Group 7

Jiaxiao Cai, Miaoyan Sun, Yifan Yuan

MEAM 510
Final project report

1. Functionality

1.1 Minimal Functionality Approach

Our design of robot fulfills the constraints and functionalities of the game, and pass all the minimum functional check successfully in Dec-04-2019.

- Dimension Constraints:

	Our Design	Requirement
Overall Dimension	5.9” X 8”	6’’ x 8’’ plan view
Robot Height (other than whisker)	7”	lower than 7”
Whisker Switch Height	8” from the ground	8” from the ground
Whisker Accessibility	top 2” (270 degrees) top 1” (360 degrees)	top 2” (270 degrees) top 1” (360 degrees)

- Physical Capabilities

The basic physical capabilities of our robot include 1) moving around the field to different locations, 2) pushing other robots, 3) triggering the enemy whisker. The car is able to move forward, backward, turn left and turn right with 4 DC motors and controlled by UDP.
During the competition, our robot was able to push other robots and move to different locations in the field. We don’t have a chance to use our weapon in the competition since there are not enemy’s whisker nearby. The demonstration of the physical capabilities could be found in the video link: https://drive.google.com/open?id=1Cm017_C3xycVyhLEk3KxB9pEiDsXCZ6V or https://www.youtube.com/watch?v=pSqs84IXIxk&feature=youtu.be.

- Virtual Attribute

The top hot of our robot shows the health value in the current moment. The weight of our robot is 133g. Thus, the initial health point of our robot during competition is 5HP. The top hot is also able to change color according to which team we are during the competition. Each time the whisker switch is triggered, the robot will lose 1 hit point (HP) which represents its health and there are a “cooldown” period of 0.5s in which repeated triggers on the whisker switch within that period will be ignored. When our robot’s health is reduced to zero, the robot turns off all LED’s and not move. After the death time of 15 seconds has expired, our robot could be restarted. The demonstration of virtual attribute could be found in the video link: https://drive.google.com/open?id=1cJiJ9xE45Dec18TKQ3nf-NRvfjdZ7T-0 or https://www.youtube.com/watch?v=6AmCsdlMeNY&feature=youtu.be.

- Autonomous Mode

We tested our autonomous mode in the game field, and because we are afraid we may hurt our Mega Team’s robots, we did not turn on autonomous mode during the competition. And we checked off the autonomous mode on Dec-12-2019. Our robot is able to identify its location using the Vive Beacon. In order to recognize orientation and make turns, we use 2 VIVE, which are attached to the rear and front part of the robot respectively. In the video, the robot is able to self-localization using VIVE. The video link is https://drive.google.com/open?id=16NLWH5S5SUVshvRsoBR-c1JFV5eTL8l1 or

https://www.youtube.com/watch?v=HjIMckd3rrU&feature=youtu.be. As we can see from the video, when using two VIVE beacon, the reading of one of them may become unstable. Therefore, we use the VIVE with stable reading to indicate robot location, and incorporate the unstable one to identify orientation.

After identifying its location, the robot is able to compare its x and y location, and compare its location with the target location. Then, the robot will perform path planning to reach target. he detail of the coding architecture could be found in session 4: Processor architecture and code architecture. The video showing autonomous mode could be found in https://drive.google.com/open?id=1BlDZhtQxFlfeppiFvRUzRxo9mCnw8RxH or https://www.youtube.com/watch?v=j5N-poUfIr0&feature=youtu.be.

- Other Minimum Requirement Items
    - Be WiFi controlled: Our robot could be controlled by remote controller via WiFi, which could be found in the previous 2 videos)
    - Have a Top Hat with LED ring and whisker switch integrated into the design: Our design integrated top hat and LED ring, and the details of mechanical design could be found in the mechanical design session.
    - Start when alive signal is received from the Top Hat: During the competition, our robot starts move when it receives a signal from the central control. This could be found in the first video.
    - Stop when health is depleted: Our robot's health doesn't depleted in the competition, but you could find in the second video when health is depleted, both the wheel and the weapon can't be moved.
    - Communicate with the Top Hat over I2C to receive alive signal and receive health status and Display through the Top Hat LED ring: Our Top Hat ESP is able to communicate with the central ESP to receive alive signal and health status. The details of their communication could be found in Processor Architecture session.

1.2 Extra Components

- **Light:** We use two LED matrices to show the game status, which are assembled to each side of the car body. For example, when the robot is dead, the two LED matrices will turn off, while when robot is alive and game is ON, the two LED matrices will turn ON and blink.

- **Sound:** We used the speaker and the memory card to generate sounds. When the robot is dead, it will play music to indicate its status.

The demonstration of light effect and sound effect could be found in https://drive.google.com/open?id=11pOvOTfWW5dWBxAzyDH9l61Gpz3MtRsl or https://www.youtube.com/watch?v=2Qqhy73MTFk&feature=youtu.be.

Figure 1. Lighting on the Car

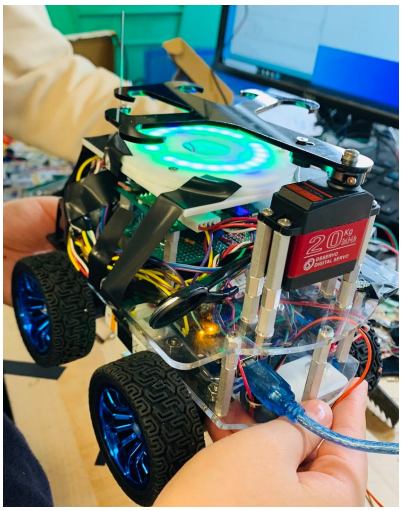


Figure 2. Speaker on the Car

2. Mechanical Design

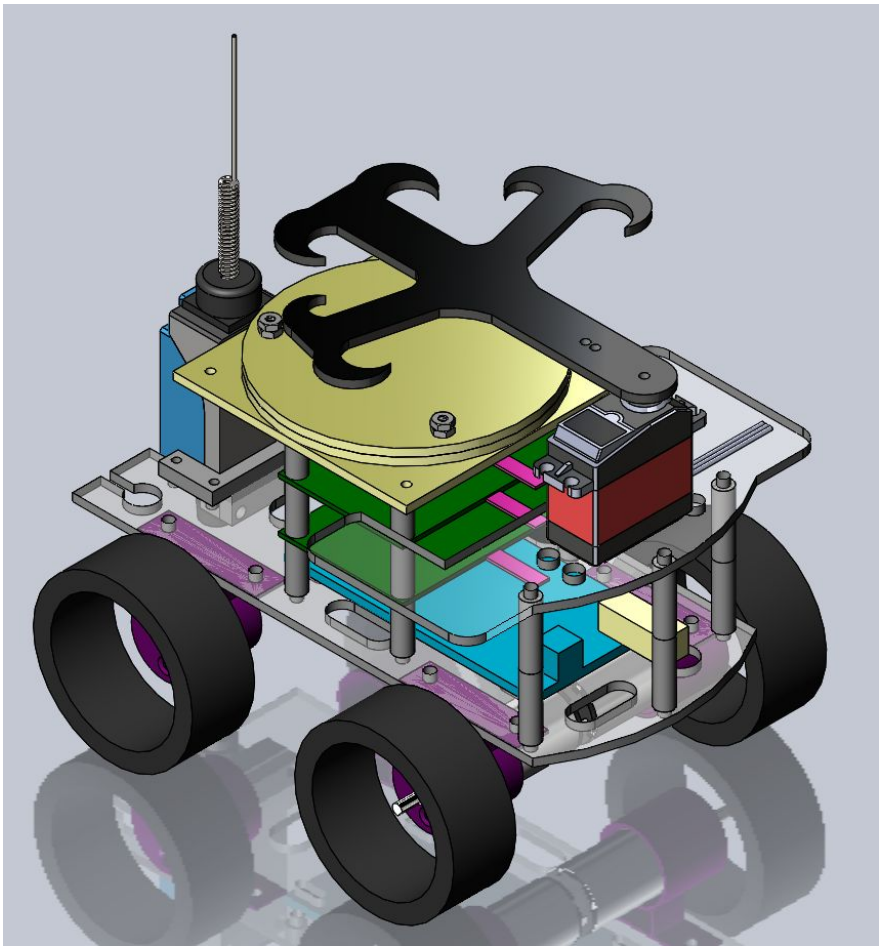


Figure 3. Overall CAD of our car

2.1 Remote Control

The mechanical design of the remote control is shown in Figure 3. The dimension of remote control is 4.6'' x 6'' x 2''.

