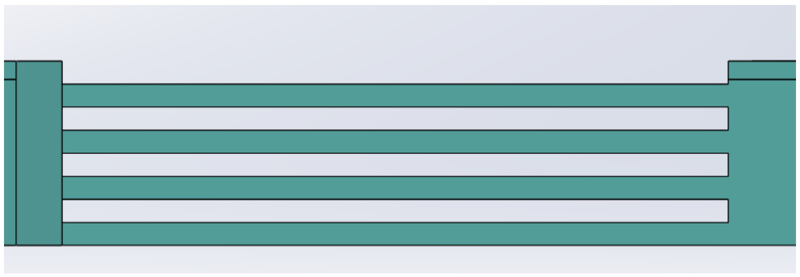


Figure 16: Sidewall design of the holding area.

A cardboard prototype was made per the 3D model to ensure proper fitting and alignment under the car (Figure 17). After making the cardboard prototype, I noticed there was some misalignment between the mounting platforms on the collection system and the holes in the chassis towards the back of the car. The model was corrected before 3D printing.

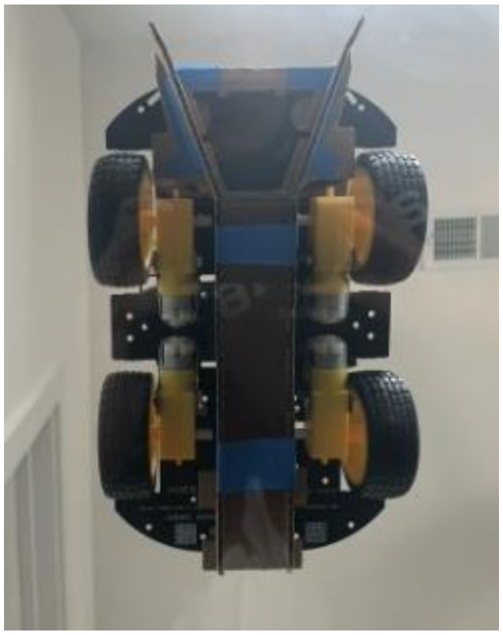


Figure 17: Cardboard prototype. Picture was taken by James H.

There are two channels notched in the sidewalls in the front and back of the collection system, which were designed to guide the front and back gates into a vertical movement when pushed by springs and pulled by servo movement. Each two-channel pair is 5mm longer (in total) than its respective gate and approximately 2 mm wider than the sections of the gate that slide vertically inside the channels (Figure 18).

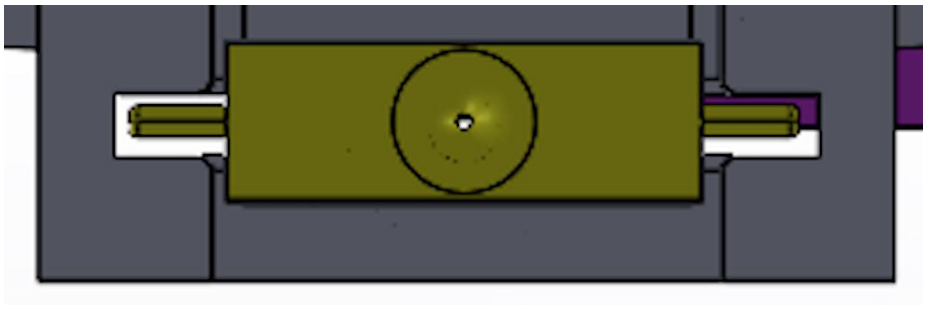


Figure 18: Placement of the back gate in the two back channels (bottom view).

Gates:

In order to contain the balls in the holding area, there are two gates, one towards the front of the vehicle and one in the channel towards the back of the vehicle. The gates are designed to provide a means for the ball to transfer from the collection area in the front to the holding area under the car, provide a barrier to trap the balls in the holding area during gameplay, and to provide means for the balls to leave the holding area after the game is over. The gates slide along vertical grooves in the sidewalls of the collection system. The open position is when the top of the gate contacts the underside of the chassis, and the closed position is when the bottom of the gate is low enough such that the balls cannot pass through the gate. While the gate is in the open position, a ball can easily roll under it without making contact.

The front gate is located towards the front of the car in the tapered section of the collection mechanism (Figure 19). There were a few motivations for the front gate placement. First, it is located just below two rectangular slits in the chassis. The rectangular slits provide access for the control mechanism that is on top of the chassis to lift the front gate that is below the chassis. Second, it is located towards the front of the car, which increases the capacity of the ball holding area. Third, it is in a sturdy section of the collection mechanism, where the channels would not cause any structural issues within the 3D printed collector.

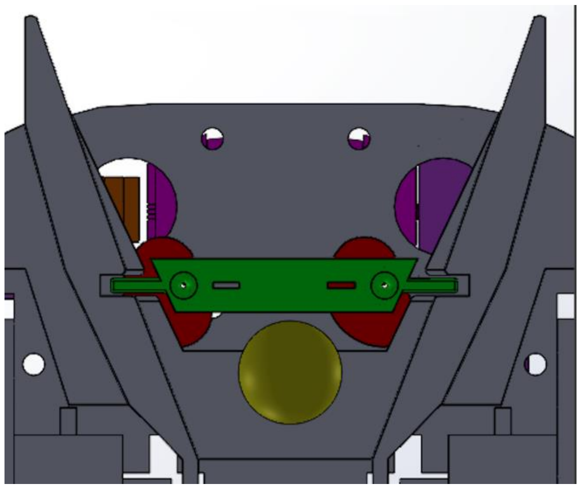


Figure 19: Front gate location (bottom view).

The original design for the front gate was simple (Figure 20). A thin string would be tied around each of the two horizontal cylindrical sections, one on each side. A servo would spin a cylindrical object that would evenly pull the strings to lift the gate. The servo would spin in the opposite direction to let gravity lower the gate.

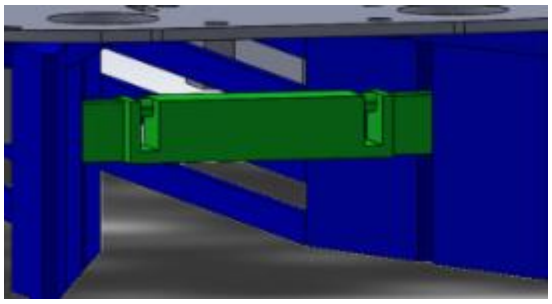


Figure 20: Original front gate design in the channels (front view)

Our professor shared his concern that the front gate would be too light, and that gravity might not reliably pull the gate down as far as needed. After our class, Mohammed suggested using a spring to push the gate down. That seemed to be a viable option that we ended up using, and it ended up introducing technical challenges in the design of the front gate that were overcome in the design process. A few of the technical challenges include:

- The gate needed to raise high enough to allow the ball to pass under it, which means the length of the compressed spring needed to be considered.
- When compressing, a spring naturally tends to bend outwards or inwards, which would cause strain and friction between the lightweight gate and the channel walls.
- The spring needed a solid base on each end to press against that is strong enough to withstand the spring pressure.
- A structure needed to prevent the spring from shifting horizontally during compression and releasing.
- The inner diameter of the spring needed to be wide enough to enable placement of a guide for the spring to travel against. The guide needed to be large enough to contain a 1mm diameter hole for the spring and have enough mass to be rigid during gameplay.
- No parts of the gate should stick below 22mm from the ground when the gate is in the up position.
- The gate had to travel 15mm. The spring's uncompressed length minus compressed length should be minimum 15mm.
- Two springs were needed for the front gate (utilizing the two rectangular openings in the chassis). The two springs needed to be compressed evenly to avoid binding in the collector channel.
- The two springs greatly increased the force needed to be overcome by the servo. Springs with lower spring constants needed to be used in order to control gate movement with a light and small servo.

In researching springs, I started looking in the direction of click pen springs. James helped me look for good pen springs to use, and we settled on the Pentel Energel springs because they most closely met the requirements: 4.5mm inner diameter, 7mm compressed length, and 22mm uncompressed length. I felt the least amount of resistance when compressing it by hand of all the pen springs I tested. I determined the spring constant by placing the spring on a mass scale, zeroing the scale, and then pushing down on the spring until it reached the fully compressed position. I did it several times and took the average mass, which was approximately 500g each.

Front Gate Design:

The overall shape of the front gate is such that it will not contact the tapered section of the collection mechanism. Two 1mm diameter holes run vertically inside the gate, one on each side. The purpose of the hole is to provide a means for a strong, thin string, such as a fishing line, to thread through. The thin string was used to connect the gate underneath the car to the lifting assembly on top of the car. There are two deep fillets on the bottom of the gate, centered around the 1mm holes (Figure 21). These allow the thin string to be tied in a knot below the gate without affecting the ball movement under the gate when in the up position.

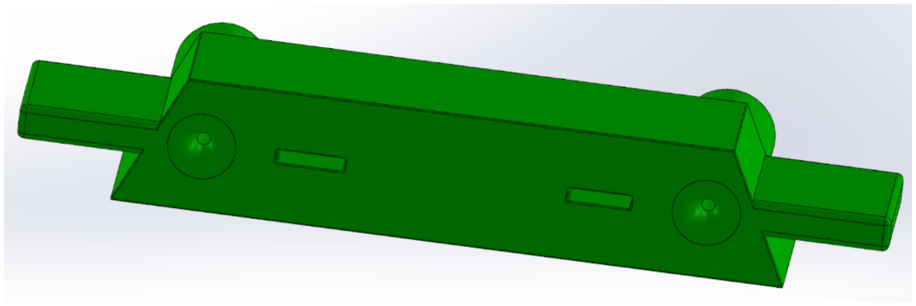


Figure 21: Final front gate design (bottom view)

The top of the gate was designed to have outside cylindrical guides for the springs to sit inside, a strong, flat surface for the springs to compress against (which are level with the top surface of the gate), and hollow areas for the spring guide to travel inside (Figure 22). Two rectangular slits are cut into the front gate's large cylindrical channels in order to provide means for air movement when the gate is raised or lowered.

