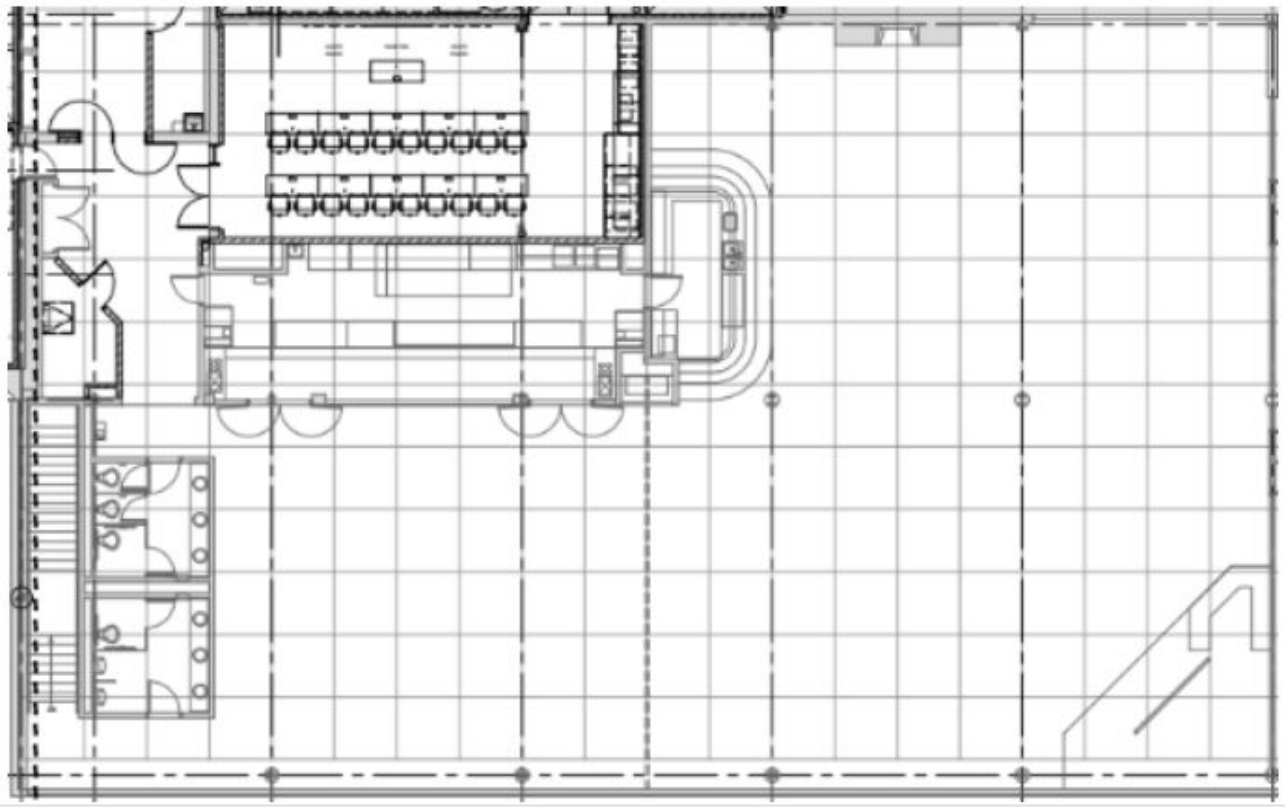
ENR 281 Design Project Report (Spring 2024)

Leo Devick, Caleb Raes, Morgan Frisch, Elliot Kooiker, Cayden Floyd, Dr. Jim Carrico

Abstract

This report documents Team Carrico's design work for the Marauders Cart project assigned in the sophomore design lab ENR 281. Each team was tasked to design a static and dynamic structure for the race course, as well as their own remote controlled car to race.

We share the progress achieved during the first five weeks of the Design Lab project, including details about the first car prototype, the first segment of the dynamic obstacle, and the initial progress on the static obstacle.



Contents

Executive Summary

This report presents the progress made during the first five weeks of the ENR 281 Design Lab project. Exceptional progress was made in all areas of the project. The first car prototype was created, which features a fully remote controlled car and the controller. The design for the car's belt system was printed and assembled, as well as a prototype to test the car's motors and power requirements. The first segment of the dynamic obstacle was created, and the dynamic aspect was revised after testing the solenoids. The static obstacle's first half was assembled, and the maximum ramp angles calculated. Research from the previous semester, ENR 280, is also presented, and details about our decisions for the project, including weighted decisions matrixes. We include relevant figures and code snippets in the Appendix.

1. Design Problem and Objectives

The problem presented to us in ENR 280/1 was to design and construct a remote-controlled car and a full race course to be located in Chick's Place, LVUC. A map of the space the race course had to occupy is shown in Figure 1. The goal of this project was for each team was to design the following: 1) A remote control car, 2) a dynamic structure, and 3) a static structure.

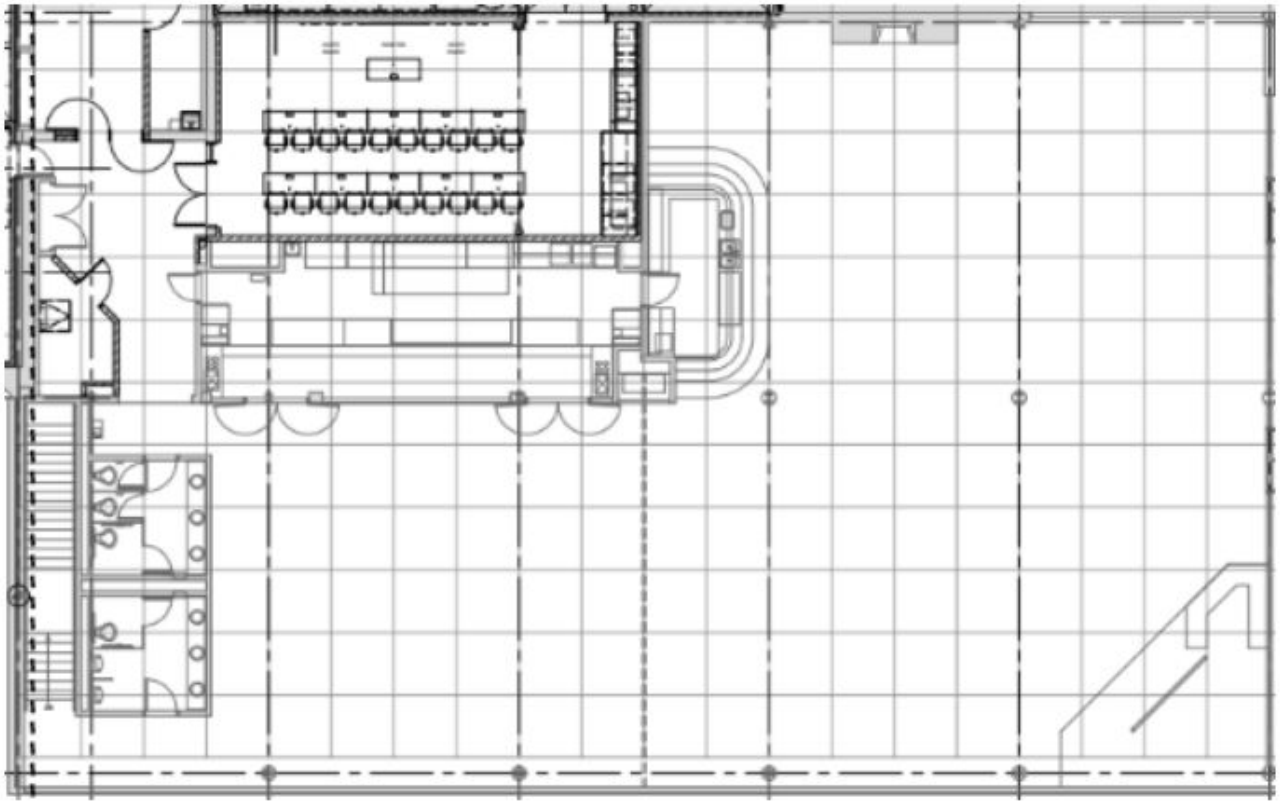


Figure 1. Figure 1. Chick's Place Space for the Marauder's Cart course

1.1 Design Specifications

Our team manager gave us requirements for our remote controlled car and our obstacles. These requirements are presented in table 1, and are given details in the following sections.

Table 1. Design Specifications

Need	Design	Specifications
Car	Arduino remote controlled car	Flippable Design No weapons/hacking/signal jamming Big Wheels Differential Steering
Dynamic Structure	Trapdoor Bridge	Span: 6 ft Width Capacity: 2 Cars Arduino Controlled Doors Activation: Weight
Static Structure	Undulating Road	Road increases in amplitude over distance

1.2 Project Objectives

This project is a way for engineering students to learn about the design process, work on a team, and simulate being a member of an engineering firm. Our final deliverable is a remote controlled car and two obstacles, but our goal in this project is to have the fastest and most maneuverable car, and the most difficult to navigate obstacles.

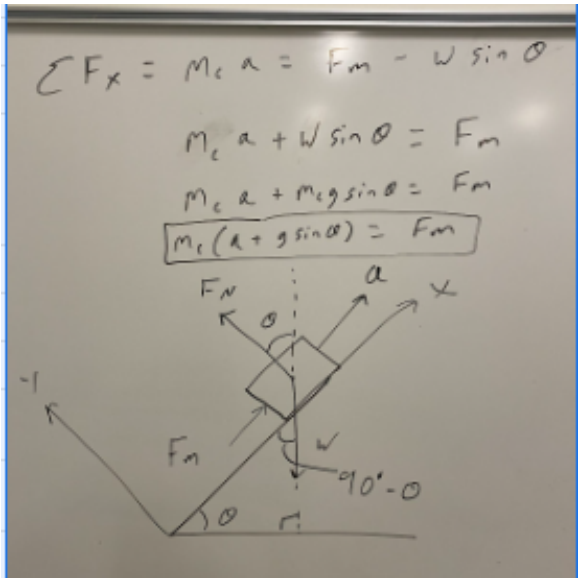


Figure 2. Drawing of calculations for Motor Specifications

2. Research

Research is an important part of the design process, especially during the early iterations. Topics we've learned about so far have been motors, batteries and integrated circuits. We also conducted research using the motors we ordered to determine the power consumption of our car and the maximum climbing angle. We experimented with our solenoids, and used those results to revise our design for the dynamic obstacle.

2.1 Motor Calculations

With the help of our team manager, we created a free body diagram and calculated the amount of torque and power we would need to drive our car up a 45 degree incline. In figure 2 we present our diagram, and table 1 contains the information which takes the ideal car and wheel parameters and calculates motor requirements.

2.2 Batteries

We plan to use Lithium Ion batteries to power our car. Lithium Ion batteries have a high energy density, which means they can store more energy for less weight, and are rechargeable which can save us money. Most Lithium Ion batteries have a charge and discharge circuit built in, which makes it easier to use the battery. They are also relatively inexpensive compared to other batteries, and recharge ability saves even more money.

One concept we investigated was battery life. Most batteries have their power ratings in milliamp hours. An expression for milliamp hours (mAh) is below.

$$\text{mAh} = \text{mA} * \text{h} \tag{1}$$

This relationship can be written to solve for time.

$$h = \frac{mAh}{mA} \tag{2}$$

Table 1. Car and Motor Specifications

Car Mass	**3.628**	kg		
Car Weight	35.586	N	8.000	lbs
Car Wheel Radius	0.075	m	2.953	in
Car wheel size	0.150	m	5.906	in
Top Speed	**0.250**	m/s	0.820	ft/sec
Wheel Circumfence	0.471	m	18.553	in
Max angular velocity of wheels = (Top Speed) / (Car Wheel Radius)	3.333	rad/sec	31.831	rot/min (RPM)
Angle	45.000	Degrees		
Acceleration due to gravity (g)	9.810	m/sˆ2		
Accel Desired up hill (a)	0.500	m/sˆ2	1.640	ftsecˆ2
Force needed for going up hill (Fm) Refer to the diagram	26.977	N	6.065	lbs
Total motor torque needed for going up hill	2.023	N-m	17.907	lbf-in
Torque per motor asuming 2 motors	1.012	N-m	8.954	lbf-in
Power(total)	6.744	Watts	0.009	hp
Power(per motor assuming 2 motors)	3.372	Watts	0.005	hp

Since we are using 12 Volt Motors, drawing 4 Watts, we need to plan for 12 Volts and $\frac{1}{3}$ Amperes to be supplied by our power source.

2.3 Integrated Circuits

We researched a few different ICs to use in our car and remote control design. These components include the L293D Two Channel Motor Controller, the 7805 Power Regulator, and the HC12 Arduino Wireless Serial Board.