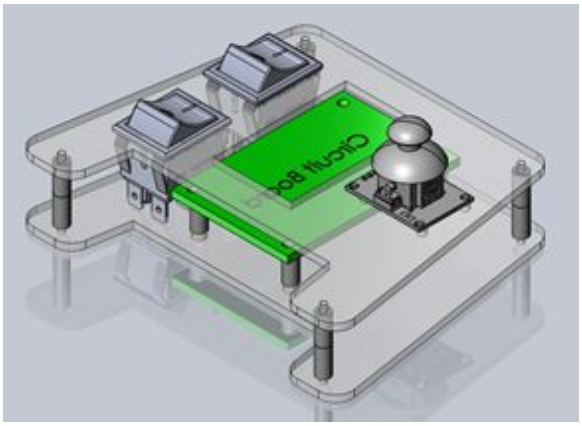


Figure 4.1 Isometric View of Remote Control

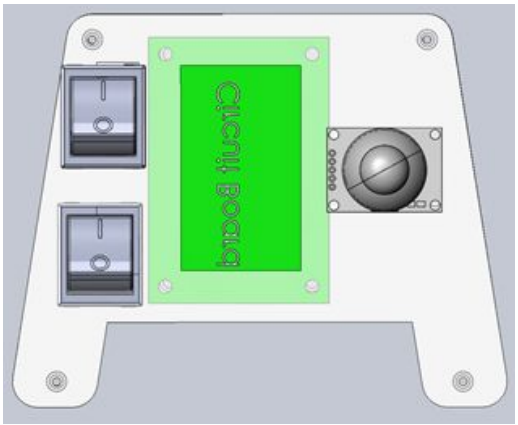


Figure 4.2 Top View of Remote Control

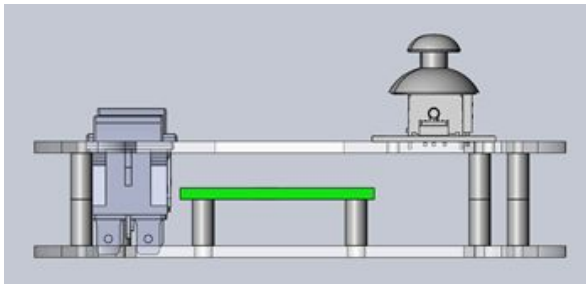


Figure 4.3 Front View of Remote Control

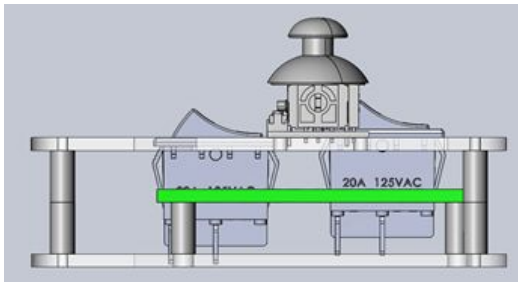


Figure43.4 Left View of Remote Control

Figure 4 Different Views of Remote Control

After fabricating all the components using laser cutting, our remote control is assembled as Figure 4.

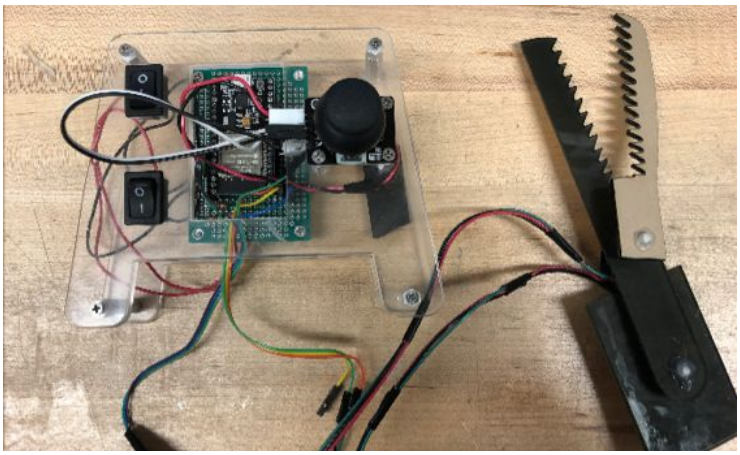


Figure 5. Photo of Remote Control

2.2 Wheels and Motors

The motors and wheels we bought came as a set. After joining them together using the components given, we soon realized that a single side takes about 3.5'', meaning that if we stick to this original design we have no choice but to run on the 8'' side. And this may cause a lot of unexpected troubles along the way, such as passing through obstacles. As a result, we decided to 3D print our own joints and mountings in order to reduce its length and meanwhile reduce the weight as we replace the brass and iron parts.

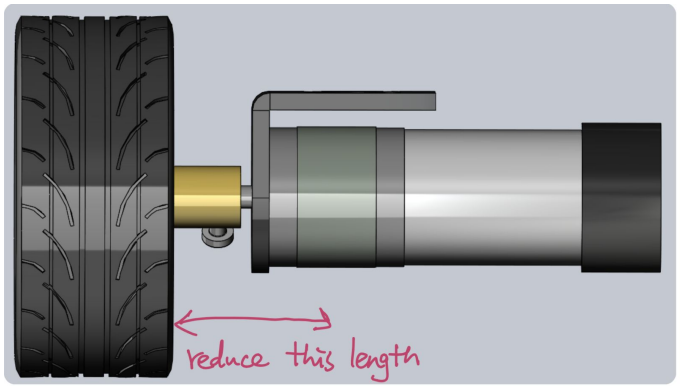


Figure 6: Side view of motor and wheel system

After observation, we found that the motor shaft is a D shaft, and the shaft joint has a hexagon side. Hence, we 3D printed a small light hexagon shaped joint with D-shaped holes to as the joint. In order to further enhance the strength, we applied hot glue to the shaft end and the wheel.

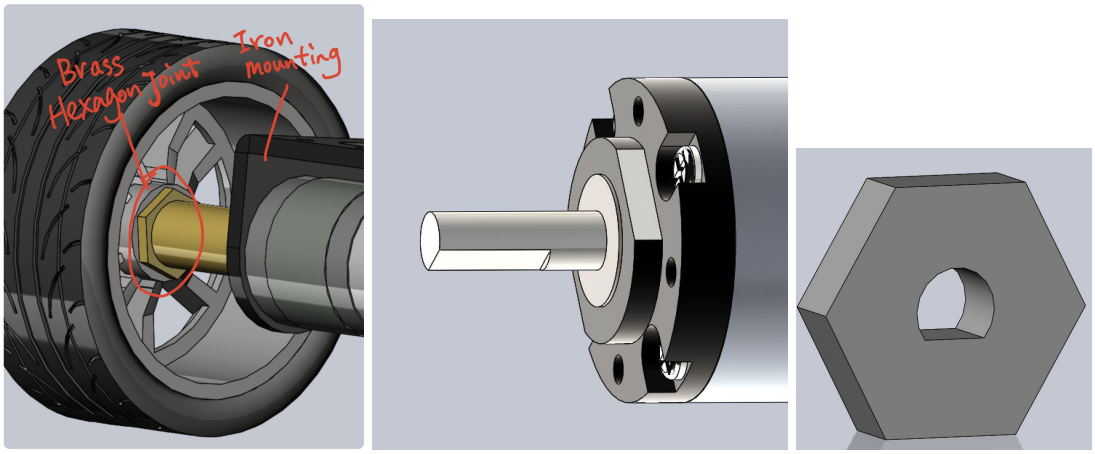


Figure 7, 8, 9 Redesign of motor shaft joint

Figure 10: Picture of glue shaft with motor

We also designed and 3D printed motor mountings to hold the motors in place. In order to prevent the positive/negative ends of the opposite motors from touching each other, we designed two types of mountings. Now opposite motors are in 90 deg to each other.