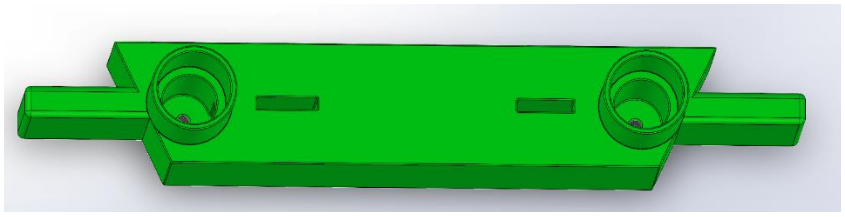


Figure 22: Final front gate design (top view)

Adding springs to the system required an additional component to be included with the front gate (Figure 24). The spring alignment component in the figure has a large flat area that is designed to be glued to the underside of the car. The purpose of the flat area is to create an even surface for the spring to press against when compressing. Since it was glued to the car chassis, the load applied to the flat area when compressing the spring transfers to the chassis. For this reason and the limitation of the space under the car, the platform was designed to be 1mm thick. There is a cylindrical protrusion on the top side of the flat area, which is designed to feed through the 5mm wide rectangular holes in the chassis. I designed it in such a way to make the OD as small as possible while maintaining structural integrity during gameplay, in order to facilitate installation. The bottom cylindrical protrusion was designed to be as large as possible while fitting inside of the spring. Its purpose is to be a guide for the vertical compression of the spring and to prevent the spring from bending during compression. It is like the design inside of the Pentel pen.

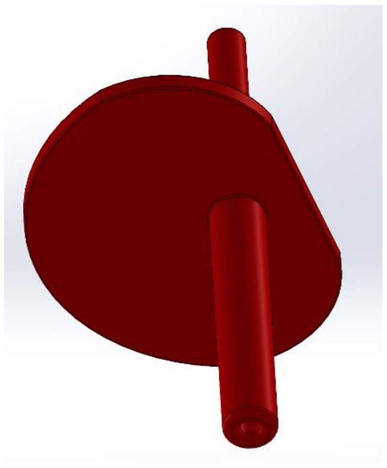


Figure 23: Spring alignment component (bottom view)

The spacing between the 1mm diameter hole inside the spring alignment component, the outer diameter of the spring guide, the inner and outer diameters of the springs, the diameter of the inner and outer front gate holes, and the inner diameter of the outside guide on the front gate are very tight.

Two of the spring alignment components are required, one for each spring. Looking back, I should have connected the two pieces together with an exact distance apart in the 3D modeling instead of two separate pieces because manual alignment during assembly was very difficult.

It was quite the challenge to design a means to guide each spring, so that the spring does not bend during compression while at the same time, avoiding protrusions that would affect gameplay when the gate is in the up position. A two-part system was utilized to combat spring bending during compression. The spring guide on the spring alignment component would provide alignment inside the upper half of the spring when it is fully extended, and the outside guide on the gate would provide alignment outside of the bottom half of the spring when fully extended. Also, the system was designed to prevent lowering of the gate below the extended length of the spring. As the gate lifts, the spring rides along the inner spring guides of the red piece while compressing into the outside spring guide of the gate (Figure 24).

The protrusions that stick up above the chassis ensure vertical alignment of the section of string that travels below the chassis.

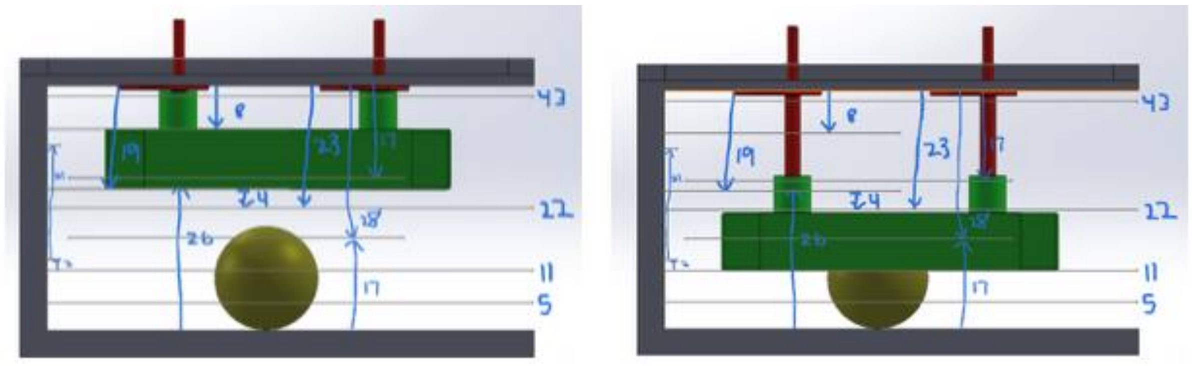


Figure 24: Front gate lifting mechanism (side view on left and top view on right).

Front Gate Lifting Mechanism

The objective of the front gate lifting mechanism is to evenly pull the gate straight upwards in order to allow the gate to slide easily along the vertical channels in the collection mechanism. The front gate mechanism consists of a servo, a cylindrical object in horizontal configuration with an edge directly above the gate, and a thin, strong, flexible string, or fishing wire, that gets tied in a knot on one end, threaded up through the 1mm hole in the bottom of the gate, up through the center of the spring, and into the 1mm hole of the red piece, and then gets wrapped around the barrel a couple times and secured to the barrel with hot glue (Figure 25). Two string/spring systems are used, one for each side.

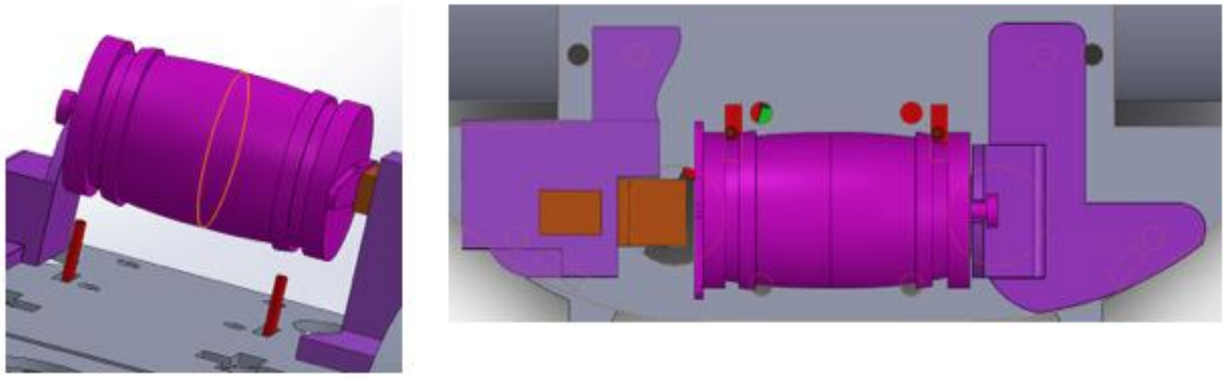


Figure 25: Front gate lifting mechanism (side view on left and top view on right).

While working on the design, it was decided to use the rotating motions of a servo to mechanically pull the gate upwards. One challenge I faced is the servo spins in a circle and does not do vertical movements. In order to overcome this barrier, a cylindrical object in a horizontal position was used with the idea that as the barrel rotates in a direction away from the front gate, a thin, flexible string would wrap around with the barrel while maintaining vertical alignment to the front gate system below at the point where it first contacts the edge of the barrel.

The cylindrical object needed to have a length at least equal the spacing between the two rectangular cutouts of the car, on which the springs and gate were centered, and the circumference at the location of the thin strings had to be equal and a be at least of the length of gate travel (15mm) divided by two, because the servo can only rotate 180 degrees. The weight of the cylindrical object also had to be considered in order to minimize the total weight on the car. In the beginning of the design process, it was unclear whether we would have access to a 3D printer. So, while James and I were at a craft store looking for wooden balls that could work for our project, James noticed a pack of unpainted wooden barrels. I measured them with my tape measure and determined that they fulfilled the design requirements for the cylindrical object needed. I had to add on edges in order to prevent the string from slipping off during gameplay. I had a paint stick at the house that I used to cut two circular shapes to put on each end of the barrel. I went shopping for wood screws that had a smooth, non-threaded section near the head. The non-threaded section would rest in a rounded slot and spin with the movements of the barrel. The 3D printed piece for the round slot in which the screw sits, wsa the only piece that had to be modified after printing because the slot was located a few millimeters too high. The piece has two mounting platforms that are aligned with holes in the car. This is also the only piece that is 3D printed using carbon fiber filament.

The challenge for the servo side was how to connect the barrel setup to the servo. I figured there would be a decent amount of torque used to spin the barrel, so the connection needed to be relatively strong. I decided to use the attachment that was provided with the servo and secure the attachment to the barrel with two wood screws, one on each side of the shaft of the servo (Figure 26). The screws would also secure the circle-shaped paint-stick to the end of the barrel.