2.4 Motor Experimentation

To test the motors, we 3D printed brackets to house the motors, and an interface so that the motor could be attached directly to the wheels. We mounted the motors to a piece of scrap wood, and attached a caster to the back of the body to allow the prototype to move freely. All the electronics were mounted to the body of the car. Power was supplied from a bench top power supply, and ran to the car using long wires. A photo of the first prototype is shown in figure 3.

IC	Use
7805	Accepts an input of 7-24 Volts, and steps it down to a regulated 5V output. We plan to use this for our car and controller Arduino so that we can use a larger and more convenient rechargeable battery but still power our 5V microcontroller
L293D	Accepts two inputs, one to power the chip itself and another to direct to the motors. This can allow us to power the chip from the Arduino with 5V but power the motors directly from our source, which will let us have more current through the motors than usually allowed for by the Arduino
HC12	This is a Transceiver chip which allows for virtualizing serial communication between two Arduinos. This allows the car to be controlled remotely.

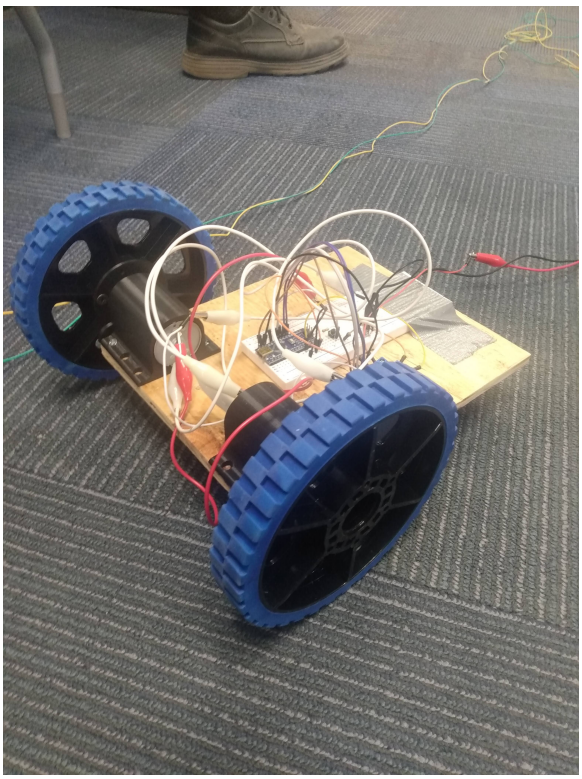


Figure 3. The first car prototype

Using the power supply was advantageous for two reasons: first, it allowed us to alter the voltage and current limit running through the circuit, to simulate our battery requirements. This gave us a sense of what we would need for a battery, and specifications to search for in our research. Second, the power supply gave us live feedback for current draw from the motors. We might need to implement a current limiting circuit in our design to avoid damaging the micro controllers or over-discharging the battery.

We also tested the maximum angle that the car could climb. For testing, we used the power supply as a theoretical 12V battery, with a maximum current draw of 1.5A. The car was able to climb a 30 degree angle successfully using a front wheel drive configuration, and 36 degrees with rear wheel drive.

Figure 4. Caleb and Freddy testing the car's maximum angle

The motor test was very informative, and will help us determine what kinds of batteries we need to purchase to give our car adequate power to do well in the race.

2.5 Solenoid Research

The solenoid we ordered was also tested, and key information was discovered. A note about the solenoids: The component we received was not the component we ordered. Our team talked to Rodrigo and showed him the correct part, and placed a new order. We performed testing on the incorrect part, in order to understand how these components work in general.

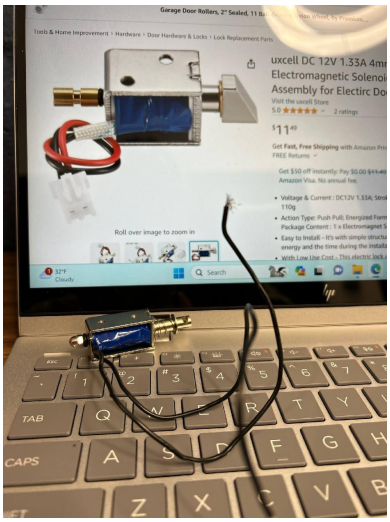


Figure 5. Photo comparing the solenoid we received with our amazon link

Concerning the solenoid's electrical draw, there is a point when the current is high enough for the metal bar to retract by itself. For this part, it retracted at 700mA. This means we will need to use transistors or another electrical switch to control the solenoid with the Arduino, as the Arduino can only supply a maximum of 40mA of current. We still have 2 L293D motor driver chips, which could be used to control the solenoids, and save money and time on components.

The solenoids also get very warm when kept in a steady retracted state. We did not keep they retracted for more than a minute for fear of damaging the part, but even that time was enough for them to get

uncomfortable to touch. We are considering options for cooling, like heat sinks or repurposing computer fans to cool the component.

The team also considered the use of the pressure sensor in the final design during the test. The test solenoids were not strong enough to retract while a load of more than 1lb was applied to them, which leads us to believe that triggering the solenoid on the door while a car is driving over it won't cause the door to properly open.

A solution to the problem of the overheating solenoids, the solenoids not being strong enough to retract under pressure, and making good use of our resources, like the pressure sensor we purchased already, is to put the sensor at the front of our obstacle and open the doors which need to be opened only when a car is there. The pressure sensor can be mounted underneath a thin piece of wood, and calibrated to ignore the weight of the wood. When a car drives across the sensor, the Arduino which is monitoring the sensor will be activated, and each door that has been programmed to open will open. Most likely, the circuit will be on a timer, so after a few minutes the solenoids will retract again, and not be reactivated until another car drives through.

2.6 Spring Loaded Hinges

While researching for hinges, we had to take into account that we needed it to be spring loaded so that it was able to shoot back up and reload into the upright position after dropping a car. For this to work correctly, we need a spring that is string enough to be able to lift the weight of the door. After researching, the following formula was given to be able to calculate the spring constant.

$$k=(d(4)*E)/(64*DN) \tag{3}$$

In this equation, d is the diameter of the spring material, E is the elastic modulus of the spring material, D is the diameter of the coil, and N is the number of turns that the spring has. Our spring loaded hinges that we ordered is made out of 304 stainless steel, and the elastic modulus is $193,000 N/mm^2$. After using this equation, the spring constant was calculated to be 25.73 N-mm/rad.

2.7 ESP32 Microcontrollers

After a lot of trial and error with the HC12 transceiver, we decided to look into an alternate way to create wireless communication between our car and controller. The ESP32 microcontroller is an inexpensive development board with a rich feature set. Integrated into the board itself is a WiFi and Bluetooth module, allowing for information to be sent easily and reliably via the board's proprietary ESP-NOW data transfer protocol. These boards are powered by 5V, but run on 3.3V logic, which makes them consume less energy, but they require a 3.3V to 5V step circuit to correctly interface with our motor controllers.

They have a great resolution for analog read at 12 bits, allowing us to distinguish a value between 0 and 4095 when using our joysticks. They are programmed using the Arduino IDE, and are compatible with most of the Arduino API, making changes to our existing code very slim.

2.8 3.3V to 5V Level Shifter

Since the ESP 32 Microcontroller operates on 3.3V pin logic, we needed a way to make it use 5V logic to interface correcting with our motor controllers.

Figure 6. ESP32 Development Board

3. Design Conceptualization, Initial Ideas, Process and Decisions

3.1 Design Concepts and Initial Ideas

3.1.1 Car

We have two main ideas to consider for our car's chassis and wheel layout. Design 1 has a rugged chassis with good space for batteries and control systems. Design 2 has asymmetric wheels, but the design makes for smoother flips.

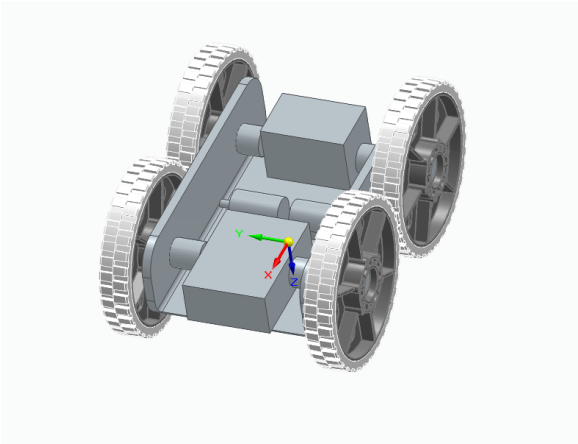


Figure 7. Car Design 1

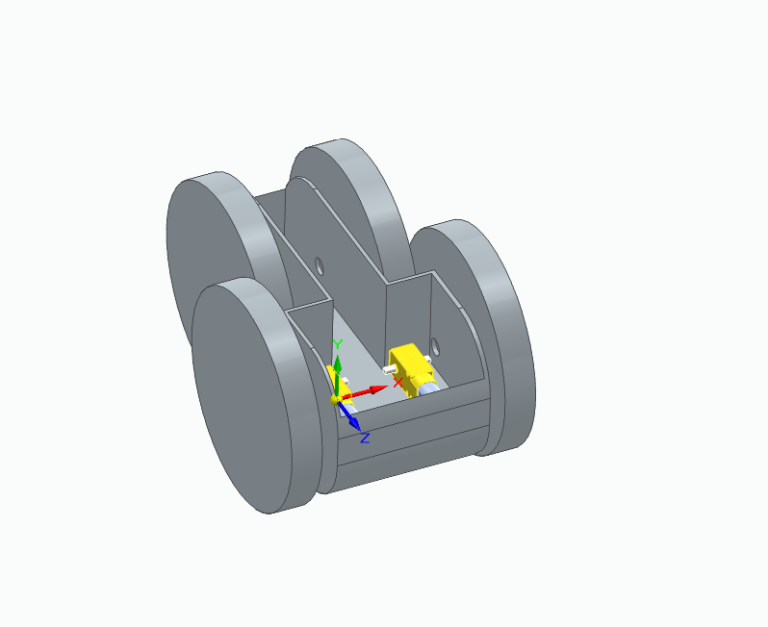


Figure 8. Car Design 2

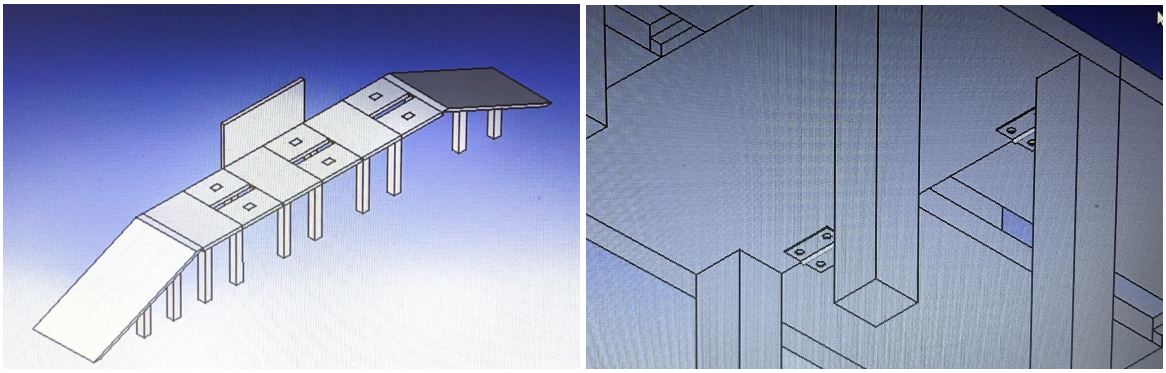


Figure 9. First Bridge Cad

3.1.2 Dynamic Structure

We created CAD models for the trapdoor bridge. The bridge has three sets of two trapdoors, and a staging area in front of every pair. The bridge is a puzzle; next to the bridge will be a sign or a computer monitor which gives drivers a puzzle to solve. The doors will be marked with an answer, and if the driver gives the right answer and drives across the corresponding bridge, they will proceed across safely.

3.1.3 Static Structure

We started with sketching the undulating road, then creating a mockup prototype out of PVC pipes and wood.

3.2 Process and Decisions

Using the Decision Matrix tool, we weighted the benefits and hindrances of each design and used that to inform our decision making.