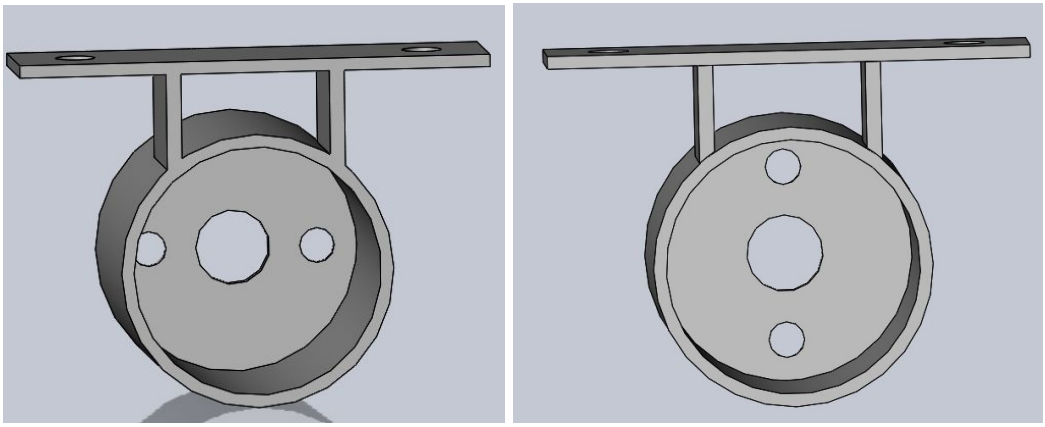


Figure 11, 12: Redesign of motor mounting

Picture below shows the view after redesign. Now the side is less than 6''.

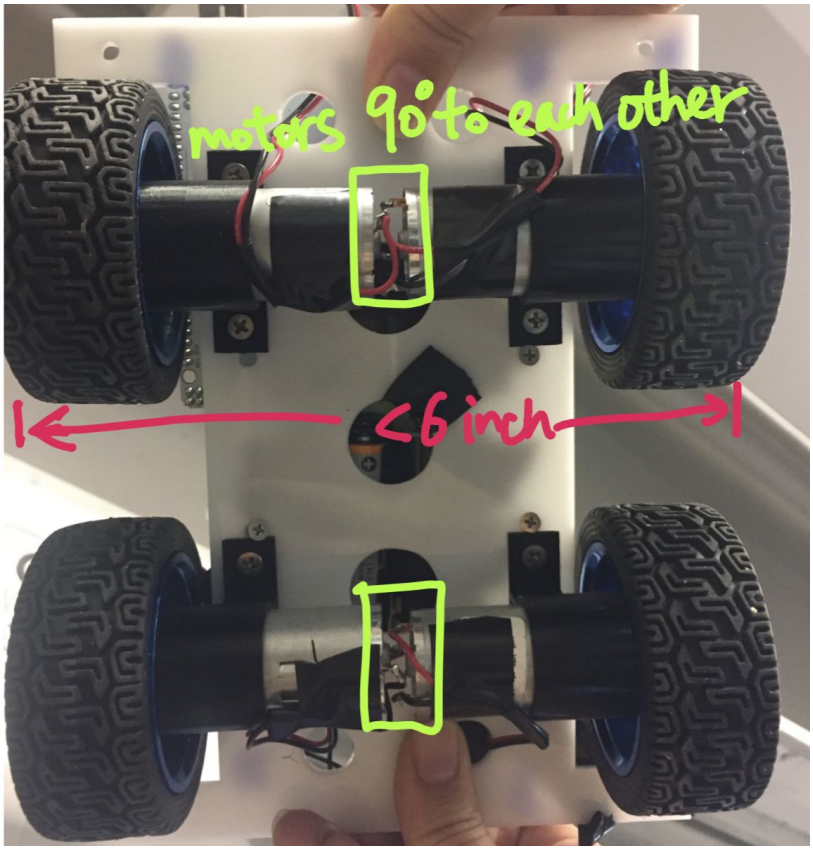


Figure 13: Picture of motors and wheels arrangement after redesign.

2.3 The base

The base board is designed to mount the batteries, motors and wheels below it, and every other things above it. It is the most important board of the car and hence must be carefully designed. Below is a picture of our base board. The left side is the front while the right side is the rear. Big long holes are designed for passing through battery lines. Mega is also catered with 4 long holes as it position may be adjusted to fit the wirings. At the rear side, an open hole is there for the whisker wires as it comes from the bottom of the whisker and goes to the top hat. Other than that, small holes of 0.14'' are cut for screwing. We leave a side open to whisker for easy mounting.

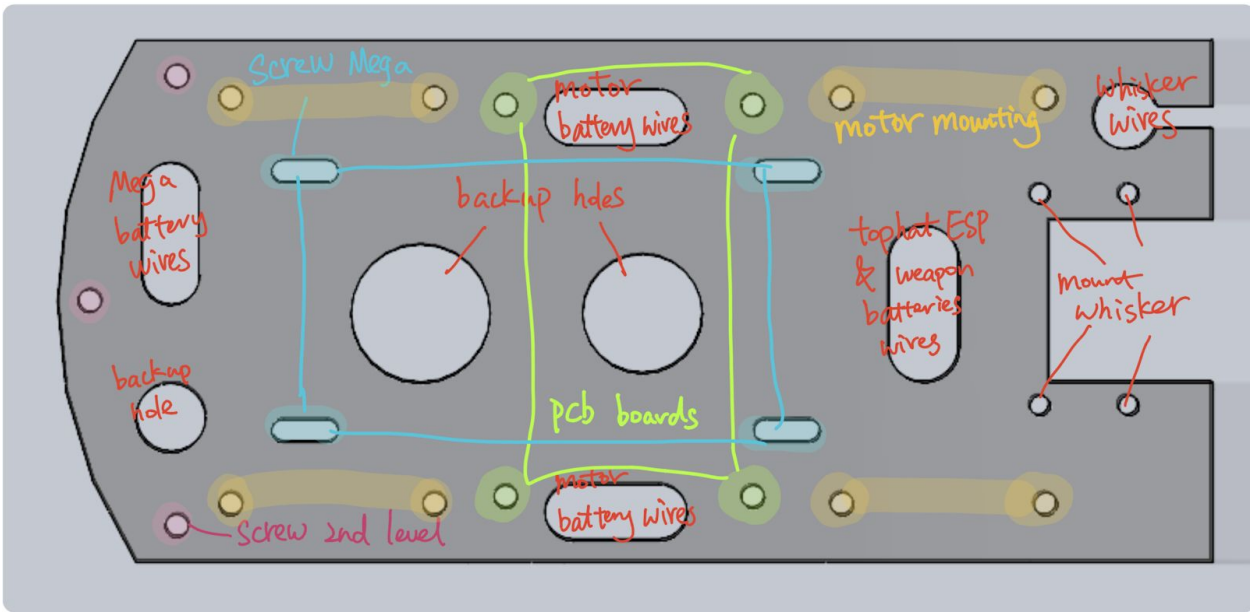


Figure 14: Picture of base board

2.4 Second level board

Below is the second level board. We did not make it a full-range board covering the whole 6'' x 8'' space because we feel it unnecessary. The front part is needed as it allows the servo mounts onto it and it can be used to push the button and other cars during the competition. We leave the left and right side with a little bit of plastic as we believe that this saves space for the wiring and makes it much easier to debug the circuit.