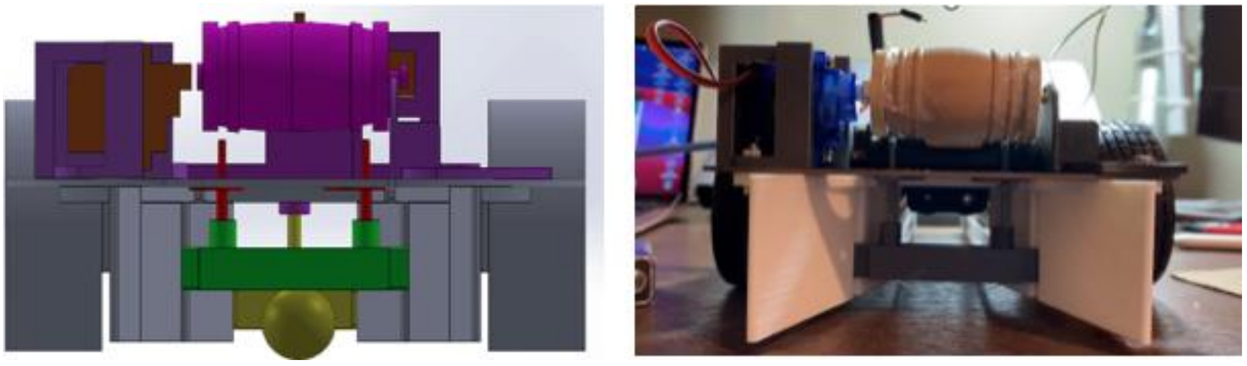
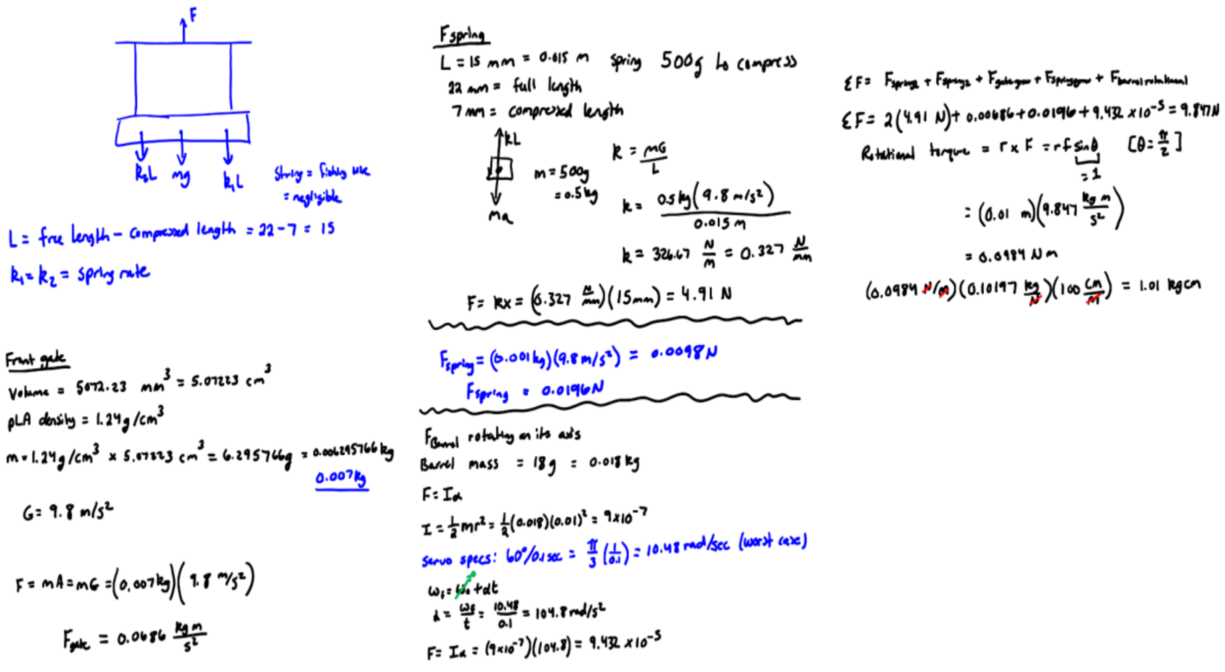


Figure 26: Front gate lifting mechanism (front view).

The SG90 servos were purchased, based on the torque calculations of the first gate design with no springs. James helped me calculate the torque required to rotate the front gate lifting mechanism, which is equal to the stall torque of the servos that we purchased (Figure 27). We did some testing, and one servo can lift the gate with minor strain.


F
k_2L
mg
k_1L
String = fishing wire
= negligible

L = free length − compressed length = 22 − 7 = 15

$k_1 = k_2$ = spring rate

Front gate

Volume = $5072.23\ mm^3 = 5.07223\ cm^3$

pLA density = $1.24 g/cm^3$

$m = 1.24 g/cm^3 \times 5.07223\ cm^3 = 6.295766 g = 0.006295766 kg$

$\underline{0.007 kg}$

$G = 9.8\ m/s^2$

$F = mA = mG = (0.007 kg)(9.8\ m/s^2)$

$F_{gate} = 0.0686 \frac{kg\ m}{s^2}$

F spring

$L = 15\ mm = 0.015\ m$ spring 500g to compress

22 mm = full length

7 mm = compressed length

kL
ma

$m = 500g$
$= 0.5 kg$

$k = \frac{mG}{L}$

$k = \frac{0.5 kg (9.8\ m/s^2)}{0.015\ m}$

$k = 326.67 \frac{N}{m} = 0.327 \frac{N}{mm}$

$F = kx = (0.327 \frac{N}{mm})(15 mm) = 4.91\ N$

$F_{spring} = (0.001 kg)(9.8\ m/s^2) = 0.0098 N$

$F_{spring} = 0.0196 N$

F_{Barrel} rotating on its axis

Barrel mass = 18 g = 0.018 kg

$F = I\alpha$

$I = \frac{1}{2}mr^2 = \frac{1}{2}(0.018)(0.01)^2 = 9 \times 10^{-7}$

servo specs: $60°/0.1 sec = \frac{\pi}{3}\left(\frac{1}{0.1}\right) = 10.48$ rad/sec (worst case)

$\omega_f = \overset{0}{\cancel{\omega_0}} + \alpha t$

$\alpha = \frac{\omega_f}{t} = \frac{10.48}{0.1} = 104.8\ rad/s^2$

$F = I\alpha = (9 \times 10^{-7})(104.8) = 9.432 \times 10^{-5}$

$\Sigma F = F_{spring1} + F_{spring2} + F_{gate\,gravity} + F_{spring\,gravity} + F_{barrel\,rotational}$

$\Sigma F = 2(4.91\ N) + 0.00686 + 0.0196 + 9.432 \times 10^{-5} = 9.847 N$

Rotational torque $= r \times F = rf\underbrace{\sin\theta}_{=1}$ $[\theta = \frac{\pi}{2}]$

$= (0.01\ m)\left(9.847 \frac{kg\ m}{s^2}\right)$

$= 0.0984\ N\,m$

$(0.0984\ \cancel{N}/\cancel{m})(0.10197 \frac{kg}{\cancel{N}})(100 \frac{cm}{\cancel{m}}) = 1.01\ kg\,cm$

Figure 27: Torque calculations for front gate servo.

<u>Front Gate Servo Support:</u>

The objective for the front gate servo support was to securely hold the servo in a horizontal position while it raises and lowers the front gate. It needed to be lightweight, if possible, align with the mounting holes in the chassis, and provide room for wiring. I inserted as many holes as possible and extended the base in order to have two mounting contact points. The front gate servo support was designed so that the servo can easily slide into it with minimal side to side movement after insertion. The servo is supported on all four sides and screwed into the fixture at the two mounting points on the servo (Figure 28).

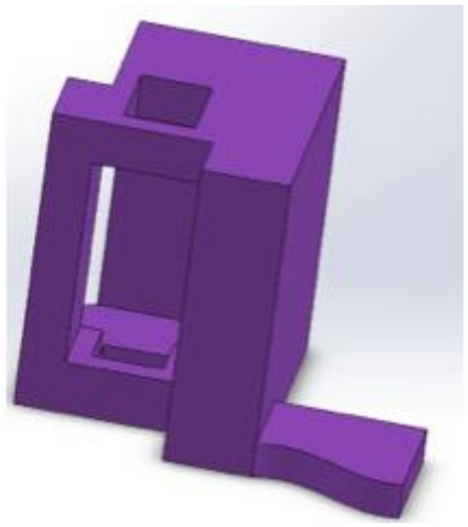
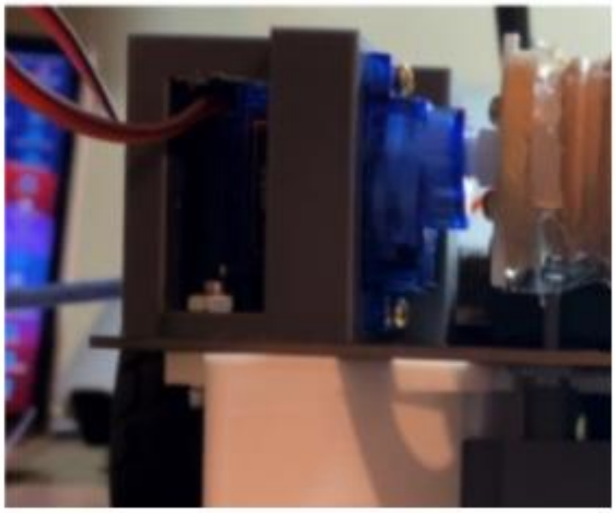



Figure 28: Front gate servo fixture.