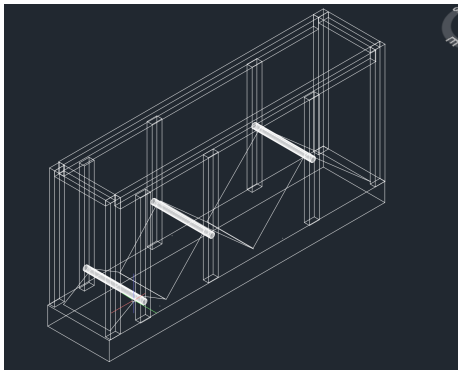


Figure 10. Undulating Road Sketch

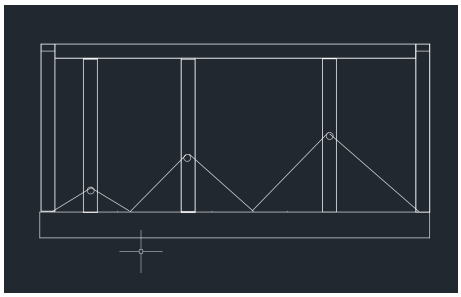


Figure 11. Undulating Road Sketch

3.2.1 Car Designs

Each design was created with our Team Managers specifications in mind, but solved the problem in different ways (See figure 54 for the completed car matrix).

The Design Matrix for deciding the car designs used the criteria of estimated cost, simplicity, ground clearance, robustness, tires, motors, body, and flexibility.

Our two-car designs reflected our team manager's desire for a car that had big wheels and would be able to still drive, even if it was flipped upside down.

Design one had one set of wheels inboard the other, allowing larger wheels than the 12-inch car length would allow. The wheel size was a large 8 inches, which would allow for greater ground clearance , but at the cost of space and center of gravity. The wheels would be driven by one motor on each side using a belt system since the front of the car would be too narrow to fit motors to each wheel. The wheels would be a 3D-printed core with rubber tread glued onto it.

Design two used 6-inch wheels aligned colinearly. The wheels would have their diameter reduced by 1/16th on the lathe to allow them to be as large as possible while still fitting the parameters. The tires would be driven using a similar belt drive-two motor system, with the motors placed in the middle of the car and the belts transferring the power to the wheels. A central part of the design was its ability to use a metal C channel extrusion as its main body, giving it greater strength.

The estimated cost was based on the overall cost of components and the needed hardware for the car. Design 1 scored a 6 due to it using more homemade components instead of boughten parts. Design 2 scored a 5 since it used prefabricated tires, which are more expensive than the 3D printed ones.

The simplicity of the design was based on how many moving parts and different systems each car had, and how many things could go wrong. Design 1 scored a 5 since the 3D-printed tires and belt system could fail. Design 2 scored a 7 since its prefabricated tires are more durable.

Ground clearance was decided on by the distance between the bottom of the car and the floor both right sides up or upside down. It also considered the break over, approach, and departure angles of the two designs. Design 1 scored a 7 while Design 2 scored a 6 due to Design 1 having larger tires that gave it more clearance in all directions.

Robustness was the measure of the design's ability to resist damage during the race. Design 1 scored a 6 while Design 2 scored a 7 due to the metal extrusion giving it extra strength.

Flexibility was the rating of how the design of the car and its components could be changed if there were serious flaws or circumstances that would require the design to be quickly adapted. Design 1 scored a 6 due to its specialized shape constraining the size and placement of the motors and control systems. Design 2 scored an 8 due to the ability of its motor and drive system to be changed easily. Its body could be printed as well as made for metal; it could run a four-motor as well as a two-motor system; the batteries and control systems could be easily repositioned.

The next three categories were based on specific components of the cars. The Body category is a measure of the internal space of the chassis. Design 1 scored a 6 due to it having a deeper, although narrower chassis. However, the design also meant that those components would be stored in one end of the car, giving it a lopsided center of gravity. Design 3 also scored a 6 due to it having a shorter, but wider and more rectangular body shape. This also meant that the weight could be spread more evenly.

The motor category was scored on the motors used by the car as well as the ability to use different motors and motor systems. Car 1 scored a 7, begin able through its belt system to use large or small motors. Design 2 scored an 8 since it would be able to use both two large motors as well as four smaller ones.

Tires were scored on the tires of the designs and their ability to grip, be integrated into the design, and their durability. Design 1 scored a 6 since its glued tread could be torn off and the core of the tire could be broken more easily. Design 2 scored an 8 since its tread would not fall off as easily, and it would be more durable.

3.2.2 Trapdoor Bridge Designs

We also created a matrix for the trapdoor bridge (See Appendix: Figure 55).

Material Cost is how much a given design will cost, and how exotic of components it will need to function. Durability is how resistant to breaking the design is. Since the bridge will need to be in operation for the whole race, this is an important criterion. Reliability is how effective the design is for it's cycles of operation. The design must be able to continue to operate according to our team manager's requirements for the entire race. Effectiveness is how good the bridge is at it's job of tripping up the other cars. An ineffective bridge would not trigger when a car is on the incorrect door, or have another tell which lets the driver get across without solving the puzzle. Simplicity is both simplicity of design and execution. We wanted to err on the side of simplicity so that our dynamic structure was easy to maintain, build, and transport.

3.2.3 Undulating Road Design

We created a decision matrix for the undulating road, and weighed it with similar criterions to the trapdoor bridge (See figure 56 for the full matrix).

4. Detailed Design

4.1 Undulating Road (Static Structure)

Our Manager gave us a design for an undulating road, a road with hills and valleys with three distinct sections of amplitude.

4.1.1 Assumptions

We assumed that this structure would be built chiefly out of wood, and have a flexible material to use for the driving surface. We planned for modularity, the ability for it to be assembled and disassembled for transportation.

4.1.2 Functions and Meeting Specifications

Our project manager gave us an additional design of for undulating road: a box with PVC pipe hangers that had chicken wire or sandpaper running from them to create a hill. The hill material is draped across the PVC pipes, which are suspended on either side by a box made of Plexiglas. This will make manufacturing our design a lot simpler and more cost effective while still preserving the original design parameters set by our manager.

We created a CAD model for our static obstacle. Figure 12 shows the details of how the structure works. It is composed of three hills, which are made of wire suspended by a dowel between two sheets of Plexiglas.

4.1.3 Prototypes

We have been working on the base and extension of our static obstacle. It is an undulating road with three changes in elevation. We finished the first archway and calculated the angles for all three ramps. The max angle of climb we would like is 35 degrees. The calculations figured the tallest ramp will have to be 30 inches long to make a 35 degree climb. The descent from the ramp is not a concern, as long as it is 45 degrees or less.

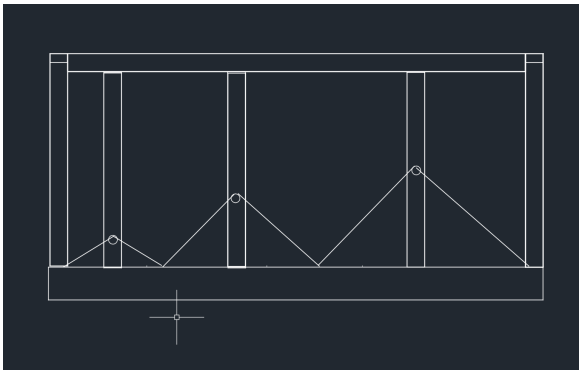


Figure 13. CAD Design of the Static Obstacle

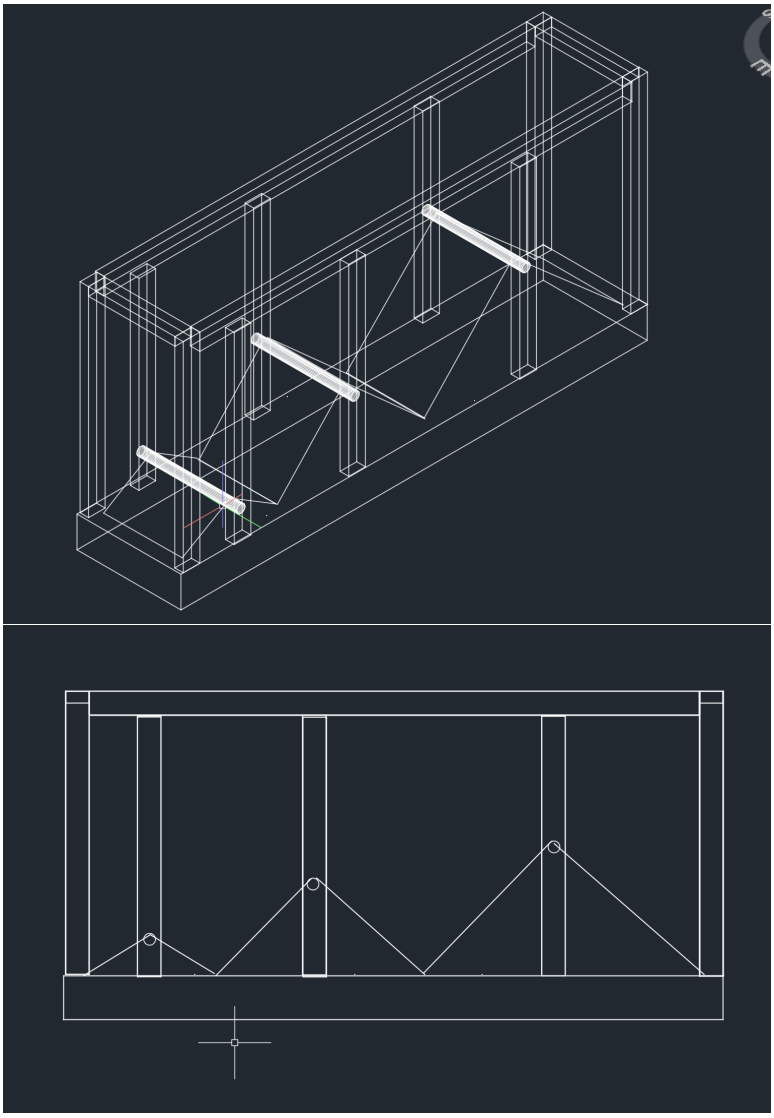


Figure 12. CAD Model for Static Obstacle

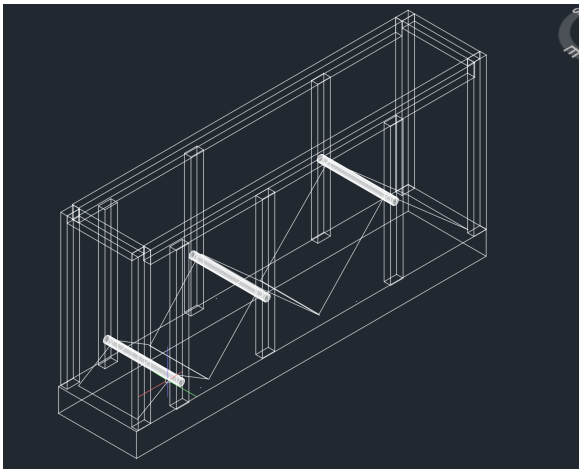


Figure 14. CAD Design of the Static Obstacle

4.1.4 Manufacturing
The structure was constructed out of plywood, 2x4 beams, chicken wire and PVC pipes.

4.1.5 Final Design
The final design performed well in the race. It was a challenge for most cars to get through.

4.2 Trapdoor Bridge (Dynamic Structure)
Our manager gave us the idea for a bridge of trapdoors. Each section on the bridge has two possible doors to drive across, and there is a puzzle associated with each set of doors. Clever drivers can solve the puzzle and get across without resetting, but other drivers can use trial and error to find the correct path across.

4.2.1 Assumptions
For our initial designs, we assume the length of the bridge to be 108 inches, and the width to be 16 inches. The bridge is modular in nature, so there are three pieces that each have a door and staging area, and the two ramps which come off. Because of modularity, the wires connected to the door locking mechanism are easy to connect and disconnect.

The bridge must be remote controlled, so that during race time different doors can be toggled. We used another HC12 to achieve the wireless capabilities and connect to our on-board Arduino. The Arduino takes the input from the controller and makes a door safe or not, depending on the state of the switch on the remote.

4.2.2 Functions and Meeting Specifications
The bridge has a computer monitor attached to it, connected to a Raspberry Pi which controls the monitor. During the race, our team manager can enter commands on a laptop connected to the Pi, and update the clue on the board, as well as which doors are opened and closed.

We trimmed a set of doors from the bridge to make our design more compact, and save money on our budget. Figure 15 showcases the CAD model of the new trapdoor bridge. The flat panel behind the bridge is the stand-in for the monitor to show the clues.