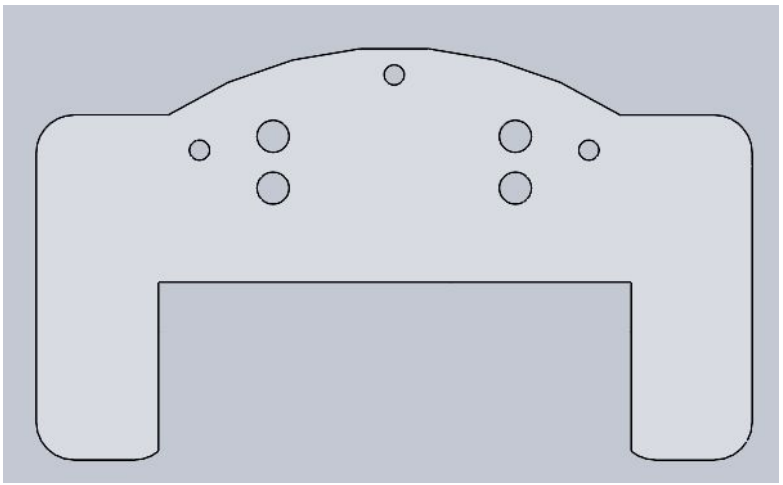
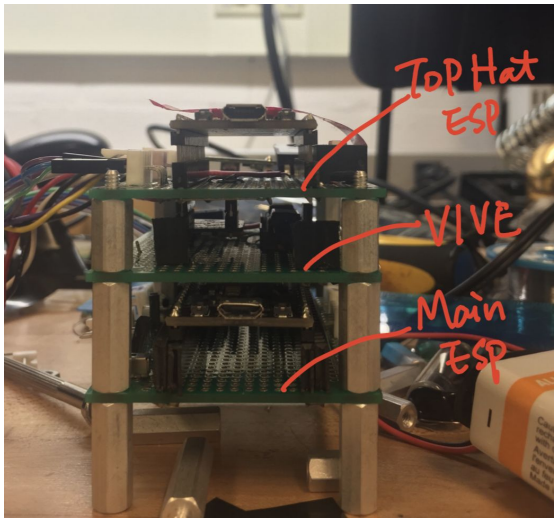
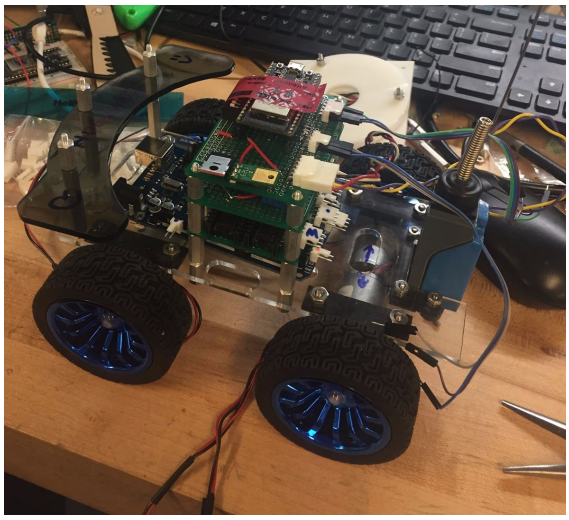


Figure 15: Picture of second level board

2.5 PCB boards and wires arrangement

We soldered the main ESP and 2 motor controllers onto one board, and two vive cells onto the other board. Other than that, we also have the TopHat ESP board. As the Mega board does not need to be soldered, we placed it on the lowest level, and screw it to the baseboard of the car. The other two pcb board stacked on top of one another. Based on the wiring, TopHat ESP board is placed on the top layer as it has little wire connection with lower boards; VIVE is sits in the middle, and the main ESP board stays at the lowest level, just above the Mega board as they form most of the connections. Spaces between levels can be adjusted slightly with different standings so long as the top hat mounted on top of them does not go beyond 6 inches.





Figures 16 and 17: Pictures of board arrangement

2.6 whisker holder

The whisker is placed at the rear side of the car. In order to hold it in place, we 3D printed a holder for it. The top 4 holes are for screwing with the base board, whereas the lower 2 holes are for the whisker.

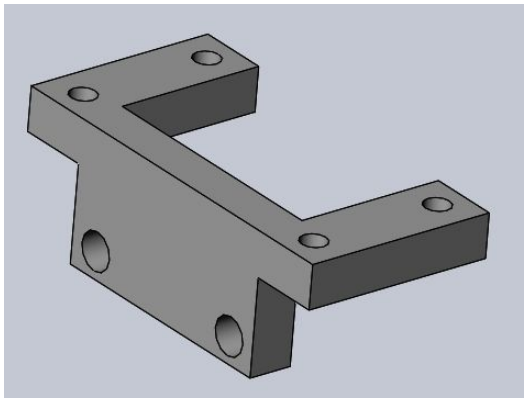
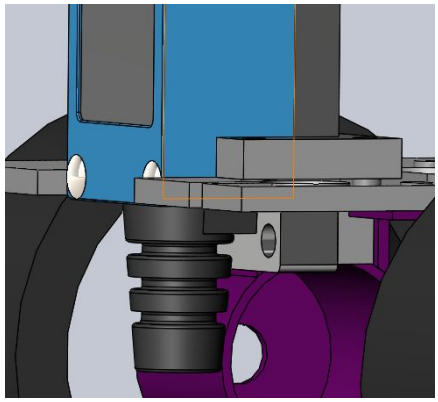



Figure 18 and 19: Picture of whisker holder

2.7 weapon

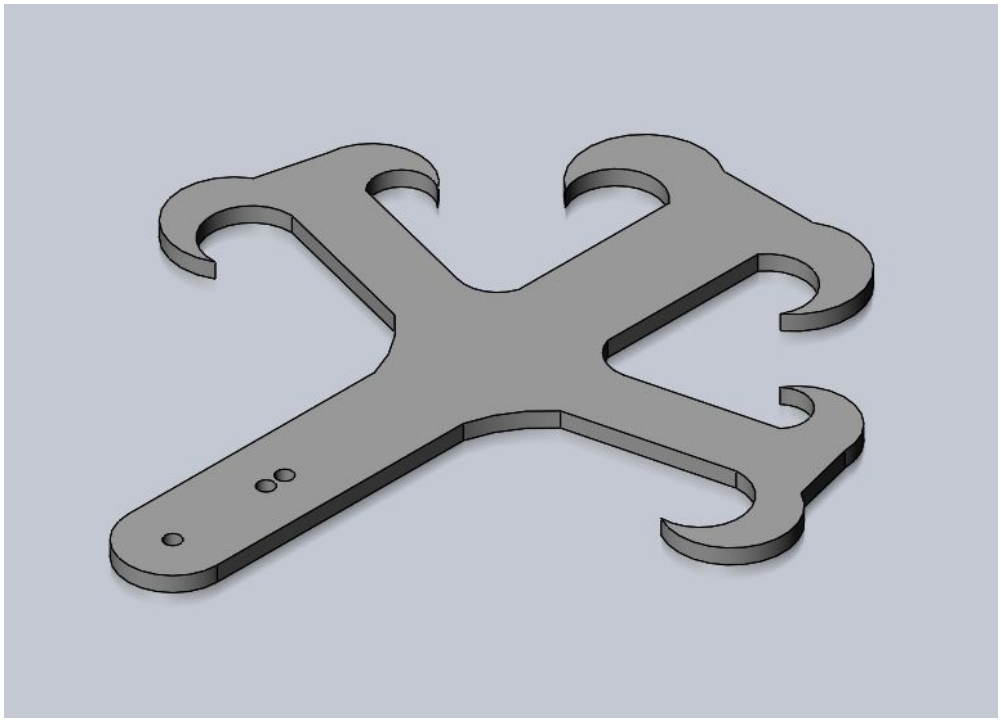


Figure 20: Picture of final weapon

This is our final version of weapon for the competition. We have two versions of weapon before. The first one is use FS90 continuous servo on the top and a SG90 on the bottom. The bottom can control the direction of the weapon and the top one can continuous rotation to hit. The second version is used two SG90 servo, like what we did in the Waldo. One servo on bottom to control the orientation. One top servo control the scissor like weapon to capture other's weapon and make the hit. However, the problem is that the SG90 servo has too low of torque so that the hitting is bot powerful enough as we thought. Thus, finally, we decided to use the DS3218, the servo with 20Kg of torque, and with the hooks as the final version of our weapon.