Back Gate:

The back gate is located at the back end of the car. The collection mechanism was extended beyond the length of the car in the back to facilitate the lifting mechanism for the back gate. The back gate is a simpler design than the front, using only one Pentel Energel spring, in the middle of the top surface of the gate. Like the front gate, the sides are shorter in width and length than the vertical channels they travel on. Also the edges are rounded to prevent binding. It has a large fillet in the bottom, that is centered around the 1mm hole that is in the gate. Unlike the front gate, the back gate has a cylindrical protrusion with an outer diameter that is slightly less than the inner diameter of the spring (Figure 29). The protrusion was designed to guide the spring, during compression and extension, so that the spring does not buckle or bend during movement. There is a collar at the base of the shaft on the top of the back gate, which is designed to prevent horizontal shifting of the spring. The collar is less than 1mm shorter in diameter than the inner diameter of the spring, so that the bottom of the spring can slide over the collar and stay in place.

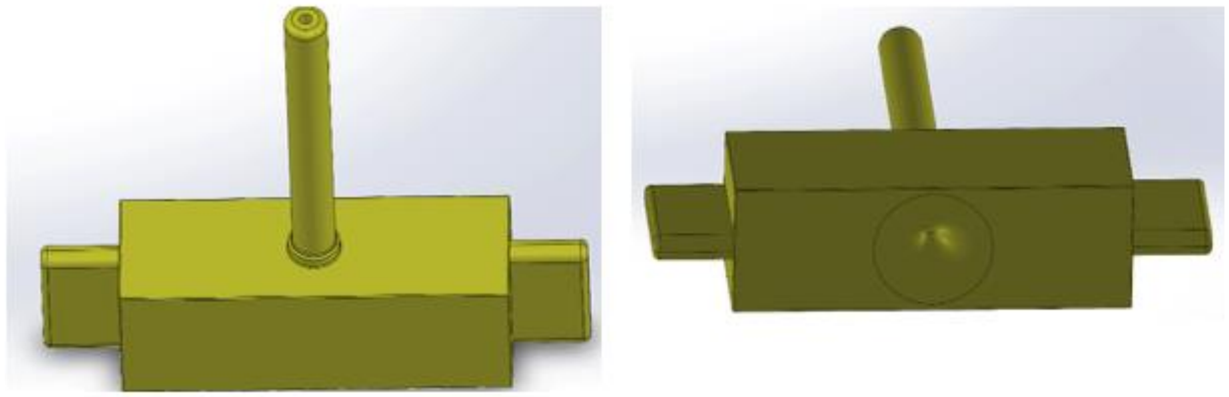


Figure 29: Back gate top view (left) and bottom view (right).

The top of the spring slides into an outside spring guide that is 8mm deep, and the top of the spring contacts a flat surface that prevents the spring from travelling up the tube with the cylindrical guide as the gate lifts (Figure 30). The outside guide holds the upper section of the spring in place and prevents it from shifting horizontally during compression and release.

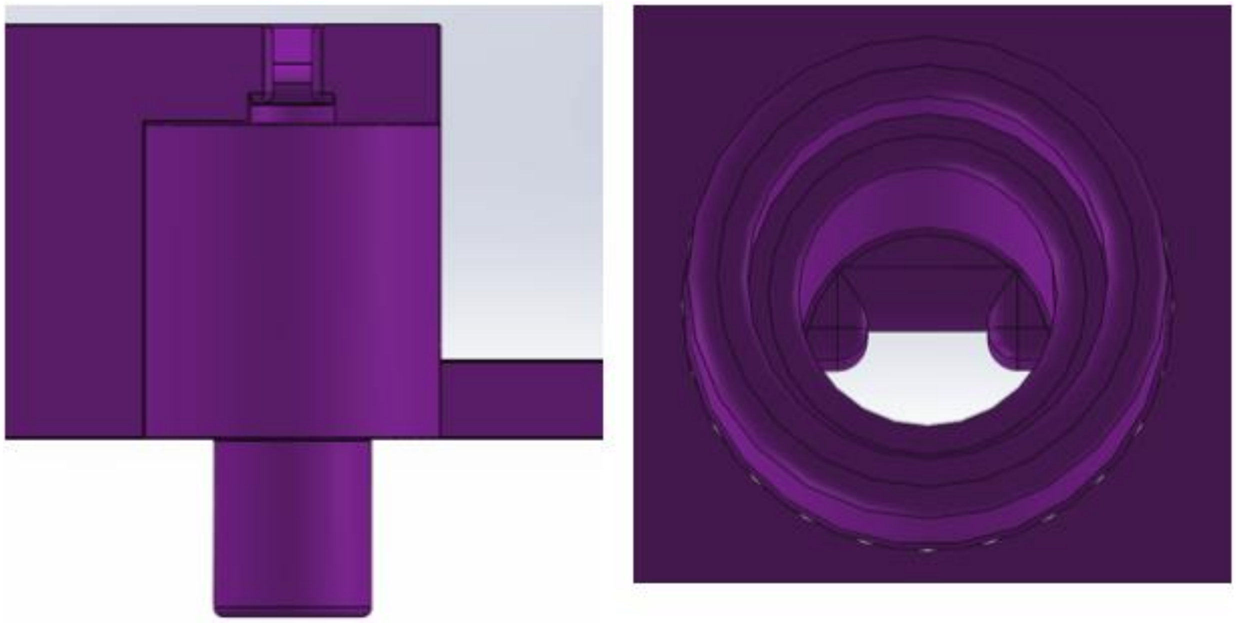


Figure 30: Back gate fixture.

Similar to the front gate, the vertical motion of the back gate is 15mm. As the gate is lifted, the vertical shaft on the gate travels into the cylindrical section of the back gate servo support. When the back gate is lifted, the collected balls can easily pass under the gate as the car drives forwards, and when the gate is in the down position during gameplay, the balls are kept in the holding area (Figure 31).

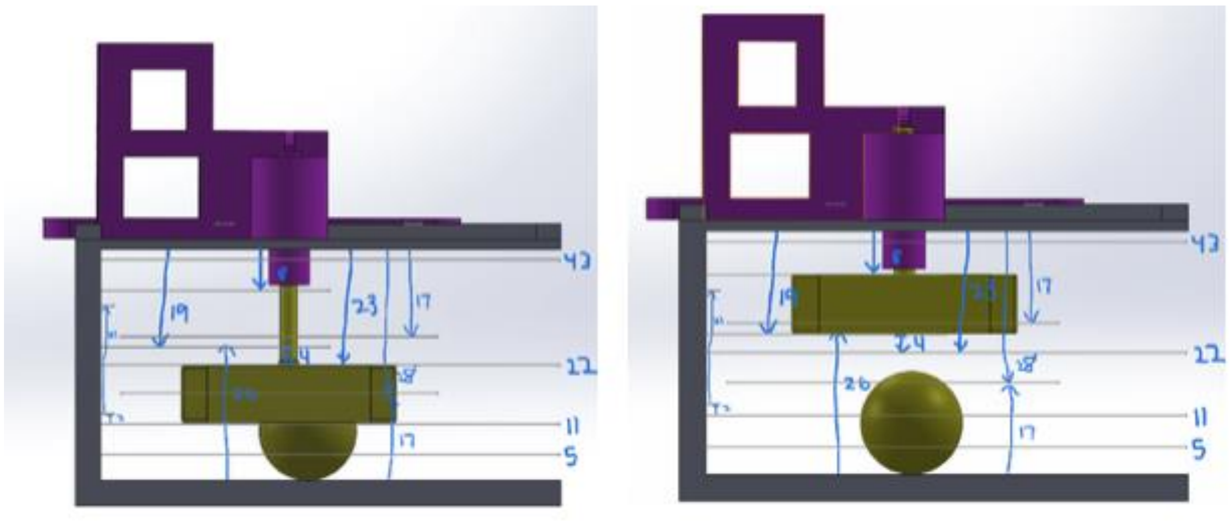


Figure 31: Back gate lowered to show ball blocking and raised to show ball clearance.

Back Gate Lifting Mechanism:

The back gate lifting mechanism consists of one spring, a thin, strong, flexible string, such as fishing wire, and a servo. The string is tied into a knot on one end. The other end is threaded through the 1mm hole in the bottom of the back gate, up through the cylindrical protrusion, around a rounded notch in the back gate servo support and tied to the servo attachment. The circumference of motion of the servo attachment had to be equal to the vertical movement of the gate (15mm) times four. The servo needs to be capable of lifting the gate with a 90-degree rotation. The short attachment provided with the servo meets the minimum radius requirement.

The servo needed to be raised above the rounded notch in order to effectively lift the gate with a 90-degree spin. The rounded notch divides the string into the vertical alignment below the notch and the diagonal alignment between the notch and the servo. The notch is lined up in a way that it is a little bit off center to the 1mm hole in the back gate. This considers the width of the string, and centers it with the cylindrical tube for the shaft to travel through, which is centered with the vertical channels on the sides of the collection system (Figure 32).