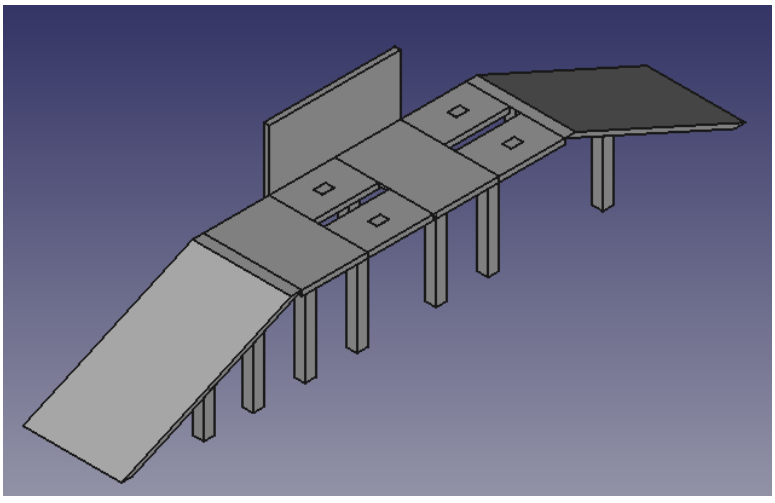


Figure 15. CAD Model for Dynamic Bridge with Two Doors

We also needed a microcontroller component for our structure. The diagram in Figure 16 shows how all the electronic components fit together, and what protocols they use to communicate.

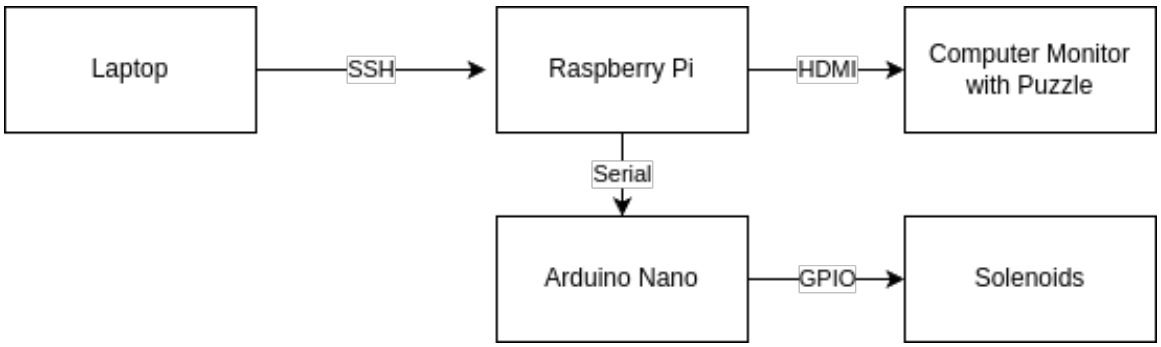


Figure 16. Dynamic Structure Tech Stack

4.2.3 Prototypes

We finished all three platforms for the dynamic structure (Figure 18). After our completion of the first platform, we were pleased with the design and made no modifications. We requested an order for two more packages of spring hinges. As seen below, we need three spring hinges for every trap door, due to the weight of the wood (Figure 17). This brings the total amount of spring hinges to 12. There are four major steps to finish the dynamic obstacle. First, we need to install the remaining spring hinges to the last three doors. This step will be delayed until the spring hinges are delivered. Second, we need to put a strip of wood over the screws from the spring hinges. This will help make the structure more aesthetically pleasing by leveling out the doors and also cover up the screws that are protruding. Next, we will need to assemble ramps for the structure. We need to wait until we have our car assembled so we can be sure that our own car can make it up the ramp. Our design is to make the upward ramp with an incline of 34.06 degrees and the downward ramp with an incline of 50 degrees. The final step needed to complete the body of the dynamic structure is to make beams between platforms and underneath of the doors. This will add structural support and keep even spacing between platforms.

Figure 17. One Bridge Module. Note that Three hinges are better than One

Figure 18. All three completed modules for the Dynamic Structure

4.2.4 Manufacturing

The bridge was built in three modules: the first module featuring the on ramp and first platform, the second module with the first set of doors and a platform, and the third module with the second set of doors and the off ramp. To hold these three modules together, we 3D printed an L bracket, which connects to a wooden slat, which is then bolted between the legs of two modules to provide stability to the modules and prevent separation during the race.

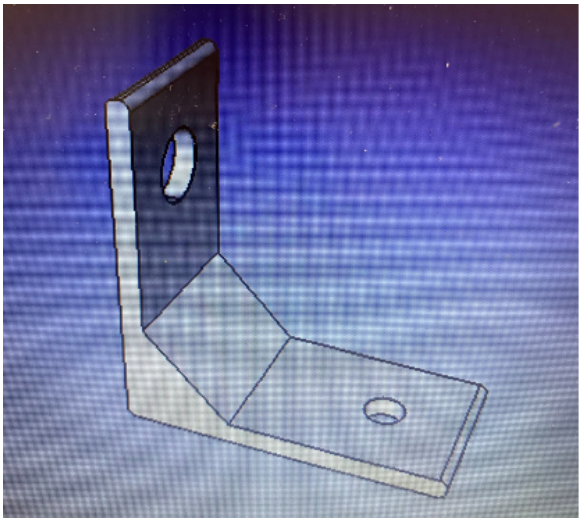


Figure 19. CAD Model of L Bracket to combine bridge modules

Figure 20. L Bracket installed on Bridge Retaining Piece

4.2.5 Final Design

The design performed as expected in the race. The Arduino and Raspberry Pi communication worked like a charm, and we were able to connect to the Pi by SSH and set up the puzzle correctly. We had a monitor adjacent to the structure, connected to a laptop, which displayed the clue for the structure. This was not the original plan, as everything was supposed to run off of the raspberry pi, but we ran out of time and had to improvise. However, heat management of the solenoids ended up being an issue. At the end of the race, one solenoid got too hot and it melted through the bracket that was holding it to the structure (Figure 21). This could have been avoided if we changed the timer on the structure to a shorter time, but we did not realize that the tolerance of PLA was so low.

Figure 21. Melted Solenoid Bracket

4.3 Car

4.3.1 Assumptions

Our design parameters were to have a car which was durable, could still drive if it was flipped, had bigger wheels and steered with a differential steering. We began by making CAD models of a few different design ideas, and brainstormed ways to use the requirements in our design.

4.3.2 Functions and Meeting Specifications

For controlling the car, our controller will have different speed modes the user can toggle between. There will be a high speed mode for quickly maneuvering around the course, and a low speed mode for crossing obstacles and bridges. It will also have a joystick for gross turning, and push buttons for precision turning.

The car body is a durable C channel that will be able to with stand damage and drops, and safely house all the components. Due to other obstacles, the body will also be water proof to ensure the electronics inside do not get damaged.

4.3.3 Prototypes

Figure 22 shows the first prototype of the car design. It has most of the final components, including the wheels, belts, pulleys, and motors. Creating a workable design as early as possible helped to visualize how the components fit together and the challenges that would face because of them. For example, it became apparent that the larger motors and belt system would take up considerable space inside the body of the car, which would potentially cause issues with fitting the battery, electrical components, and transceivers. Also note the machined changes to the tires, which include using the lathe to remove 1/16th of an inch from the radius in order to stay within in the size parameter. We will also be using the mill to remove 0.145 inches of excess plastic from the sides of the rims. The wheel hubs will consist of 3D printed PLA pieces that bolt onto the rim of the tire. This hub will have a core made up of a rod aluminum milled into a

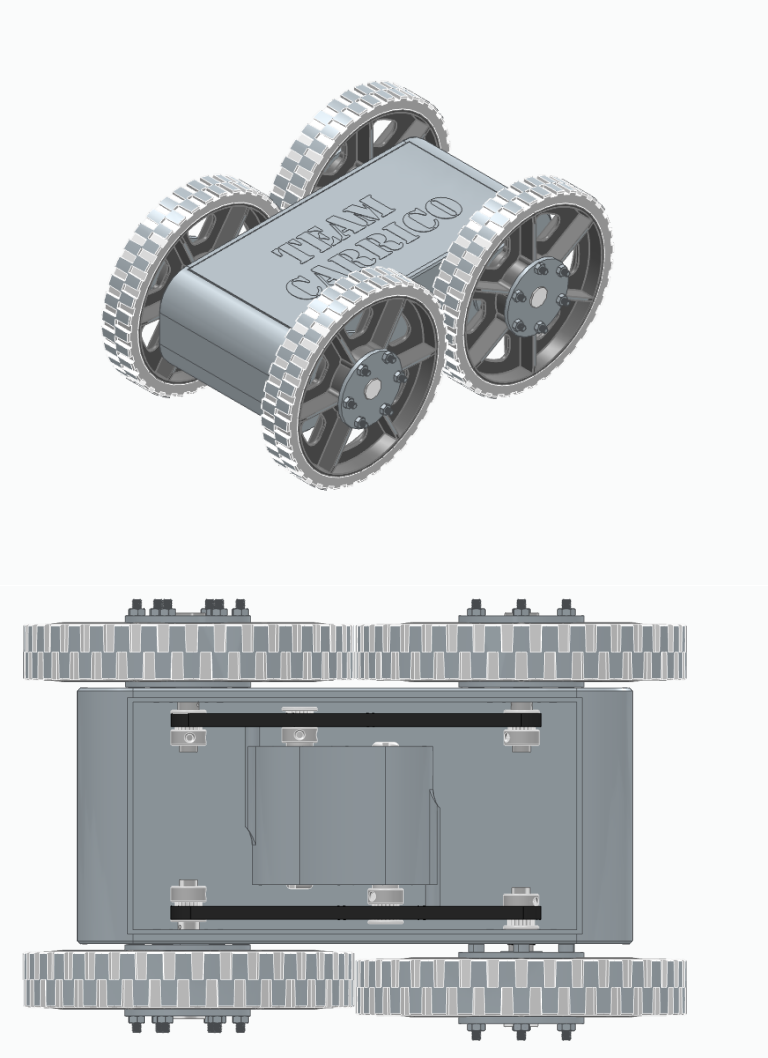


Figure 22. CAD Models for the Car Prototype

cross axle to fit onto the hub. This axle will run on an oil-less bearing mounted into the side of the car and have the belt pulley attached onto the end of it. This is the CAD of the first prototype of the car design.

Using duct tape as a binding agent, we created the first autonomous car prototype using the components we received at the beginning of last semester. The car is controlled using a single joystick, and only has one speed. The motors are driven using the L293D H-Bridge Motor Driver. It is powered by two 9V batteries wired in series, and uses a 7805 Voltage Regulator IC to step down the input voltage of 18V to 5V to power the Arduino and L293D Logic. It communicates wirelessly with the controller through an HC12 transceiver.

The controller uses a single joystick input to send commands. It is powered by a 9V battery, regulated to 5V output by a breadboard power converter. It uses the HC12 transceiver to communicate with the car.

Communication is only one way in this prototype. The controller issues commands, and the car receives and executes (more details about communication protocols are presented in the next section). The maximum distance for a stable response is 30 feet. The arena for the race, Chick's Place, is a lot bigger than this, so we will need to increase our operating distance. Two avenues we will consider are purchasing a larger antenna, and changing the baud rate or frequency of the HC12 firmware.

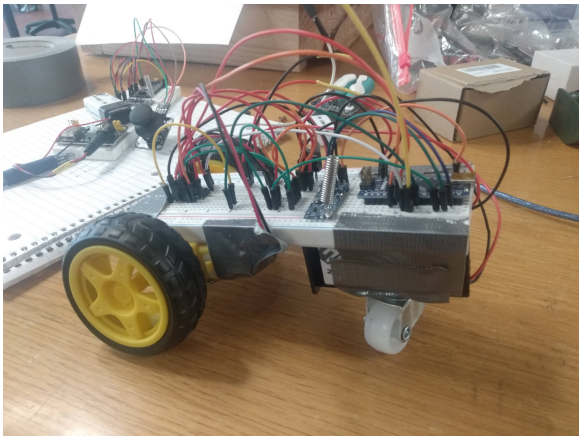


Figure 23. Prototype 1 - Car

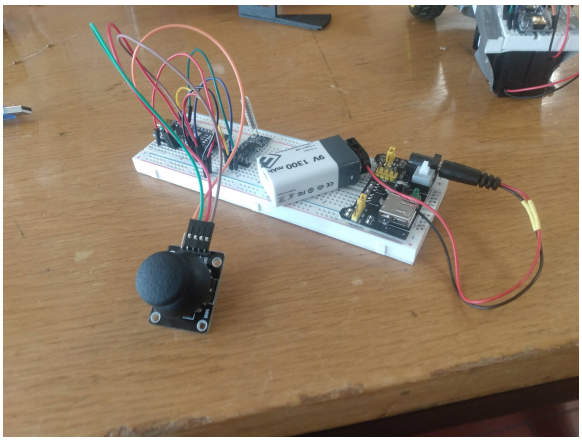


Figure 24. Prototype 1 - Controller

For our first prototype, we are pleased with how it turned out, and information gathered from this first model will be reflected in our design decisions for the rest of the semester. The programs written for the car and controller are given in the appendix (Listings 1 and 2), and will be kept and adapted for the final product.