3. Electrical Design

3.1 Car side

In this project, we used two microcontroller on the car to control the motion of the car: 1 of ESP 32 and 1 of Arduino Mega. The ESP 32 is used for communicate with remote control side and the tophat ESP and the Mega is used to control the motors, the servos, and the light and sound effect. The reason we choose to use the Mega is that we are planning to use 4 wheel drive in this project and it will spend 8 ports to control the two H bridges. And it will spend 1 to 2 ports for servo of the weapon, 2 to 4 ports for sound and light effect, and 2 to 4 ports for vive and ultrasonic sensors. Thus, we finally decided to use the Mega to control everything and ESP only for communication.

The reason we chose to use four wheel drive with new motors is that our car in the Lab 4 doesn't run so fast with the 2 wheel drive. One thing is that the control of potentiometer limited us to use 9V or 12V powering our car. Another reason is that the motor we used last time is not stable with lower torque. Besides, with the four wheel drive, the car can run fast and be more powerful in the competition. Thus, we used the method of differential to control the speed of the motor to control the motion of our car.

The original design of our car is to have 2 H bridges to control the motion of left and right side; two parts of the weapon and each controlled by one servo, same type that we used in Walso; and two vive to detect the position of the car; four ultrasonic sensors to detect the obstacles in each side; one speaker' several LEDs. After we did the minimum requirement check off, we found that the current weapon design is not robust and cannot attack efficiently. Thus, we changed our design into the one 20 KG servo to control one part of the weapon. For the motor, we used to think to have small differences in the speed between the front and back wheels but after attempations, we gave up this idea and use the same speed for front and back wheels. As for the vive and autonomous, we tried to do it before the competition but we were running out of time on it so that we did it after the competition.

Besides, since we used the mega board for the controlling the main function of the car, we firstly use the jumper wire to connect the each part of the circuit. However, after we finish the first prototype of our car, we found that using jumper wire will cause noise and unstable. Thus, we soldered the entire circuit again to make the plugs as less as possible. And we also replaced the jumper wires by the molex and plugers. The photos of old version and new version of our physical circuit are in the Appendix.

In the car side, we used five 9V battery to powering all the components. Two of the 9V batteries are provided to power the two sides of the DC motors. One of 9V battery is provided for the servo of the weapon. Two of the batteries are proposed to powering the Tophat ESP 32 and the Mega/ESP 32 board. Firstly, we decided to powering everything by 9V batteries because of the weight. Since the DC motors and wheels we chose are heavy and the LiPo battery only provide 7.4V voltage with double weight, we chose to use 9V batteries for everything without any LiPo in our car. However, at the end, we found that 9V batteries is limited since the current is not enough for ESP 32 and Mega after using a while, especially for Tophat ESP 32. But we didn't have time to change the batteries into LiPo at that time. When the batteries are lower than 9V, they would not work for Tophat ESP 32; when the batteries are lower than 8.5V, they would not work for the main ESP 32 and Mega. This is one of reasons why our car cannot move in the first

day. After the game of first day when I change the battery providing power fo main ESP 32 and Mega, the light of tophat showed up and it can run in the lab.

The other reason the car cannot move in the first day of the game is the some problem of the Tophat code or the central ESP. In the second day of the game, we went to the Wu & Chen for testing before the game starts. Out tophat cannot connect to the WiFi of the central ESP. At the end, Walker helped us to reflash the Tophat code and reset the central ESP 32 and our Tophat ESP 32 works well in the second day of game.

One problem of the project is the Mega board. Since Mega board need the wires to plug in for functioning. The first prototype of our car is using the jumper wire to connect the ESP 32 and Mega board. After we connect every wire we need, we found that we just create the spaghetti wires. Thus, we decided to re soldering the circuit by using molex and plugers to connect. It improved our circuit in arrangement and help us a lot while debugging the circuit. However, it still brought us some problems. In the last game, since we use the battery connectors that is different from others for powering the Mega board and we are not able to put the connector in the middle of the car. Thus, the connector has been dragged out and the Mega lose power during the competition.

For the extra credit, we bought several speakers and use the Mega board to control the sound. We used the note attached in the pitches.h and call them in the function of tone in the main loop. And we used the delay to control the length of each tone of the melody. Since we used delay to control the sound, we decided to put the music in the 15 s respawn time period when nothing of the car can move. Thus, we should write the code to make sure after the music end, the data received is still in the same order as before. Therefore, we use some flag and shift the order in case the order of the data changes.

The schematic diagram of car side is in the Appendix. The detail light module is drawn separately.

3.2 Remote Control side

For the remote control side, our original design is to use one joystick to control the motion of the car; two potentiometers to control the motion of weapon and two switches to control the on and off of the car and the weapon. We used ESP 32 as the microcontroller of the remote control. From the last time in the racecar, we used two slide potentiometers to control the forward/backward and left/right of the car. But it is hard to control so that we decided to use the joystick to control the motion of the car.

However, since we change the design of the weapon from 2 parts to one part before the competition.We only used one potentiometer and one servo as the weapon part. Thus, we only use the second potentiometer and the port 9 on the Mega board to control the new weapon.

3.3 Vive side

In the part of autonomous part, firstly, we used to design using the two photodiode and several ultrasonic sensors to detect. But Since we are running out of time, we decided to test the auto

mode after the competition. Thus, after the competition, we only need the photodiode for the auto mode though we did learnt how to use the ultrasonic sensors.

A circuit diagram is attached at the end of this report. We use an opamp to amplify the signal, followed by a high-pass filter to filter out low frequency ambient lights, then we use a rail-to-rail comparator (T272I) to pull-up the sync and x- y- signals to 5V, while leaving others to 0V. We soldered two VIVE systems onto one pcb board. Upper half is one cell and lower half is the second cell. Output signals go to interrupt pins 18 and 19 on Mega 2560 board. When tested with computer, both cells gives accurate readings. Very big number (7000+) comes out occasionally with the second cell but this has been solved when several “Serial.print()” statements are commented out.

4. Processor architecture and code architecture

4.1 Block Diagram

The system block diagram is shown in Figure 21.