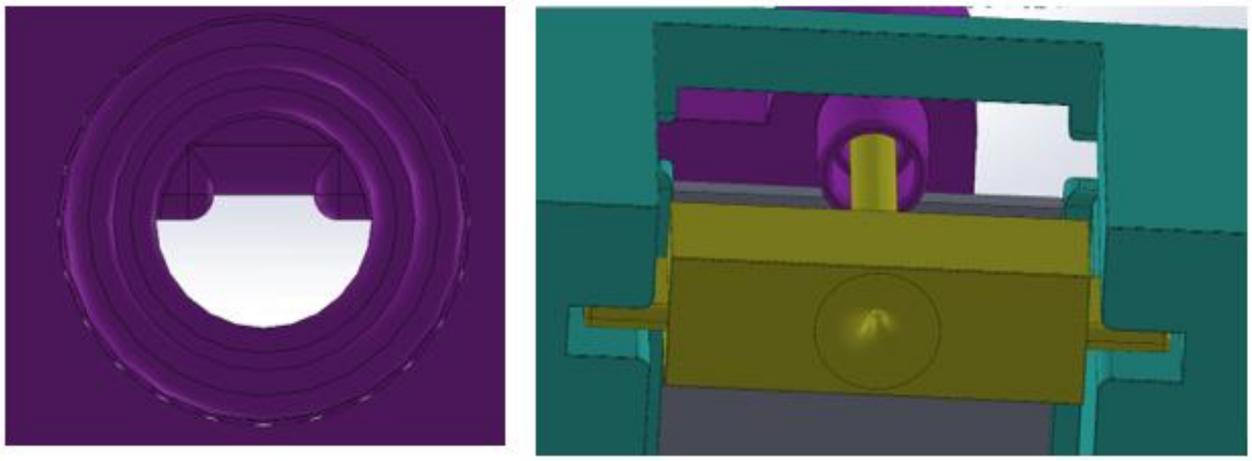


Figure 32: Back gate alignment.

The back gate servo mounting structure integrates the upper section of the spring holding mechanism, the cylindrical section for the back gate shaft to travel inside of, the rounded notch guide for the string, and the servo mounting into one 3D printed piece. The back gate servo support structure is designed to securely hold the servo in place, like the front gate servo structure. It has extensions to the base to align with three mounting holes in the chassis of the car. I made as many openings as I could in order to reduce the weight while maintaining structural soundness (Figure 33).

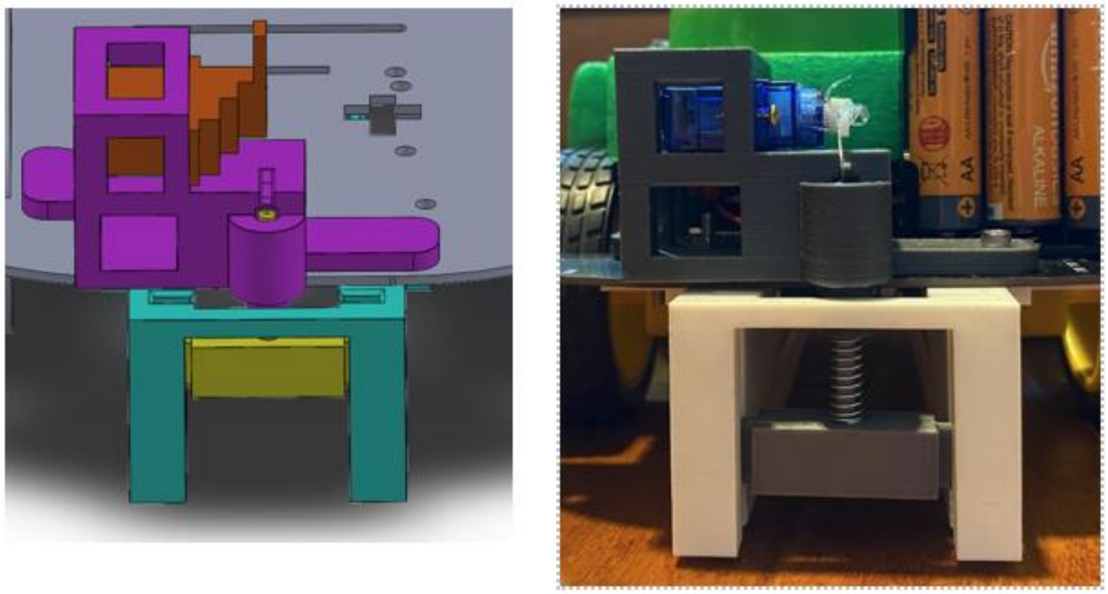


Figure 33: Back gate/ fixture assembly.

Ball Size:

The ball needed to be large enough to stay inside the holding area with the gates down and small enough to be able to roll under the gates when the gates are in the up position (Table 2).

Table 2: Ball Diameter Determination

Min ball diameter: Clearance of bottom of holding area wall: 9mm Min ball diameter = 19mm.	Max ball diameter: Width of inner walls of collection channel: 34mm Clearance under the gate in up 26mm Maximum diameter of ball = 24mm Clearance between max ball diameter and inner channel walls: 10mm

Assembly:

Once all the parts were successfully 3D printed, Mohammed, James, and I met up in a lab to assemble the car. Mohammed helped me pre-drill straight holes in the barrel and attachments using a drill press for the barrel assembly to be aligned in parallel with the top of the car chassis. We also determined the exact position of the 3D printed parts in relation to the other items that were to be secured to the car, and we drilled holes in the 3D printed components to line up with the holes in the car chassis. Mohammed had an idea to hot glue two of the double battery banks underneath the car, and James thought of using zip ties to secure the Uno and the motor driver to the chassis, since the holes did not line up where we put them. Our goal was to have one even layer of components mounted on the car if possible and to distribute the weights as evenly as possible, and to arrange the components such that the wiring terminal pairs would be relatively close to each other. Mohammed had the idea to orient the Uno in a way to make it easy to connect a cable to download code. James helped me with some of the soldering of the wires, and he had to figure out which motor wires connected to which output pins on the motor driver after the assembly. Looking back, we should have labeled the wires before removing them during assembly.

All the battery casings are hot glued to the chassis. The upper section of the front gate is glued to the underside of the chassis using superglue. Since the wide section is 1mm thick PLA material, I was concerned that the heat from using hot glue might deform it, and that piece is essential for vertical alignment of the string and the spring.

Near the end of final testing, I went back and soldered all the wires together, soldered wires to the motor driver pins and Nano pins and covered most of the solder joints with heat shrink tubing. I had to be careful to aim the heat gun away from the 3D printed PLA material.

The final weight of the car is 1049g. The car chassis kit weights 300g, and the components added to the car weigh 749g. The weight capacity of the car is 500g. Because the car is overweight, we had to be very specific with the play area surface. The motors struggle on surfaces that have higher friction than a smooth wood floor.

Hippo Base and Hippo:

Within the last two weeks of the semester, our group had the car working with the seek and find algorithm meeting our requirements. I then decided to add some customizations to our Hippautonomous car for it to look more like a fun hippo instead of a ball collecting autonomous vehicle. The goal was to cover as much of the wiring and electrical components as possible to make it look more like a finished product. In order to accomplish this, I bought a 5.6-inch diameter styrofoam ball and a heated styrofoam cutter, and I went to work carving the styrofoam. I noticed early on that I could create only two contact points between the styrofoam and the car chassis, one on each side. The back of the styrofoam would just over the heatsink on the motor driver, and the sides were notched away, so that the foam would not contact the spinning wheels. I learned that sandpaper eliminates rough edges and smooths uneven surfaces in the styrofoam.

I realized that I could not leave the batteries exposed for easy battery replacement, so I decided to velcro the styrofoam onto the chassis, so that the foam piece can be easily removed. I wanted to use the green hippo that came with the game to put on top of the styrofoam base. In order to reduce some of the weight, I cut the bottom skirt off the hippo using a dremel and left a flat area for gluing to the styrofoam. The styrofoam was notched out in the shape of the hippo, so that the hippo rests slightly inside the foam. Since I was planning to install LEDs in the hippo, I cut out a large area in the center of the styrofoam for wiring and ease of LED assembly. The black lever that gets pushed in the game, the hippo head, and hippo neck are hot glued in place in order to prevent solder joints from being pulled apart.

I consulted with James and Mohammed about what color to paint the styrofoam. We narrowed it down to two options: blue (resembling water) and glow-in-the-dark green, and then the green was the final decision (Figure 34).

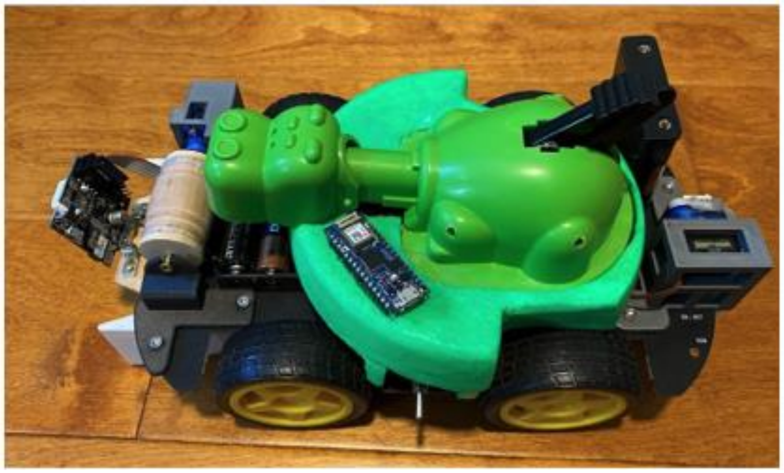


Figure 34: Finished Hippautonomous product.