The first thing I did for this week was to prepare everything for the arrival of the motors and the small gear bearings. I created a jig so I could test the fit of the gears and how the system would work. The jig was made by importing my body design into Prusa Slicer and cutting off the section I did not need. I then printed it off on the Creality CR10. The next step was to create a design for the gears that would allow the bearings to be inserted. I made a few minor improvements to the body as well to better fit the bolts. I also printed off the newest version of the motor mounts that I had adjusted for the new motors.

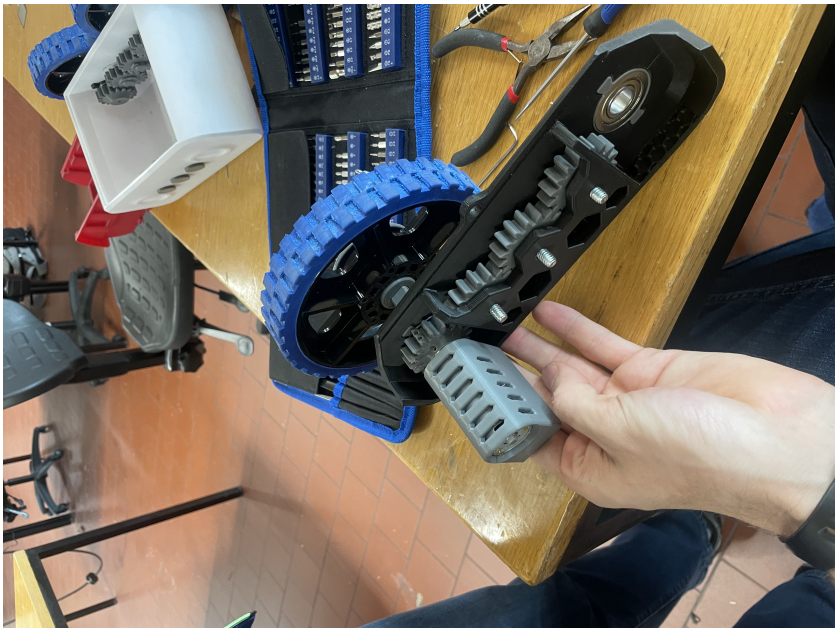


Figure 25. Completed Jig

Once the bearings and motors arrived, I tested the fit of all the parts on the jig and made changes as needed. When I knew that all the parts would fit together, I turned to printing the final body.

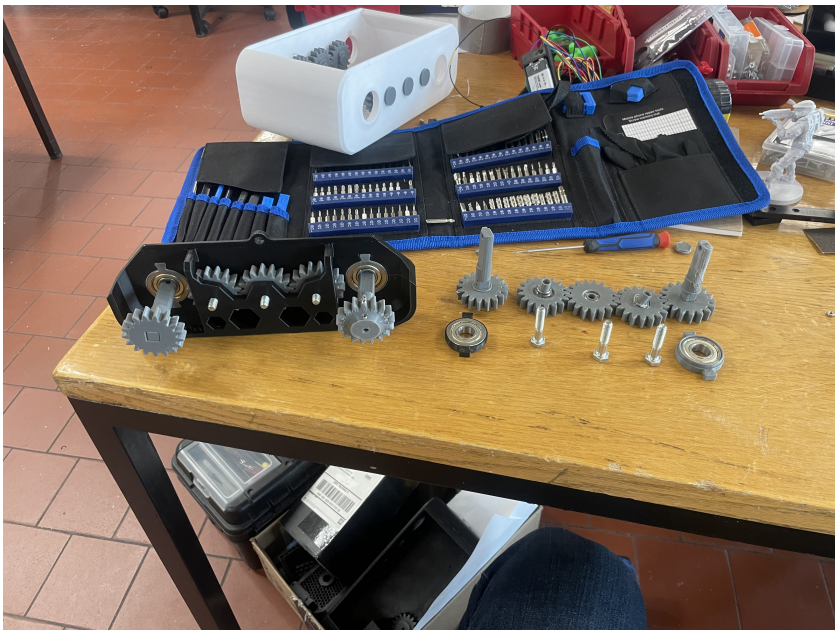


Figure 26. Parts ready for assembly

Here were the CAD changes for the final car:

1. Removed the extra room on either end of the body since it would not be needed. This made the body about and inch shorter.

2. Devised a way to insert the motor axles from the outside of the car. This will allow for the motor assembly to be inserted into the interior of the car without the axle on it, which in turn means there is less room needed to insert the motors. This means that I was able to narrow the body by a full 3/4 of an inch, which was important since we were on the outer limit of the width parameters.

3. Made changes to gear motors to make them function better. Also added a slot to insert a square nut for the set screw.

4. Created a holder for the ESP32 and motor controllers which mount on the bolts that protrude into the body. These bolts serve 3 roles: They hold the idler gears, they hold the batteries under them, and they are attachment points for the motor and ESP32 holders.

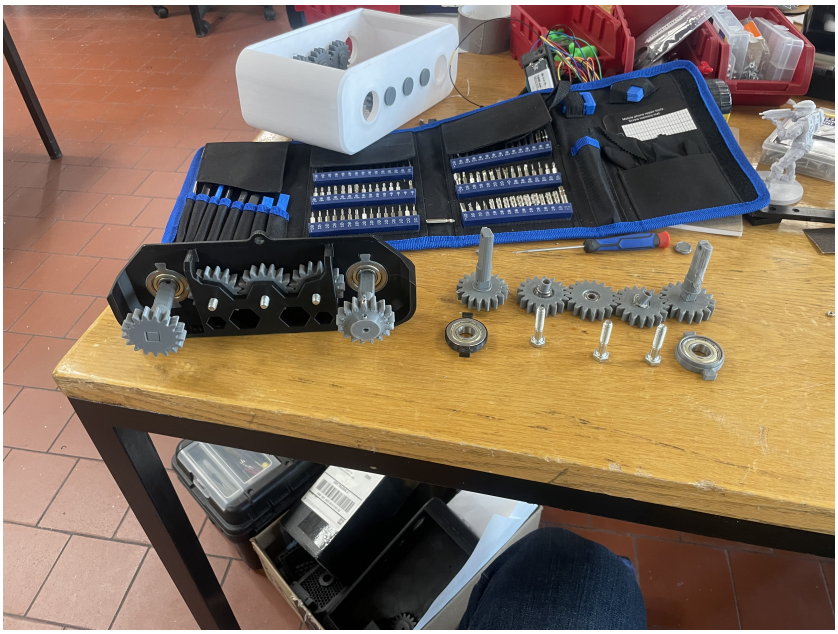


Figure 27. ESP21 Holder

5. I also created and printed brackets for the bin that will house the Pi and controls for the dynamic structure.

6. Lastly, I took Leo's suggestions and updated and printed the remote for the car.

4.4

I started the second body on a CR10 as soon as I knew that all the parts would fit. I had made some changes to the body that decreased its size due to a different method of inserting the motor axles. It was a 18 hour print, but I was able to get a large amount of printing done overnight. There was slight warping at the edges of the model but it did not affect it in any functional way.
In the meantime I had printed all the parts needed for the full car, and once the body was completed I began to assemble it. I sanded down the tight-fitting parts of the model so it would go together more easily. In the end, everything fit together just as it was intended. The one major change I made was to add a small square nut in the motor adapter gear so it the set screw I was using to hold the motor shaft in place would not strip out the PLA adapter.

The assembly process is as follows:

1. Insert the large bearings into the bearing adapters using the vice.
2. insert the bearings and bearing holder assembly into the body using the vice.
3. Press the non-motor axle into its place and make sure the retaining ring is flush with the side of the bearing. The non-motor axle is pressed in from the interior of the body.
4. Insert the motor into the motor mounts using 2 M3 machine screws.
5. Attach the motor-gear adapter to the motor and screw in the set screw.
6. Press in the motor axles. These axles press in from the exterior of the body. This is what allows to the body to be thinner than originally planned.
7. Fit the motor mount assemblies onto the motor axles. Make sure the set screw holes in the motor adapter line up with the holes in the axle.
8. Screw in the retaining screws though the bottom of the body into the motor mounts. Make sure the amount stays square to the axle.
9. Attach the square-holed non-motor gears to the square section of the non-motor axles.
10. Press fit the small bearings into the idler gears using the vice and a 9 size socket.
11. Place the gears in their locations and slide the 1/4 inch bolts through the holes. But leave them not fully inserted so that the batteries can be placed.
12. Place the batteries on the bottom of the car, under where the bolts were protrude into the body.
13. Push the bolts so they fit flush in their holes. Use a hammer if needed.
14. Insert the motor controllers into the motor controller holder.
15. Push the motor controller holder onto the bolts. Make sure that the longer side is toward the motors.
16. Bolt in the ESP32 into its holder and attach it in a similar manner as listed above on the remaining bolt.
17. Attach the wires and make sure that nothing will catch in the gears.
18. Turn on the motor controller using the included switches.
19. Bolt on the lid.

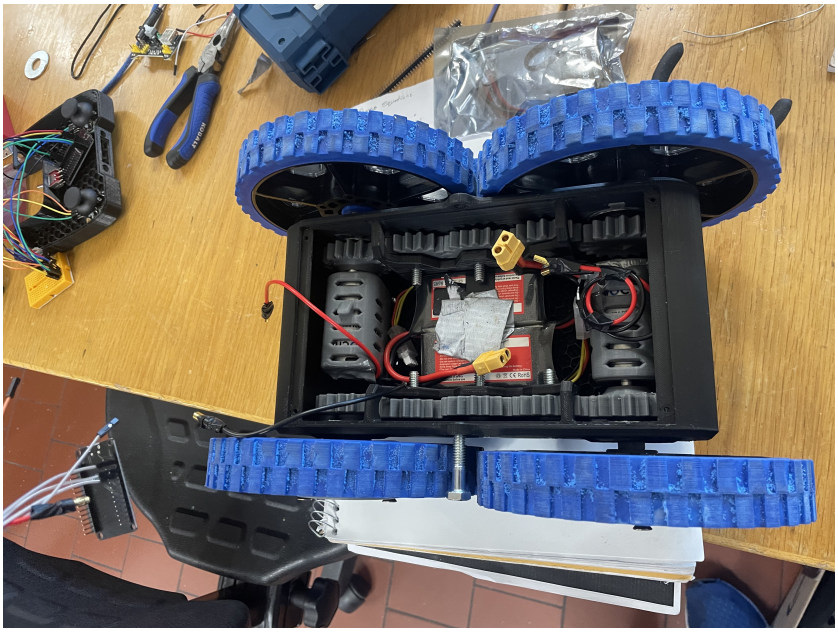


Figure 28. Assembled Car without the motor controllers and ESP32

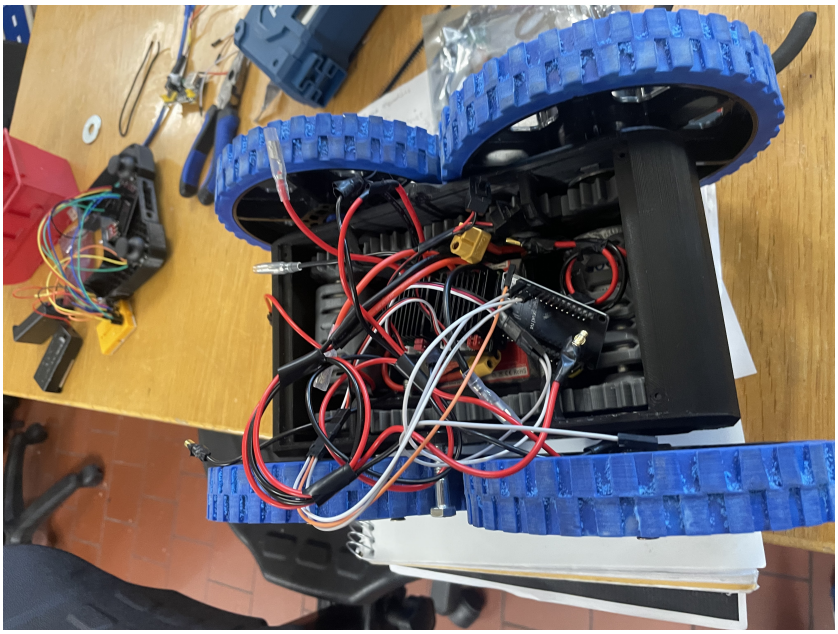


Figure 29. Assembled Car with the motor controllers and ESP32

4.5

I was not able to do as extensive testing as I wished due to Leo being gone over the weekend. However, we noted a couple of potential issues. Much of the problems area are a result of code and electronics, and are outside the scope of my abilities. We were able in our limited testing to fix most of the problems.

1. A motor would begin to stutter under load. One side would be fine, but the other would suffer. Leo believes this is due to bad soldering joint and was easily fixable.

2. The car would sometimes loose connection to one motor, and sometimes both motors. Leo believes that it is an issue with the motor controller communicating with the ESP32.