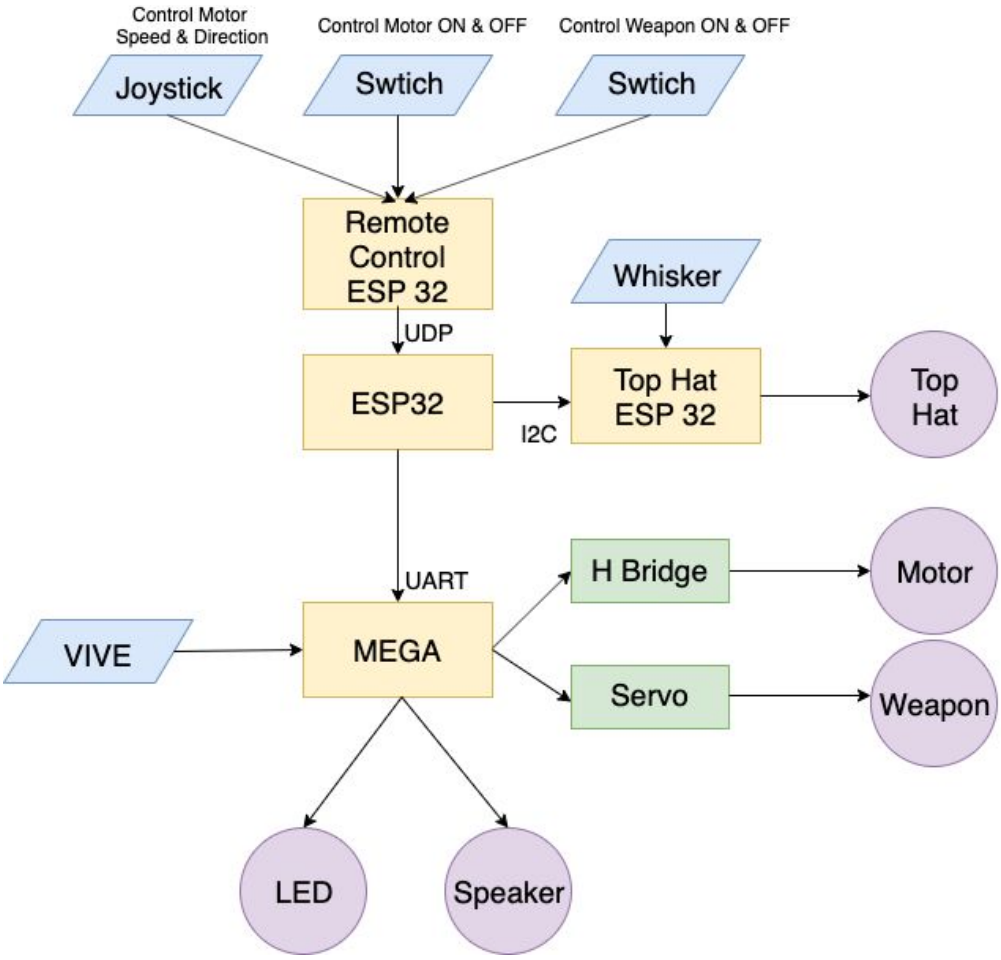


Figure 21. System Block Diagram

The above block diagram shows the overall architecture of our robot and all electronic components that we used for constructing the robot. We used 4 Arduino boards in total.

The central ESP is used to receive signals from different boards and coordinates the tasks.

In the remote control side, there are two switches and one joystick. The switches are used to control the ON / OFF of the motor and weapon respectively, while the joystick is used to control the direction and speed of the motor. The remote control ESP is communicated with the central ESP using UDP.

As for top hat module, top hat ESP is communicated with central ESP using I2C. The top hat ESP receives whisker value as input, and displays health value in top hat.

In terms of the MEGA, it communicates with central ESP using UART. And MEGA is able to control motor via H bridge, control weapon using servo, display lighting and play sound. MEGA also connects VIVE to receive localization information.

4.2 Coding Architecture

4.2.1 Coding Architecture of Remote Controller

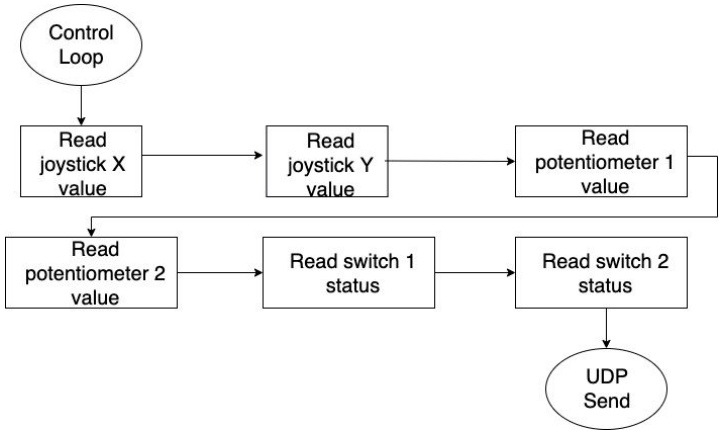


Figure 22. Coding Architecture of Remote Controller

The coding architecture of remote controller is rather straight-forward. Inside the loop, it will read the X, Y value of joystick, read the value of potentiometer, and the status of 2 switches. It will then pass these five data to the central ESP using UDP. The code of remote control could be found in code submission file with file name *remote_control.ino*.

4.2.2 Coding Architecture of VIVE

```
/*
 * 7 falling edges are needed to calculate for 6 intervals
 *  |--sync--|--sync--|-x-|--|-y-|--|
 *  0          1          2   3  4  5  6   index of time_record
 *  |time-diff| is used to distinguish sync-sync interval from the others
 *  sync-sync interval is more than 8000 us, other 4 are less than 8000 us.
 */
```

We use a counter called “falling” to count the number of times that the system catches a falling edge. As shown in the above drawing, 7 falling edges are needed to calculate for 6 intervals. Then, an array called “time_record[7]” records down the system timing of detecting the interrupt. A second array called “time_diff[6]” calculates the timings for the 6 intervals from “time_record[7]”. A third boolean array called “check[6]” then iterates through the time_diff[6] and decide which interval is sync and which is not. Interval between two sync pulses is larger than 8000ms while others are far less than that. In the end, the check[] array is looped through to decide which position holds x and which position holds y.

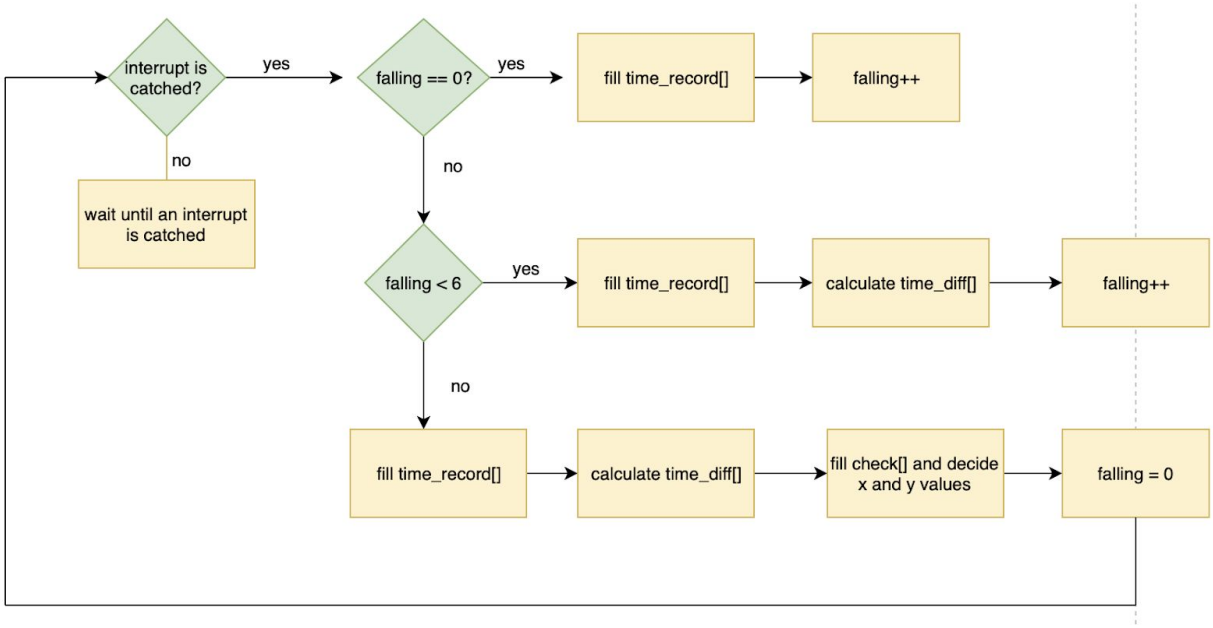


Figure 23: Code Architecture for VIVE system

4.2.3 Coding Architecture of Autonomous Mode

We could set any point in the field as our target location, and place our robot in any location in the field. Our robot was able to conduct path planning in random case. We will use the rule of competition to elaborate our code. Under such circumstances, the target point is the blue button in hill and the starting point is any random point near the blue nexus. The sketch of field is shown in Figure 24. The VIVE reading in different locations in the field is shown in Figure 25.