<u>LEDs and Nano:</u>

When installing LEDs, there were a few requirements that I needed to keep in mind while designing the LED system.

- The LEDs draw up to 60mA each.
- Better LED effects needed to be run in a loop, so a separate microprocessor needed to be used.
- There were a small number of pwm and digital pins available on the Uno.
- It was desired for the LEDs to change color and not just stay one solid color throughout the game.
- Placement of the LEDs should be such that they can be seen from any angle.
- A finished look was desired.

My main concern was the additional power draw. Based on earlier testing, I determined that it would be best to share the power source with the Pixy2 and the Uno. I minimized the number of LEDs while strategically placing them symmetrically around the body of the hippo in such a way that the LEDs could be enjoyed throughout the entire game. Two LEDs are installed in the eyes, one on a corner of each leg, and one in the slit in the back. I drilled the holes for the LEDs using a drill press. The LEDs are addressable RGB, so that each LED can be controlled individually using one wire. Since the Nano was an extra addition after the car was fully assembled, I consulted with James and Mohammed on where to put the Nano. I suggested putting it on the outside of the styrofoam base, in order to add a “cool” factor to robotic car and to inspire people to ask questions about the electronic aspect of the car. I cut a notch in the styrofoam and made some holes to route wires from the Nano down into the styrofoam structure to hide the wires. Looking back, putting the Nano on the side of the styrofoam could have given it a more deliberate/finished look than putting it on top.

I cut LEDs from an LED strip and soldered wires to the LED terminals. The LEDs were then tested to make sure they were working correctly before being hot glued to the inside of the hippo. The LEDs need a 5V input, and they were connected to a 4AA battery bank, which was 6V. In order to drop the voltage to a safe level, I used two 1N4001 diodes in series between the positive battery bank wire and the V+ input of the LEDs. I also added a 400 Ohm resistor between the signal wire and the Nano output pin based on Adafruit recommendations [11] to prevent damage to the first LED due to spikes on the data line. I then soldered the wires together and soldered them to the Nano. I made sure the power, ground, and signal wires that connected to the components on the car were long enough to easily remove the hippo base for battery replacement (Figure 35).

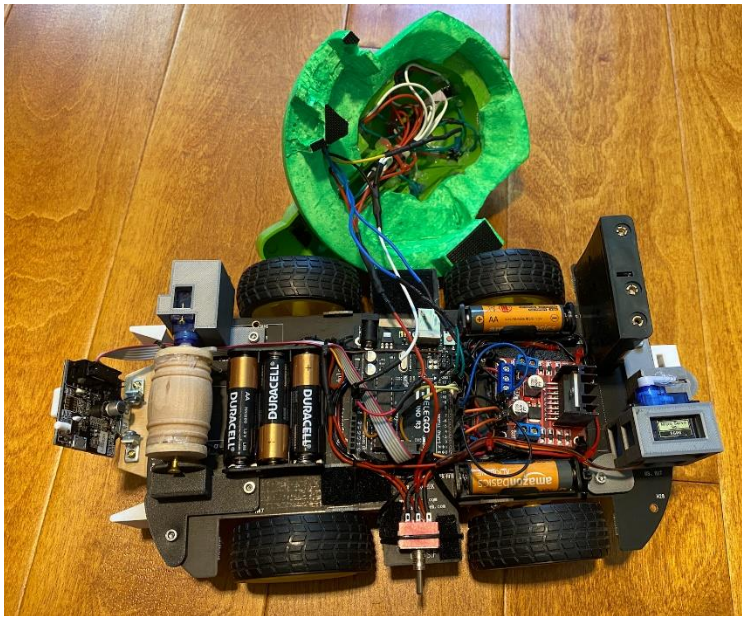


Figure 35: Finished Hippautonomous product with the hippo base removed

I consulted with my group on the best way to set up the Uno to send signals to the Nano. Mohammed and James suggested having the Uno set the output pins high sequentially, and the Nano would continuously read the output pins. We met up a few days before the final demonstration to determine different sequences for LEDs depending on hippo movement and the stage of the game. We then came up with some pseudo code for the LED sequencing. I was able to program the Nano using our pseudo code and to switch LED colors and sequence loops based on the HIGH/LOW status of the digital pins. I wrote simple code for the UNO and Nano to sequence through the different loops. Afterwards, our group met together to try to integrate the code with the main hippo movement program. A couple lessons learned for me include discovering that digital pin 13 is connected to the onboard LED on the Nano and is not a reliable output pin to use and the digital output pins do not stay high for a long period of time after they are set to high. One solution that we did not have time to try is to include the output high command inside more than one loop within the code, so that the Uno continuously writes high values to the respective pins during each of the sequences. Another solution that our group came up with is to add another signal wire between the Uno and the Nano, to have three in total. When the Nano sees Output A turn high, then do sequence A until the Nano reads a change of state from a different output pin, in which case it would trigger a different LED sequence loop.

Power System:

A few challenges arose when designing the power system. Since the car is mobile, it cannot be plugged into a power source that is capable of providing more than enough current for the entire system. Size, weight, and weight distribution of the mobile power banks had to be considered when mounting to the chassis. The power system has to have enough capacity to operate all the electronic components with no issues.

Most of the wiring on the car is 24 gauge. The wiring connected directly to the motors, battery packs, and servos were pre-connected to each of the items. The maximum current travelling through any one wire was measured approximately 1A in short bursts, drawn from the front servo while lifting the front gate. The continuous power calculations based on datasheets of the individual components are in Table 3.

Table 3: Power Requirements/ Calculations.

Component	Input Voltage	Applied Voltage	Output Voltage	Current Draw (active)	Weight
Pixy2 [7]	5V; 6V-10V (unregulated input)	5V	N/A	140mA	G.W: 20g; N.W: 10g
Uno R3 [6]	5V, 7~12V (recommended)	5V	N/A	150mA	25g
Servo [7]	4.8-6V	6V	N/A	220 +/- 50 mA	9g
Motor [8]	3V - 6V	5.5V	N/A	150mA (3Vdc), 200mA (6Vdc) (no load); 500mA max (3V) ~700mA (6Vdc)	30.6g
L298N DC Motor Driver [3][4]	3.2Vdc - 40Vdc	7.5V	2V less than input voltage	36mA	25g
Driver 5V output pin [4]	N/A	N/A	5V	0.5A output	N/A
Arduino Nano 33 IoT [10]	4.5-21	6V	N/A	21mA	5g
RGB LEDs [11]	5V	4.5V		420mA	<1g total

Based on the power calculations, I decided to use AA batteries for the power source. I did some research into using 9V batteries because they are lighter, and AA batteries have a much larger capacity than 9V batteries. Looking back, after having to add the additional 4 batteries after assembly, I should have done more research into high-capacity rechargeable battery banks.

Originally, I designed the system so that the Uno, Pixy, and motor driver were on the same power source. Upon testing, James found that the car did not reliably respond to the change in input signals from the Uno, but when he connected the Uno and Pixy to a separate battery source, then the car movement responded as expected. Considering the results of the testing, I redesigned the power system, such that the motors had their own 5-battery battery pack, and the servos would share a 4-battery pack with Uno and Pixy on a separate circuit. When testing the new design of the front gate, I found that the Uno would reset every time the front servo would begin to raise the front gate. I then realized that the solution was going to require three separate circuits: the motor circuit, the servo circuit, and the Uno/Pixy2 circuit. The Nano and LEDs were connected to the same circuit with the Uno/Pixy2 (Figure 36).