3. Due to the nature of the motor controllers, the motors have problems reversing, and the current method is a bit unwieldily. This is possible to be fixed in the code, and Leo has a crude but functional reversing mode.

4. There is no slow mode yet, and the controllers are very binary, earlier on or off. I hope that will be able to be fixed in the code.

We also discovered a number of favorable traits:

1. The new motors are very torque-rich, and the car accelerates and climbs well.

2. It does not struggle to overcome the high grip of the tires when turning.

3. The body is robust, without any flexing or clear weak points.

4. The car easily drives while inverted and is able to flip itself on any vertical surface.

5. The large wheels did not rub once, so I will not need to have the rollers I was planning to add to stop of them from rubbing.

4.5.1 Creating the Jig

Using information from last semester, we began designing the next version of the belt jig. It would have a similar form to the first iteration, but would be designed for a much different purpose. This jig was designed from the ground up with the intention to be used in the second prototype. No parts were carried over from previous sketches, and we decided to remake the entire thing so it would give us a greater degree of freedom to make changes and improvements as time went on. The design of the main body of the jig (Figure 32) is designed to print flat on the print bed of a 3D printer while still achieving the desired form. It features holes for the axles and bushings, a sliding slot for the idler pulley, and holes to attach the motor mount. The design of the motor mount (Figure 31) is similar from the previous design, but now with a few improvements and easier to use CAD sketches. While the old design consisted of multiple sketches on different planes all taking definitions from each other, this model uses primarily one sketch with the extruded parts being defined from it. We used some advanced 3D printing techniques to get this part to print without extra material. It is orientated with the legs pointing upward so the body does not need support material. To remove the need of support material in the center hole, we used a process known as sacrificial bridging. See appendix for more information on that topic. We also designed custom supports for the upper overhang so that it did not need support. See the appendix for more information on them. We also used a method called captive nuts to hold the nuts inside the mount. See the appendix for more information on this method. The other parts, consisting of the axles and pulleys, are largely similar to the earlier models, but with dimensions that are updated to the newest design of the car. When completed, the jig will be able to be screwed to a piece of plywood. With two of the jigs, we can test the performance of the motors and belt system (Figure 33).

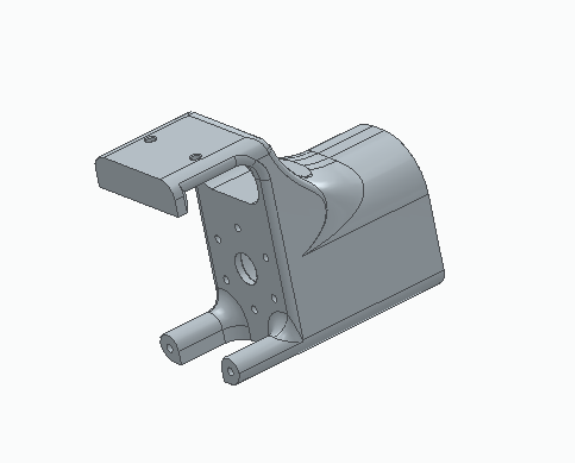


Figure 30. Motor Mount

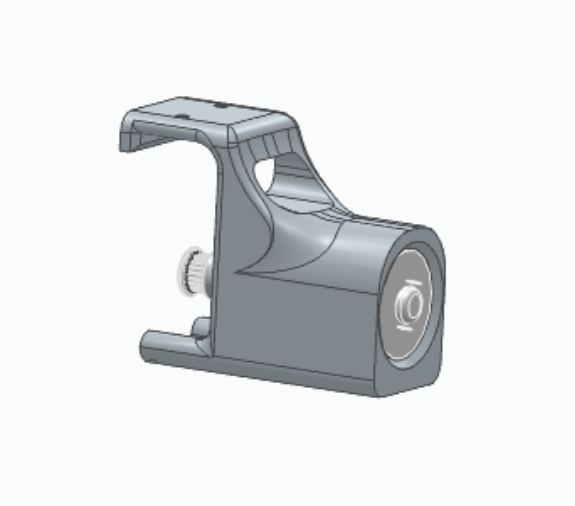


Figure 31. Motor in Mount

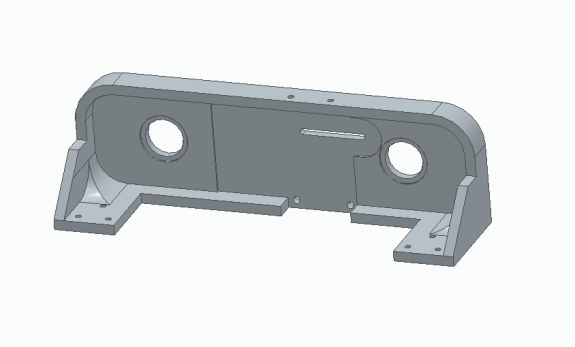


Figure 32. Test Jig Body

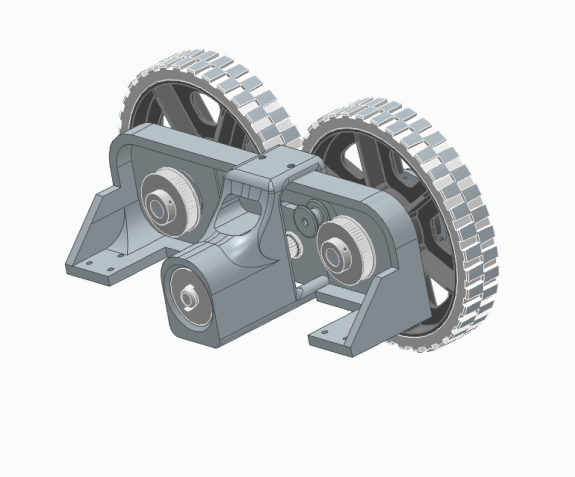


Figure 33. Completed Test Jig

4.5.2 Arduino Controls

The interface between the pilot and car is a handheld, wireless controller with an Arduino as the brain. The controller follows the classical RC car design of twin sticks for controlling, and three face buttons to change speed. However, since the car has differential steering, we found it was more intuitive for each stick to be bound to one side of the car. The left stick will control the forward and backward motion of the two left wheels, and the right stick controls the right wheels. It will operate like a tank does.

Information exchange is in the form of a datastream with numbers representing stick state and speed, separated by delimiting characters. Data loss is minimal, and maximum range hasn't been found yet, as the controller and car can communicate through the entirety of Chick's Place, the race way.

The controller is powered by an internal 9V battery, which can be removed to charge, and uses two joystick modules and three push buttons for input. There is an LED to allow the operator to know if the remote is functioning properly.

4.5.3 Manufacturing the Body and Powertrain

The wheels are slightly too large for the body we designed, so we used the lathe to trim the wheels down to a operable size. Our first attempt involved using the vise grip on the lathe and closing it around the main hub of the wheel, then applying the cutting tool to remove the material. Unfortunately, the TPU material that creates the outer part of the tire is very soft, and it tends to flex and tear instead of cutting cleanly. Two solutions we explored were slowing the speed of the lathe so that the blade cuts cleaner, or using an abrasive tool to remove material without cutting it.

We ended up creating a custom mount that would attach a dremel tool. We attached the wheel to the lathe shaft using the circular vise. The key part of this process is that the lathe was not turned, since it would spin too fast and not allow the dremel bit to remove the material. Instead, the dremel was moved it to the desired location, and the lathe was turned by hand in small increments. In this way we was able to cleanly remove 0.1 inches from the diameter of two of the wheels.

We assembled and began testing our jig, which produced valuable data. We learned that the idler system originally planned had flaws and would need to be revised. We also discovered that the frictional loss was too high, and we would need to purchase bearings.

We also decided to switch either to a geared system. The belt system would have worked well in theory, but due to our busy schedules we were not able to get it working correctly in the time we had been allotted. Our first option was a four motor system consisting of two motors on each end of the car. The

motors would be placed on top of each other, with the output shaft pointing in opposite directions. 3D printed gears to move the shaft downward or upward so that the shafts would have a common axis of rotation, and the motors would then attach to axles. The downside of this design is that we would have to spend an additional 30 dollars on motors, as well as bigger batteries and additional motor controllers.

Our second option was a two motor system with three gears. This system would have two motors located on either end of the car with output shafts pointing in opposite directions. The shaft would attach to an axle with a gear on it. The gear would connect with an idler gear and then to another gear that would be in line with the other wheel axle. The downside of this design is that the gears need to be of 3 inch pitch diameter, which means the body is considerably taller, resulting in less ground clearance.

The design we decided upon was a two motor system with five gears. This system would work in the same way as the previous design, but it would have 5 gears instead of three. This would allow the gears to have a smaller diameter of 2 diameters and therefore a shorter body and greater ground clearance.

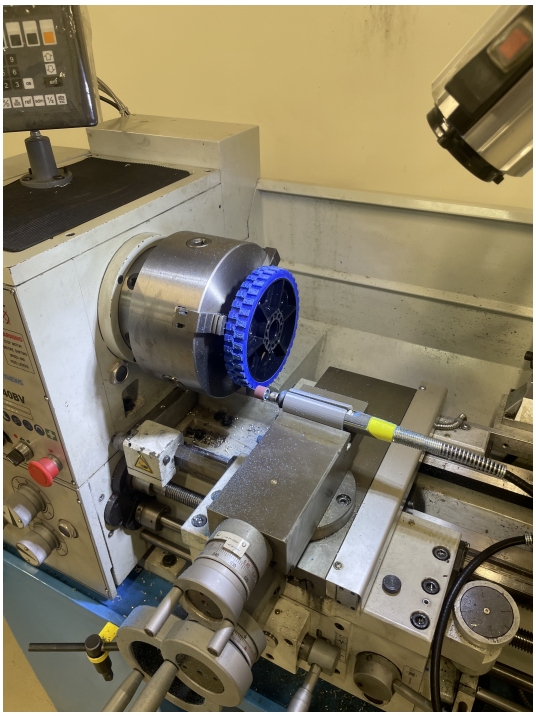


Figure 34. Lathing the wheels with the dremel mount

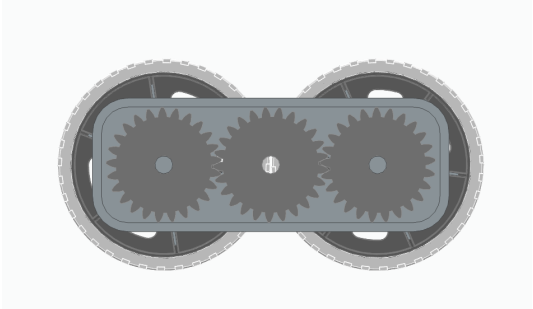


Figure 35. 3 Inch gears (too big!)

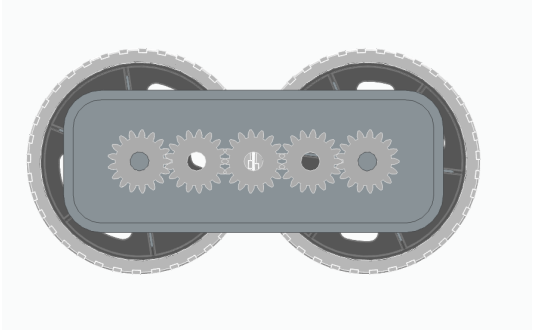


Figure 36. Small gears (just right)

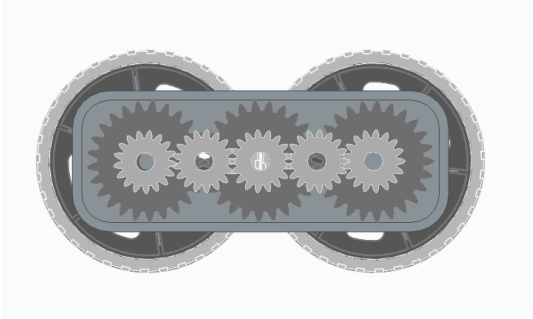


Figure 37. Comparing the gears footprints

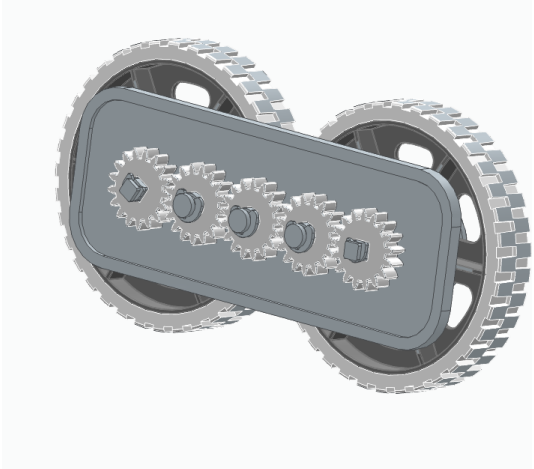


Figure 38. Orthographic Projection of train

Once the gear train passed testing, we began another iteration of the body. This body was designed with the jig profile in mind, and also featured a part which adapts the motors output shaft to the gear and wheel core.