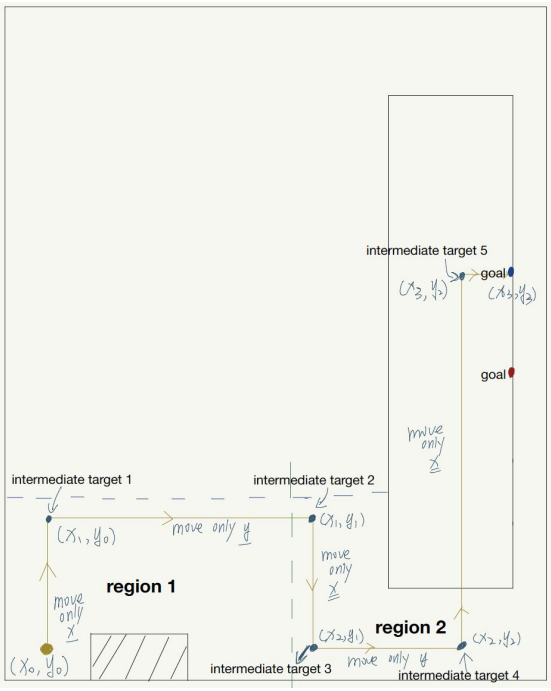


Figure 24. Sketch of Field

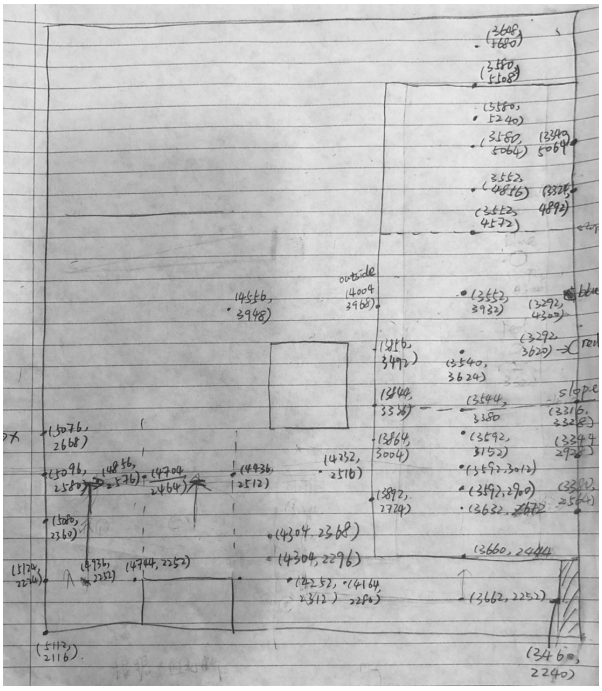


Figure 25. VIVE Reading in different locations in field

The code defines two initialization case: 1) robot being placed in region 1 or 2) robot being placed in region 2. If the robot is placed in region 1, it will define 5 intermediate target (intermediate target 1 - 5), while if the robot is placed in region 2, it will define 3 intermediate target (intermediate target 3 - 5). After that, the robot will compare its location with the intermediate target. And each time, the robot will move only in either x or y direction.

The detailed coding architecture of autonomous mode is shown in Figure 26 and Figure 27. We define 4 subroutines to control the orientation of the robot. Inside the main loop, if autonomous mode is ON, the robot will reach each intermediate target one by one. During each movement, the robot was able to use two VIVE reading to compute its orientation and call subroutine to adjust its orientation to pre-design one, and then move towards the target. After reaching one intermediate target, the robot will adjust its orientation again and then move towards the next target. The whole process repeats until robot reach final target.