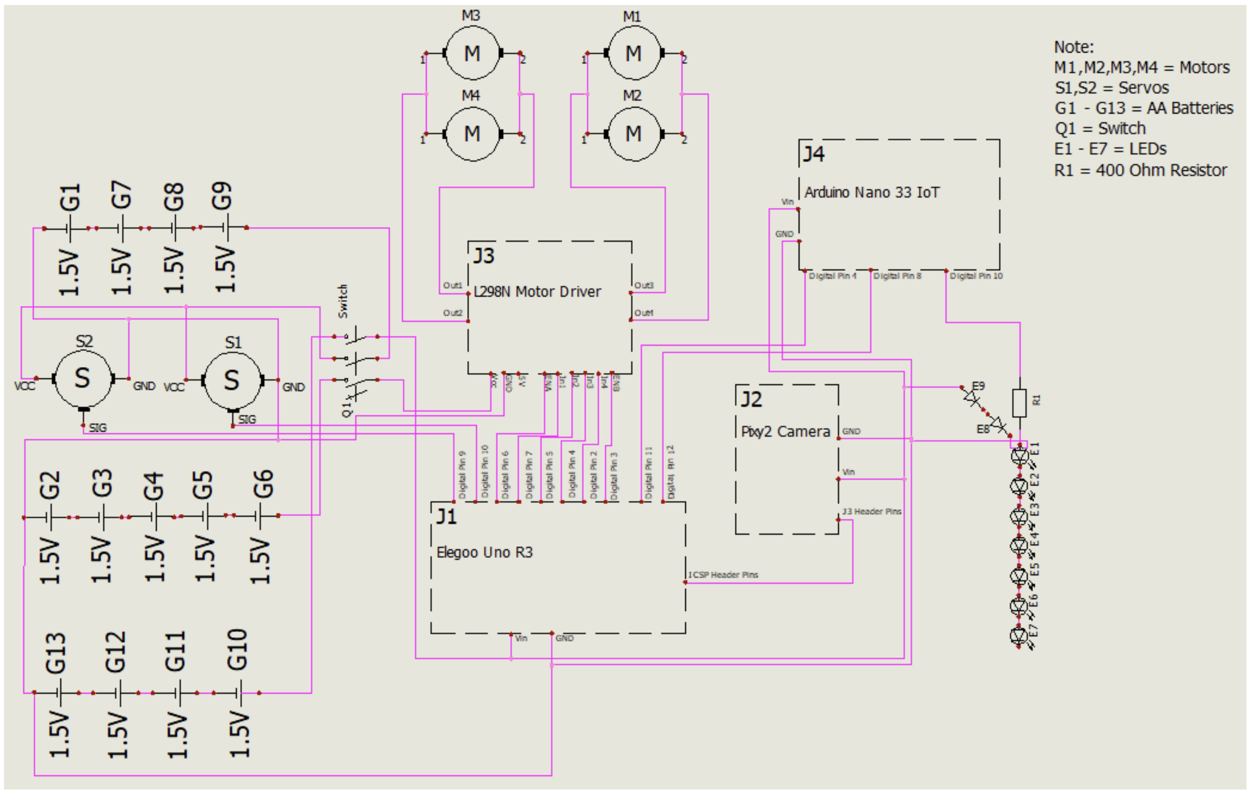


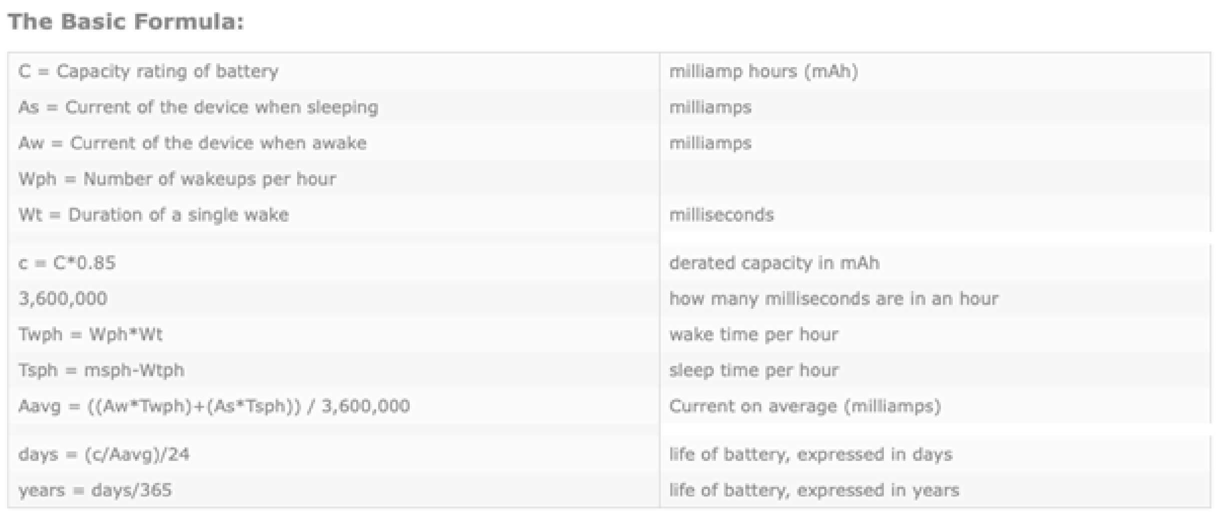
Figure 36: Wiring Diagram.

In the past, I learned that it is best to design a system capable of 130% of the current capacity needed by the system. Through the hippo car experience, I learned that online datasheets can only tell so much, and that testing is needed in addition to calculations to completely understand the power requirements.
The AA battery capacity was calculated based on the data in Table 4.

Table 4: Capacity and Duration of Battery Banks.

	Total Voltage	**Capacity**	**Weight (g)**	**Project Load (mA)**	**Duration**
AA batteries (motor circuit)	7.5V	2500mAh	113.4	1026mA	16.5 hours
AA batteries (servo circuit)	6V	2500mAh	90.8	270mA	904.5 hours
AA batteries (Microcontroller / LED circuit)	6V	2500mAh	90.8	731mA	23.2 hours

In the final weeks of the project, I searched online for a small, lightweight 3 pole, single throw switch with a current rating of at least 1A. The best I could find was a 3 pole, double throw micro switch, which is what I ended up using. The power wires from each of the battery banks are soldered across the switch terminals. The operator is instructed to turn off the switch between games in order to extend the life of the batteries. One advantage to having the batteries on three separate circuits is that they do not all need to be replaced at the same time. Some battery banks will last longer than others, as can be seen in the calculations in Table 5.



The Basic Formula:

C = Capacity rating of battery	milliamp hours (mAh)
As = Current of the device when sleeping	milliamps
Aw = Current of the device when awake	milliamps
Wph = Number of wakeups per hour	
Wt = Duration of a single wake	milliseconds
c = C*0.85	derated capacity in mAh
3,600,000	how many milliseconds are in an hour
Twph = Wph*Wt	wake time per hour
Tsph = msph-Wtph	sleep time per hour
Aavg = ((Aw*Twph)+(As*Tsph)) / 3,600,000	Current on average (milliamps)
days = (c/Aavg)/24	life of battery, expressed in days
years = days/365	life of battery, expressed in years

Figure 37: Parameters and equations for battery life calculation [11]

Table 5: Battery Life Calculations using equations Figure in Figure 37.

<u>5 Battery Bank:</u>	<u>4 Battery Bank for Servos:</u>	<u>4 Battery Bank for Uno,Pixy, Nano, LEDs</u>
Parameters: Capacity AA battery bank (C): 2500 mAh Current sleeping (As): 0.1mA (switch isolates the battery bank from the system) Current awake (Aw): 1026mA Number of wakeups per hour (Wph): 5 (five 90-second games per hour) Duration of a single wake (Wt): 90,000 ms (90 sec)	Parameters: Capacity AA battery bank (C): 2500 mAh Current sleeping (As): 0.1mA (switch isolates the battery bank from the system) Current awake (Aw): 270mA Number of wakeups per hour (Wph): 30 Duration of a single wake (Wt): 1000 ms (1 sec)	Parameters: Capacity AA battery bank (C): 2500 mAh Current sleeping (As): 0.1mA (switch isolates the battery bank from the system) Current awake (Aw): 731mA Number of wakeups per hour (Wph): 5 Duration of a single wake (Wt): 90,000 ms (90 sec)
Calculation:	Calculation: c = C*0.85 = 2500 * 0.85 = 2125 mAh	Calculation:

c = C*0.85 = 2500 * 0.85 = 2125 mAh
 	
Twph = 5 * 90,000 = 450,000 ms (active)
 	
Tsph = 3,600,000 - 90,000 = 3,510,000 ms (non-active)
 	
Aavg = ((1026 * 450,000) + (0.1 * 3,510,000)) / 3,600,000 = ~128.35mA (average current per hour)
 	
Hours = (c / Aavg) = (2125 / 129) = 16.5 hours
	Twph = 30 * 1000 = 30,000 ms (active)
	
	Tsph = 3,600,000 – 30,000 = 3,570,000 ms (non-active)
	
	Aavg = ((270 *30,000) + (0.1 * 3,570,000)) / 3,600,000 = ~2.349mA (average current per hour)
	
	Hours = (c / Aavg) = (2125 / 2.349) = 904.5 hours
		c = C*0.85 = 2500 * 0.85 = 2125 mAh
		
		Twph = 5 * 90,000 = 450,000 ms (active)
		
		Tsph = 3,600,000 - 90,000 = 3,510,000 ms (non-active)
		
		Aavg = ((731 * 450,000) + (0.1 * 3,510,000)) / 3,600,000 = ~91.48mA (average current per hour)
		
		Hours = (c / Aavg) = (2125 / 91.48) = 23.2 hours

D. Analysis of the project

As a whole our project was an overall success, both in the product we created and how we met our expectations. Our final product was a derivation from our first week of ideas, but we created a system that properly identified, captured, and released balls. In addition, our expectations for a budget and timeline matched up almost dead on with the final expenses and the time it took to do each part. There was only one noticeable difference, where the creation and assembly of the collection mechanism took a week longer than expected. However, this was not an issue as we were able to continue working on other aspects during that time.

In terms of the final presentation demonstration went perfectly, identifying every ball presented to it. We spent a minute longer than expected getting the play area set up out of an abundance of caution in setting up the tape underneath the play area. In expectation to this delay the overall presentation went even better than hoped, with no stuttering or misidentification.

E. Guidance

Throughout this project we encountered a variety of setbacks, change in ideas, and development issues and challenges. Looking back with the knowledge of hindsight we identified a few possible key points that would help a redesign team. First and foremost, would be the use of rechargeable batteries, be that AA or some sort of LiPo system, to take the place of the current power system. AA batteries do not have the best lifetime on them, as seen in a variety of different consumer products that use AA batteries. In addition, it would be best to use a better camera, as the resolution of the pixy camera is sketchy at best, terrible at worst. This would mean looking for a better object identification camera or implementing a new system to handle the object identification. Finally, it would be beneficial to include a sensor of some sort, such as a contact sensor, line identification sensor, or some other sensor that can better/more accurately handle the task of staying inside the play area and returning home.

F. Bill of materials (BOM)

Table 6: Bill of materials (BOM).