Figure 39. Printed and Assembled Body

We found a few problems with this latest design.

First, there might be too much friction in the idler gears and it would cause some power losses. Also, since the body is printed vertically, the shape of the circles is not perfect, and trying to press fit the bearings into the imperfect holes caused cracks along the layer lines. We had made the body narrower in an attempt to make that car more nimble, but found that it was too narrow and there would be too little room for the motor and gears.

To solve these issues, we decided to use ball bears on the gears, created cutouts in the body to house the inserts, and widened the body by 1/2 inch.

Figure 40. Insert with ball bearing to reduce friction

The latest body was printed with minimal material, so that we could save on filament costs as we iterated.

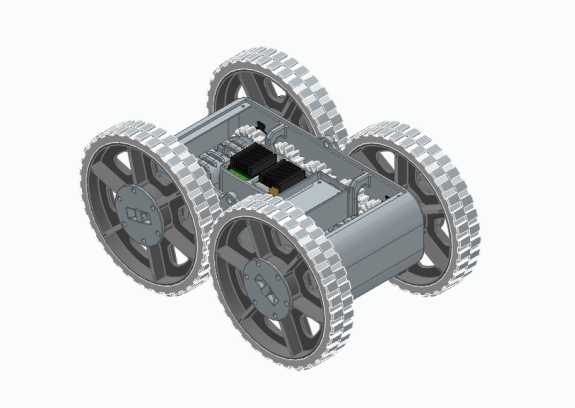


Figure 42. Final Car Design - Side Profile

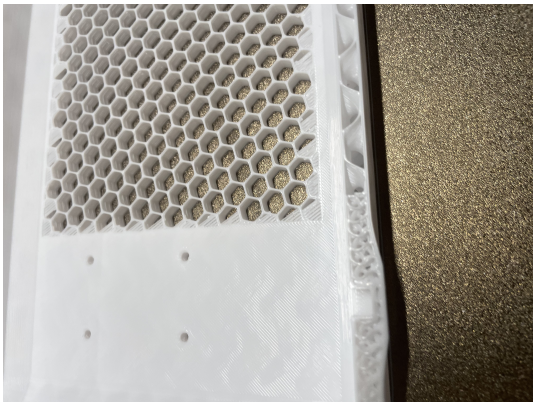


Figure 41. Cutting corners to save on filament for our test body

4.5.4 Final Design

The final car and remote worked fairly well. We made it about $\frac{3}{4}$ of the way through the course before we had to retire our car. A major issue with our car was heat management. Our gears were 3D printed PLA, and the heat buildup in the motors was enough to melt the slot on the gear which interfaced with the motor, causing it to strip out as we drove. We also had some communication issues and issues with the motor controllers. Occasionally, the car would forget the calibration frequency, and we would have to stop the car and slowly ease the sticks in the forward direction. The motors would hum, then we would be able to drive again normally.

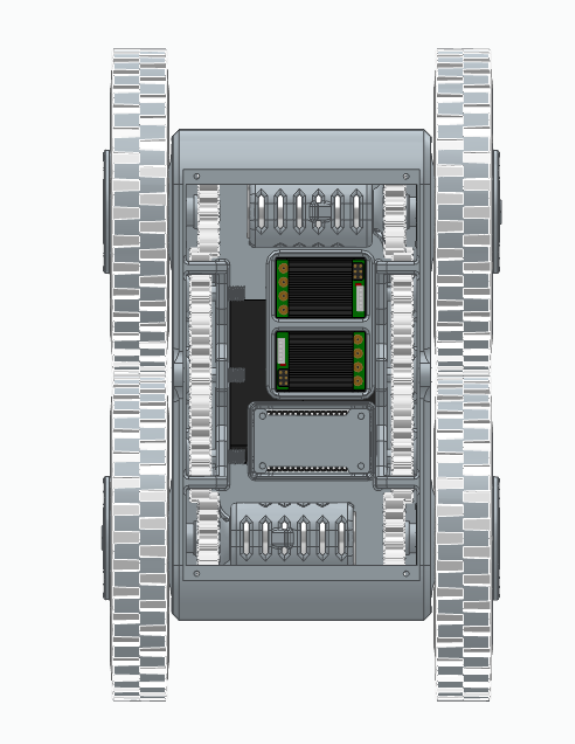


Figure 43. Final Car Design - Internals View

Figure 44. Final Controller Design

Figure 45. Unmelted Gear

Figure 46. Melted Gear

5. Bill of Materials

Table 2. Bill of Materials

Item Name	Item Detail	Cost	Quantity	Total Cost
Spring Loaded Hinges	Luomorgo 4 Pcs Spring Loaded Hinges for Cabinets	$9.99	3	$29.97
Solenoids	uxcell DC 12V 1.33A 4mm Mini Electromagnetic Solenoid Lock Assembly for Electric Door Lock	$11.99	4	$47.96
PLA	3-D Printed Material	$15.00	0.275	$4.13
Pressure Sensors	RP S40 ST High Accuracy Thin Kraftsensor Pressure Sensor Pad Film Pressure Sensor Force Sensor for Intelligent High End Seat	$9.34	2	$18.68
Raspberry Pi		$30.00	1	$30.00
Arduino		$10.00	1	$10.00

Wood		$0.005	1596	$7.98
Wood Boards		$0.005	3637.535	$18.19
Chicken Wire	20ft x 2ft	$10.000		$10.00
Nails/Screws		$5.000		$5.00
PVC	0.5” + 0.75” Diameter			$8.18
Wheels		$13.000	4	$52.00
Aluminum Core	estimated cost	$5.000	4	$20.00
Motors	Greartisan DC 12V 120RPM Gear Motor High Torque Electric Micro Speed Reduction Geared Motor Eccentric Output Shaft 37mm Diameter Gearbox	$15.000	2	$30.00
Belt	Zeelo GT2 Closed Loop Timing Belt Rubber 2GT 6mm 3D Printers Parts 400 mm Synchronous Belts Part - (4 Pc)	$14.000	1	$14.00
Gear Bearings	uxcell MF128ZZ Flanged Ball Bearing 8x12x3.5mm Double Shielded Chrome Steel Flange Bearings, 10pcs	$9.000	1	$9.00
Axle Bearings	uxcell FR8ZZ Flange Ball Bearing 1/2” x 1-1/8” x 5/16” Shielded Chrome Bearings 4pcs	$8.990	1	$8.99

Belt Pulley	3Dman GT2 Pulley 20 Teeth 8mm bore for 6mm Width 20T Timing Belt Pulley Wheel Aluminum - 10 PCS	$9.000	1	$9.00
Motor Controllers	Amazon	$13.990	2	$27.98
1.5kg of PLA	1.5kg Sunlu PLA+	$15.000	1.5	$22.50
New Motors	New Motors	$15.000	2	$30.00
Batteries	Amazon	$33.980	1	$33.98

Table 3. Budget Cost Analysis

Grand Total	**$447.53**
First Semester Budget	$75.00
Second Semester Budget	$450.00
Total Budget	$525.00
Budget Left	**$77.47**

6. Scheduling and Work Breakdown

We created a Gantt chart (Figure 47) for our project to schedule tasks and keep ourselves accountable as we worked. We know that a project is a chaotic system, and not everything goes according to plan. Our first few weeks will dictate how the rest of the project goes. We plan to use our Gantt chart as a baseline for progress, but pivoting and rescheduling as the project continues.

We also created three work breakdown structures for our three products to help us visualize all the components of our project, and how they fit together.

Figure 48 is our car WBS, and highlights the three main areas of work to be done on the car: the electronics, the power train, and the body.

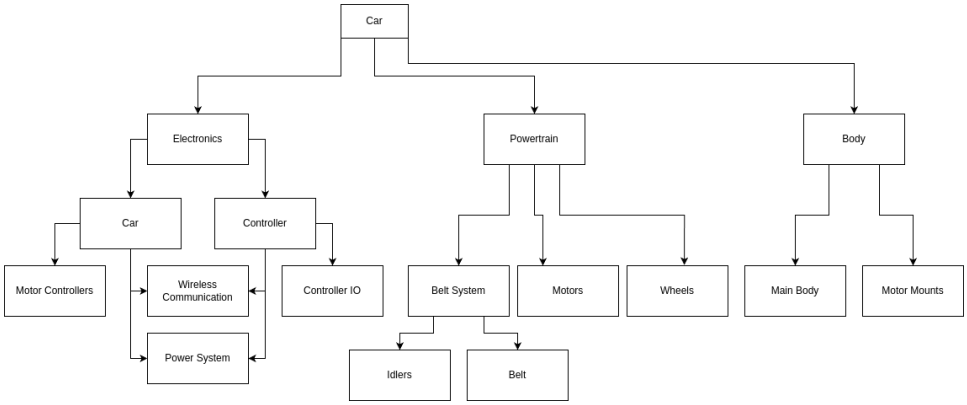


Figure 48. Car Work Breakdown Structure

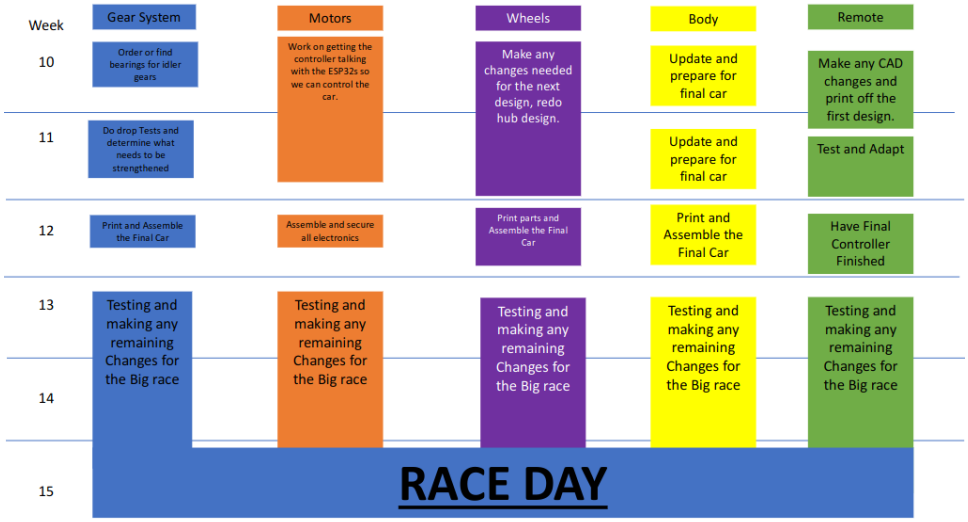


Figure 49. Car Work Breakdown Structure with Timing

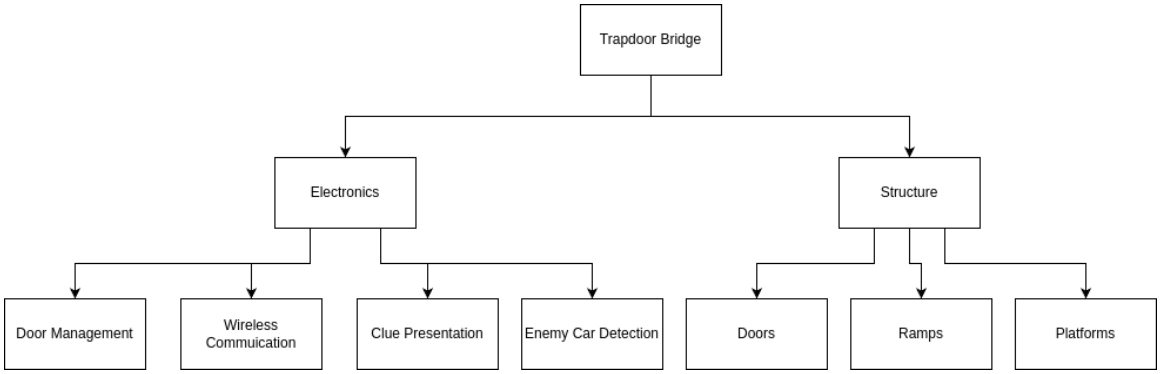


Figure 50. Dyanmic Structure Breakdown Structure

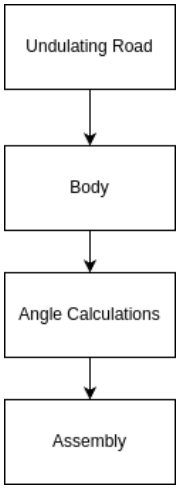


Figure 51. Static Structure Breakdown Structure

7. Conclusions

We have made considerable progress on our project thus far. We have used our research from last semester to inform our decisions and designs for this semester, and have successfully built off of them to create working prototypes for all of our designs. We have learned more about communication and teamwork, as well as accommodating for a teammate's absence. We have flourished under the leadership of Dr. Carrico, and each member of the team feels like they are contributing something meaningful to our end product.