Part Number	Item	Qty	Unit of Measure
1	Pixy2 Camera (incl. mounting hardware and cable)	1	ea
2	ELEGOO UNO R3 Microcontroller	1	ea
3	Servo (SG90 Digital)	2	ea
4	Hippo from Hungry Hippos game	1	ea
5	Arduino Nano 33 IOT	1	ea
6	1/8" Hardboard Tempered Panel (4ft x 8ft)	2	ea
7	AA Batteries	13	ea
8	Jumpers	3	ea
9	4WD Robot Chassis kit (with mounting screws)	1	ea
10	L298N Motor Driver	1	ea
11	Wood Balls (colored with pink highlighter)	5	ea
12	Wood Barrel	1	ea
13	AA Battery Holder (1 battery)	2	ea
14	AA Battery Holder (2 batteries)	2	ea
15	AA Battery Holder (3 batteries)	1	ea
16	AA Battery Holder (4 batteries)	1	ea
17	Switch (3PDT)	1	ea
18	Diodes (1N4001)	2	ea
19	RGB LEDs (WS2812B)	7	ea
20	Fishing Line (20 lb)	7	inches
21	Pen Springs (from Pentel EnerGel pens)	3	ea
22	Foam for Hippo Base	1	ea
23	Back Gate Servo Mount	1	ea
24	Front Gate Servo Mount	1	ea
25	Barrel Support Mount	1	ea
26	Undercarriage (ball collection)	1	ea
27	Front Gate	1	ea
28	Back Gate	1	ea
29	M2 Machine Screws (20mm length)	7	ea
30	M2 Machine Screws (12mm length)	2	ea

31	#2 Wood Screws (3/8in length)	6	ea
32	#4 Wood Screws (3/4in length)	1	ea
33	#2 Wood Screws (1/2in length)	3	ea
34	3/4" Wood Balls (colored with pink highlighter)	8	ea
35	Paint Stick	1	ea
36	Duct Tape	3.5	ft
37	4" Cable Zip Ties	15	ea
38	Velcro - 1.5"x1"	1	ea
39	Wire (24 gauge)	As Required	-
40	Hot glue sticks - 5/16"x4"	4	ea
41	Super glue	1	ea
42	Glow in the Dark Acrylic Paint	1	ea
43	Solder 1mm diameter roll	1	ea
44	Heat Shrink Tubing - 3mm width	15	inches
45	Heat Shrink Tubing - 4mm width	4	inches
46	Heat Shrink Tubing - 6mm	1	inches
47	Jumper Wire (28 gauge) - 4.5"	8	ea

G. Appendix

Link to Code Repository

https://github.com/Yamez3/ERAU-Hippautonomous

Link to Final Requirements

https://tinyurl.com/425spnxk

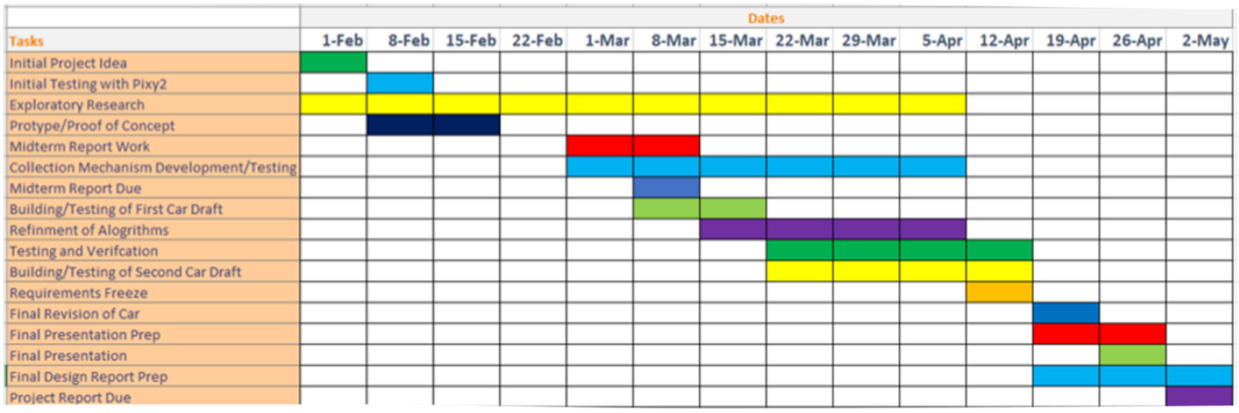
Original Budget

Item	Price	Actual Price (ea)	Quantity	Subtotal
Pixy2 Camera for Raspberry Pi	59.90	59.90	1	59.90
Raspberry Pi / Arduino	35.00	35.00	1	35.00
Servo Four Pack	8.99	8.99	1	8.99
Hungry Hippos Game	19.82	19.82	1	19.82
7 Segment Display	5.50	5.50	1	5.50
Arduino Nano 33 IOT	19.00	15.00	1	19.00
Sensor setup for detecting hippo ate marble	10.00	10.00	2	20.00
Play Arena/Camera Structure	35.00	35.00	1	35.00
Misc wire, batteries, electrical accessories, starter kits	30.00	30.00	1	30.00
RC Car (main)	15.00	21.24	1	15.00
Drivers for car	8.89	8.89	1	8.89
Taxes and shipping	34.98	40.38	1	34.98
Contingency	50.00	50.00	1	50.00
Total				**342.08**

Final Budget

Item	Price	Quantity	Total Cost
Pixy2 Camera	59.90	1	59.90
ELEGOO UNO R3 Microcontroller	13.98	1	13.98
Servos	4.50	2	8.99
Hungry Hippos Game	19.82	1	19.82
Arduino Nano 33 IOT	17.45	1	17.45
Play Area Boards	22.22	1	22.22
Misc wire, batteries, electrical accessories, hardware	20.60	-	20.60
4WD Robot Chassis kit	21.24	1	21.24
Drivers for car	8.89	1	8.89
Wood Balls Pack	2.49	1	2.49
Wood Barrel Pack	2.49	1	2.49
PLA Filament	24.99	1	24.99
Battery Holder	5.88	1	5.88
Switch	2.25	1	2.25
Diodes	0.61	2	1.22
LEDs	0.13	7	0.92
Fishing Line	0.25	1	0.25
Pen Springs	0.03	3	0.10
Foam for Hippo Base	3.00	1	3.00
Contingency	19.90	-	19.90
Shipping/Taxes	12.89	-	12.89
Total			**269.47**

Schedule



	Dates													
Tasks	1-Feb	8-Feb	15-Feb	22-Feb	1-Mar	8-Mar	15-Mar	22-Mar	29-Mar	5-Apr	12-Apr	19-Apr	26-Apr	2-May
Initial Project Idea	🟩													
Initial Testing with Pixy2		🟦												
Exploratory Research	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨				
Protype/Proof of Concept		⬛	⬛											
Midterm Report Work					🟥	🟥								
Collection Mechanism Development/Testing					🟦	🟦	🟦	🟦	🟦	🟦				
Midterm Report Due						🟦								
Building/Testing of First Car Draft						🟩	🟩							
Refinment of Alogrithms							🟪	🟪	🟪	🟪				
Testing and Verifcation								🟩	🟩	🟩	🟩			
Building/Testing of Second Car Draft								🟨	🟨	🟨				
Requirements Freeze											🟧			
Final Revision of Car												🟦		
Final Presentation Prep												🟥	🟥	
Final Presentation													🟩	
Final Design Report Prep												🟦	🟦	🟦
Project Report Due														🟪

H. References

[1] https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:ccc_api

[2] xiaorgeek.com, ‘Smart Robot Car Chassis Kit - 4WD with DC motor’, 2018. [Online]. Available: http://www.xiaorgeek.com/store/robot-accessories/smart-robot-car-chassis-kit-4wd-with-dc-motor.html. [Accessed: 05- Mar- 2021]

[3] lastminuteengineers.com, ‘Interface L298N DC Motor Driver Module with Arduino’, 2021. [Online]. Available: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/ [Accessed: 05- Mar- 2021]

[4] handsontec.com, ‘L298N Dual H-Bridge Motor Driver’. [Online]. Available: http://www.handsontec.com/dataspecs/L298N%20Motor%20Driver.pdf [Accessed: 06- Mar- 2021]

[5] media.digikey.com, ‘Pixy 2 CMUcam5 Smart Vision Sensor’. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Seeed%20Technology/102991074_Web.pdf [Accessed: 07- Mar- 2021]

[6] epow0.org, ‘Elegoo UNO R3’. [Online]. Available: https://epow0.org/~amki/car_kit/Datasheet/ELEGOO%20UNO%20R3%20Board.pdf [Accessed: 07- Mar- 2021]

[7] opencircuit.shop, ‘TOWERPRO SG90 9G MICRO SERVO MOTOR - 180°’. [Online]. Available: https://opencircuit.shop/Product/TowerPro-SG90-9G-micro-servo-motor-180 [Accessed: 06- Mar- 2021]

[8] adafruit.com, ‘DC Gearbox Motor - "TT Motor" - 200RPM - 3 to 6VDC’. [Online]. Available: https://www.adafruit.com/product/3777#technical-details [Accessed: 06- Mar- 2021]

[9] oregonembedded.com, ‘Battery Life Calculator’, 2019. [Online]. Available: https://oregonembedded.com/batterycalc.htm [Accessed: 07- Mar- 2021]

[10] github.com ‘Arduino Nano 33 IoT – Ultimate Guide’ [Online]. Available: https://github.com/ostaquet/Arduino-Nano-33-IoT-Ultimate-Guide [Accessed: 01- May- 2021]

[11] adafruit.com, ‘Basic Connections’. [Online]. Available: https://learn.adafruit.com/adafruit-neopixel-uberguide/basic-connections [Accessed: 01- May- 2021]