Appendix

Listing 1. Car.ino

```
#include <esp_now.h>
#include <WiFi.h>

const int leftPin = 2;
const int rightPin = 4;

// setting PWM properties
const int freq = 390;
const int leftChannel = 0;
const int rightChannel = 1;
const int resolution = 8;

int zeroThrottle = 90; //90 Needs to be between 66 and 101.
int maxThrottle = 210; //120 seems to be where the maximum is. Above this doesn't give any
    extra speed.
int minThrottle = 60; //60 This represents the "reverse" speed.

bool shiftLeftLow = false;
bool shiftRightLow = false;

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
    int left = zeroThrottle;
    int right = zeroThrottle;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.println("Left: ");
  Serial.println(myData.left);
  Serial.println("Right: ");
  Serial.println(myData.right);
  Serial.println();
```

```
//Left Reverse
   if (myData.left < 85){

      if (shiftLeftLow == false){

         ledcWrite(leftChannel, myData.left);
         delay(100);
         ledcWrite(leftChannel, zeroThrottle);
         delay(100);
         shiftLeftLow = true;

      } else {
            ledcWrite(leftChannel, myData.left+10);
      }

   } else {
    ledcWrite(leftChannel, myData.left);
    shiftLeftLow = false;
   }


   //Right Reverse

   if (myData.right < 85){

      if (shiftRightLow == false){

         ledcWrite(rightChannel, myData.right);
         delay(100);
         ledcWrite(rightChannel, zeroThrottle);
         delay(100);
         shiftRightLow = true;

      } else {
            ledcWrite(rightChannel, myData.right+15);
      }

   } else {
    ledcWrite(rightChannel, myData.right);
    shiftRightLow = false;
   }

}
```

```
void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi−Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP−NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP−NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);

    Serial.begin(115200);

  Serial.println("Arming!");

  // configure LED PWM functionalitites
  ledcSetup(leftChannel, freq, resolution);

  ledcSetup(rightChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(leftPin, leftChannel);
  ledcAttachPin(rightPin, rightChannel);

  ledcWrite(leftChannel, zeroThrottle);
  ledcWrite(rightChannel, zeroThrottle);
  delay(5000);


  Serial.println("Armed!");
}

void loop() {


}
```

Listing 2. Controller.ino

```
#include <SoftwareSerial.h>

const int rxPin = 10;
const int txPin = 11;

const int m1_left = 2;
const int m1_right = 3;

const int m2_left = 4;
const int m2_right = 5;

const int m1_speed = 6;
const int m2_speed = 9;

SoftwareSerial HC12(rxPin, txPin);

void setup() {

  pinMode(m1_left, OUTPUT);
  pinMode(m1_right, OUTPUT);
  pinMode(m2_left, OUTPUT);
  pinMode(m2_right, OUTPUT);
  pinMode(m1_speed, OUTPUT);
  pinMode(m2_speed, OUTPUT);

  Serial.begin(9600);
  HC12.begin(9600);

}

int x_input = 0;
int y_input = 0;
int speed = 0;

double vx = 0.0;
double vy = 0.0;


String input = "";


void readInput(){
  while(HC12.available()){
```

```
    char in = HC12.read();

    if (in == '*'){
      x_input = input.toInt();
      input = "";
    } else if (in == '&'){
      y_input = input.toInt();
      input = "";
    } else if (in == '#'){
      speed = input.toInt();
      input = "";
    } else {
      input.concat(in);
    }

  }

}

void normalizeInput(){

  double _norm = norm(x_input, y_input);

  if (_norm == 0){
    vx = 0.0;
    vy = 0.0;
    return;
  }

  vx = (double) x_input / _norm;
  vy = (double) y_input / _norm;

  Serial.println(vx);
  Serial.println(vy);

}

double norm(int x, int y){
  return sqrt(x*x + y*y);
}


void driveMotor(){
```

```
  //forward
  if (vy < 0){
    digitalWrite(m1_left, HIGH);
    digitalWrite(m1_right, LOW);
    digitalWrite(m2_left, HIGH);
    digitalWrite(m2_right, LOW);
  }
  //backward
  if (vy > 0){
    digitalWrite(m1_left, LOW);
    digitalWrite(m1_right, HIGH);
    digitalWrite(m2_left, LOW);
    digitalWrite(m2_right, HIGH);
  }

  //turning
  if (vy == 0){

    if (vx > 0){
      digitalWrite(m1_left, LOW);
      digitalWrite(m1_right, HIGH);
      digitalWrite(m2_left, LOW);
      digitalWrite(m2_right, HIGH);
    } else if (vx < 0){
      digitalWrite(m1_left, HIGH);
      digitalWrite(m1_right, LOW);
      digitalWrite(m2_left, HIGH);
      digitalWrite(m2_right, LOW);
    } else {
      digitalWrite(m1_left, LOW);
      digitalWrite(m1_right, LOW);
      digitalWrite(m2_left, LOW);
      digitalWrite(m2_right, LOW);
    }

  }

  int speed1 = map(255 + vx*255,-512,512, 0, 255);
  int speed2 = map(255 - vx*255,-512,512, 0, 255);

  analogWrite(m1_speed, speed1);
  analogWrite(m2_speed, speed2);

}
```

```
void loop() {

  //read from HC12, assign to local variables
  readInput();

  normalizeInput();

  driveMotor();

}
```

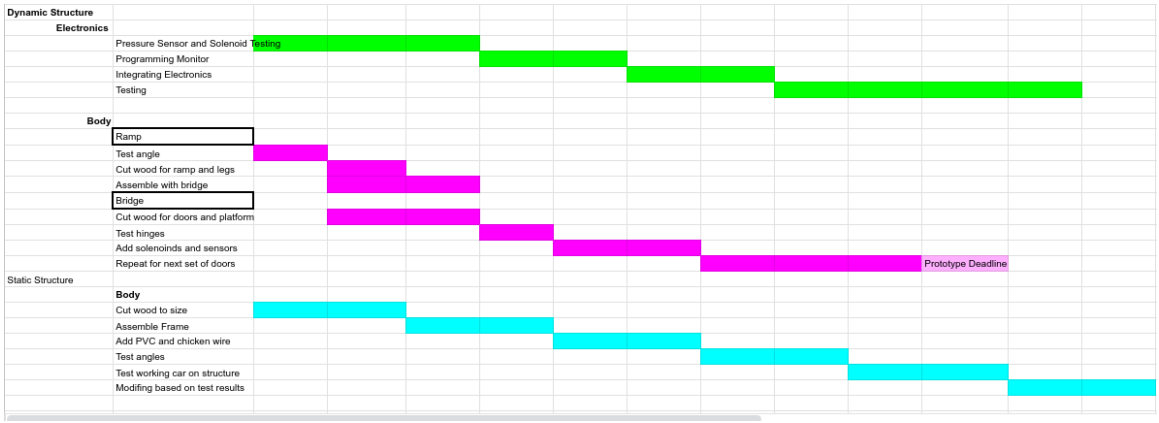


Figure 47. Gahnt Chart for Spring 2024

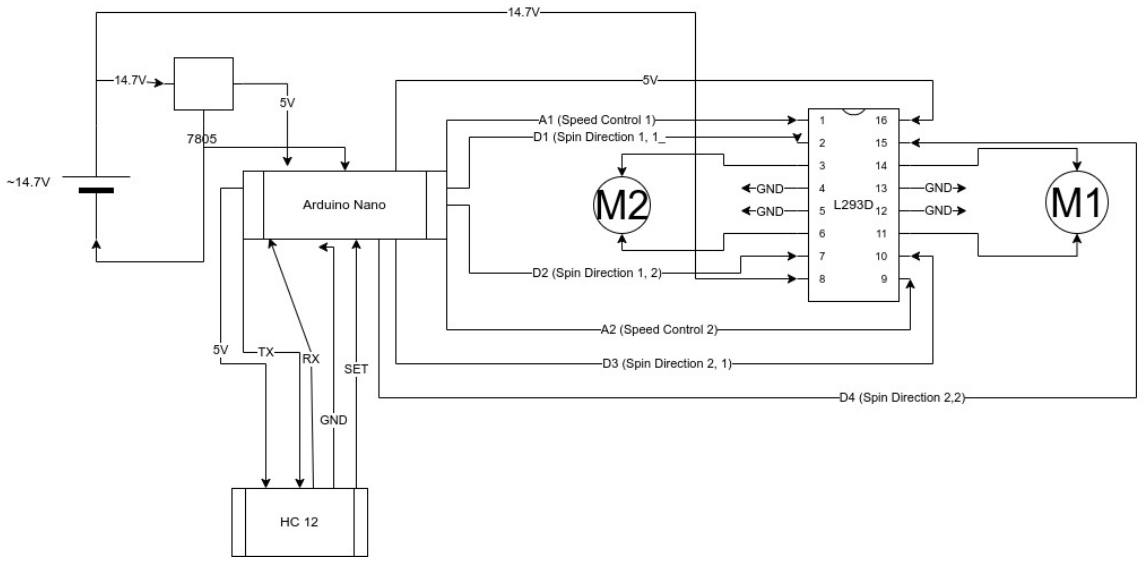


Figure 52. Diagram of car electronic circuit

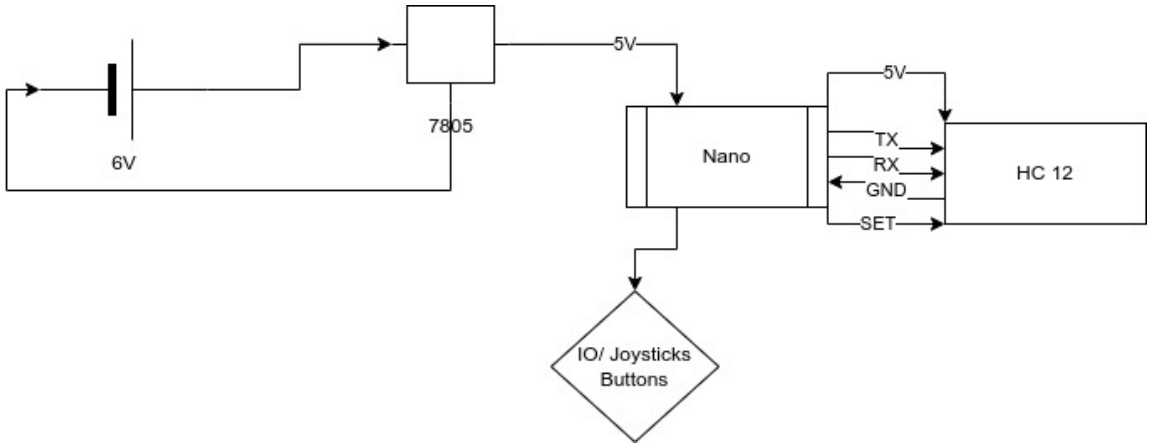


Figure 53. Diagram of car controller electronic circuit

Category	Weight (1-100%)		Design 1	Wieghted Score		Design 2	Wieghted Score
Estimated Cost	5		6	30		5	25
Simplicity	20		5	100		7	140
Ground Clearance	5		7	35		6	30
Robustness	20		6	120		7	140
Tires	20		6	120		8	160
Motors	10		7	70		8	80
Body	10		6	60		6	60
Flexibility	10		6	60		8	80
Totals	**100**			**595**			**715**

Figure 54. Decision Matrix for Car Designs

		Draw Bridge Design		Spring Design		Sensor Arm Design	
Design Criteria	**Weight Factor**	**Score**	**Rating**	**Score**	**Rating**	**Score**	**Rating**
Material Cost	0.1	6	0.6	9	0.9	4	0.4
Durability	0.2	4	0.8	8	1.6	6	1.2
Reliability	0.1	5	0.5	6	0.6	8	0.8
Effectiveness	0.3	7	2.1	7	2.1	9	2.7
Simplicity	0.3	6	1.8	8	2.4	5	1.5
Total Ratings:			**5.8**		**7.6**		**6.6**

Figure 55. Decision Matrix for Trapdoor Bridge Designs

Category	Weight (1-100%)	Orignal Design	Weighted Original Design	New Design	Weighted New Desigr
Estimated cost	10	5	50	4	40
Simplicity	15	10	150	7	105
Flexibility	50	2	100	7	350
Ease of transport	10	7	70	7	70
Durability	15	7	105	7	105
Totals	100	31	475	32	670

Figure 56. Decision Matrix for Undulating Road Designs