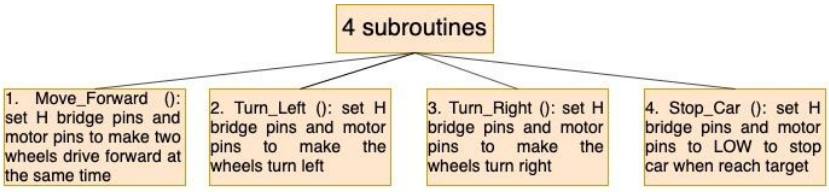


Figure 26. Subroutines of Autonomous Mode

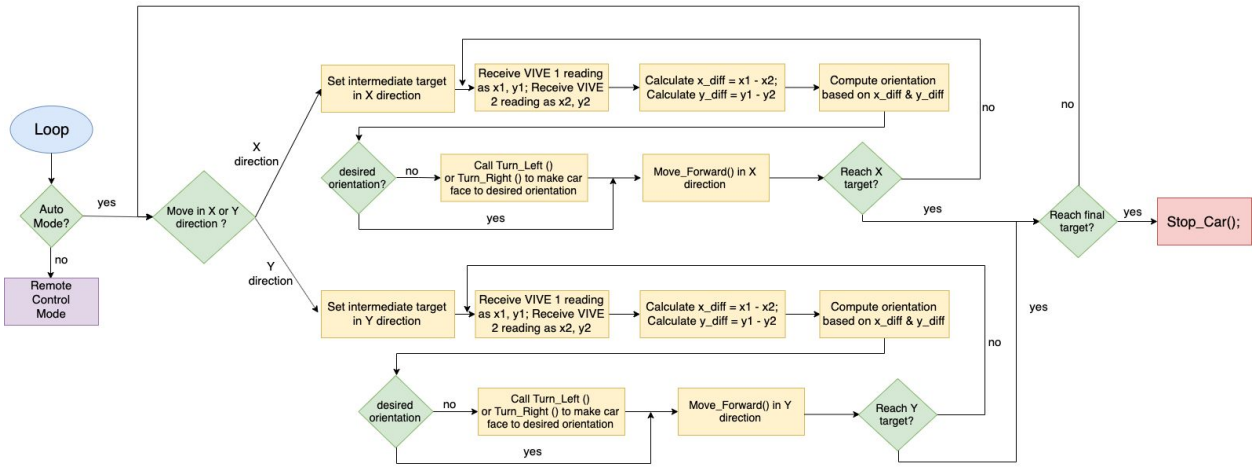


Figure 27. Coding Architecture of Autonomous Mode

4.4 Coding Architecture of the car side

DemoBot- ESP 32 on car

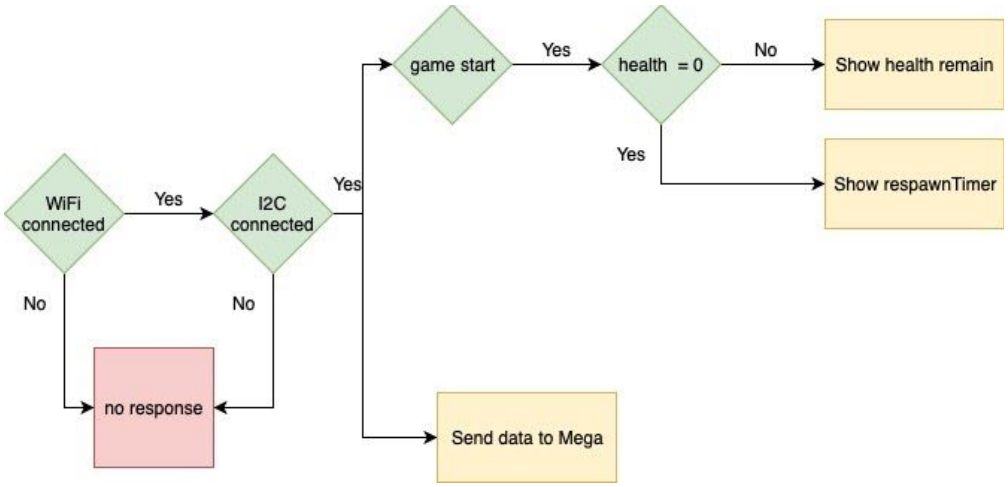


Figure 28. Coding Architecture of ESP 32 on car

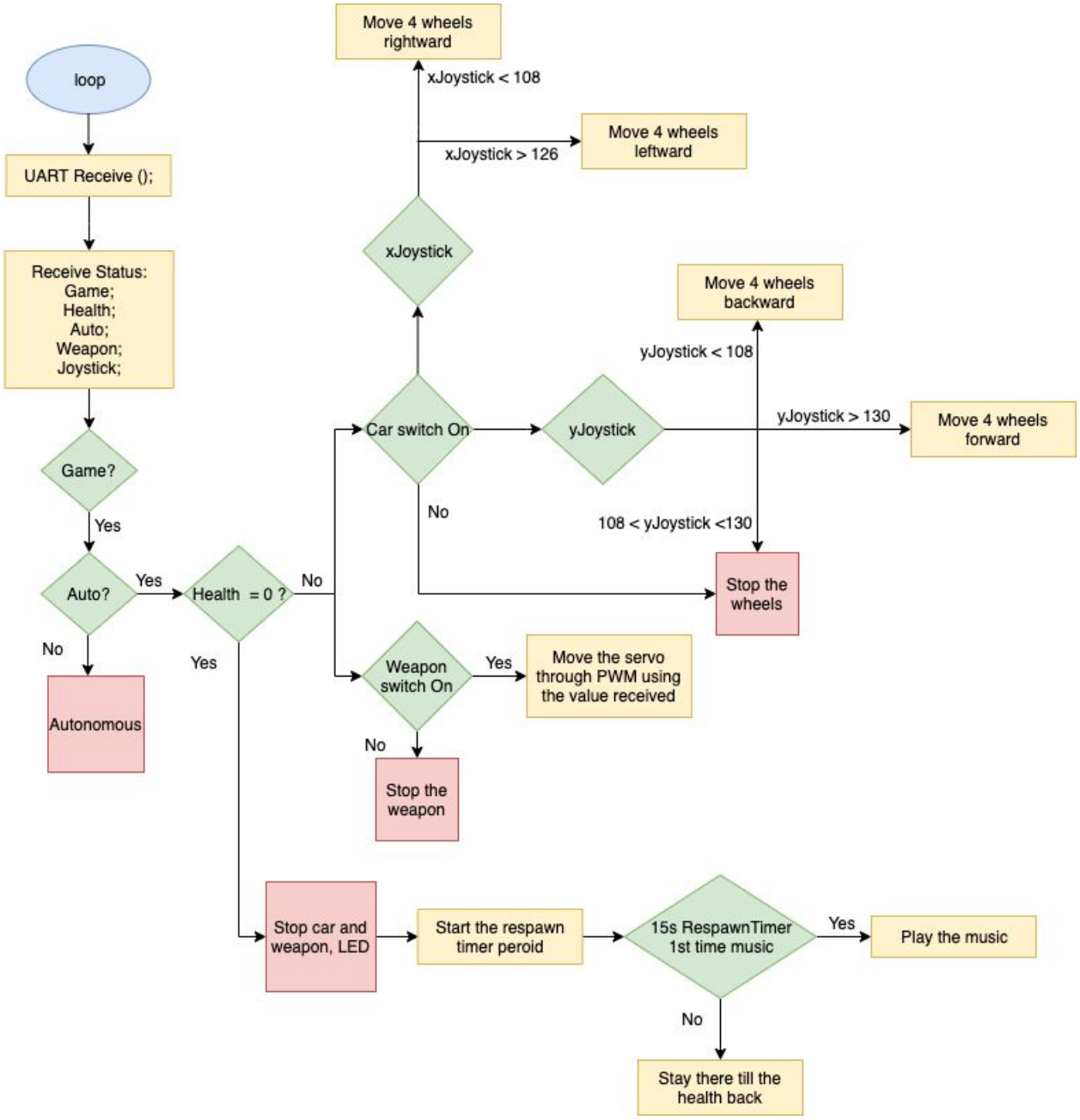


Figure 29. Coding Architecture of MEGA on car

5. Videos

Several videos clips, both in the competition and outside the competition, are included in this section to demonstrate the functionality of our robot. Each video provides youtube link and google drive link.

- **Performance in competition**

1. Selected video clip from livestream 1: https://youtu.be/reoxj4HhTFM (01:25:00 - 01:32:00)
2. Selected video clip from livestream 2: https://youtu.be/reoxj4HhTFM (03:35:00 - 03:38:00)
3. Selected video taken by ourselves during competition 1:
   https://drive.google.com/open?id=1NP_sMO0HqUX2e4gnwObc5d9LhCAMBNfR or
   https://youtu.be/1NjmHX3wr-I
4. Selected video taken by ourselves during competition 2:
   https://drive.google.com/open?id=10QxyUwbkhT9HDDD7BXqfK4b7KZiPuAIN or
   https://youtu.be/x-PNKb2QFLU

- **Functional Demonstration**

1. Physical Capability:
   https://drive.google.com/open?id=1Cm017_C3xycVyhLEk3KxB9pEiDsXCZ6V or
   https://youtu.be/pSqs84IXIxk
2. Virtual Attribute (top-hat & whisker):
   https://drive.google.com/open?id=1cJiJ9xE45Dec18TKQ3nf-NRvfjdZ7T-0 or
   https://youtu.be/6AmCsdlMeNY
3. Remote Control Mode:
   https://drive.google.com/open?id=18FNgjgCj6GFp7em0SF9v3HfGMG7qsfgD or
   https://youtu.be/if_8E6Xy7w8
4. VIVE Reading:
   https://drive.google.com/open?id=16NLWH5S5SUVshvRsoBR-c1JFV5eTL8l1 or
   https://youtu.be/HjIMckd3rrU
5. Autonomous Mode:
   https://drive.google.com/open?id=1BlDZhtQxFlfeppiFvRUzRxo9mCnw8RxH or
   https://youtu.be/j5N-poUflr0
6. Sound & Lighting:
   https://drive.google.com/open?id=11pOvOTfWW5dWBxAzyDH9l61Gpz3MtRsl or
   https://youtu.be/2Qqhy73MTFk
7. First time of car running by ESP 32
   https://drive.google.com/open?id=1wbA-7s7Bnv7_1aP_rQXa6n5Shfj7SDzn
8. First time of car running by Mega
   https://drive.google.com/open?id=1E491c5oGXSdYKQhoppe_HYxRXxnFxIjt

6. Retrospective

6.1 Retrospective from Jiaxiao Cai

I think the most important skills I learnt from this class are hands-on ability and knowledge about electrical and circuits. I have little background in electrical, and this course really benefits me a lot, from building simple LED at the beginning to complex multi-module robot car at the end. I think the best parts of the class are the quiz and recitation. The quiz in the lecture ensures us know the theoretical knowledge, while TA helps us to convert the knowledge into practical projects during recitation. I think I had the most trouble with the first two labs with Teensy. It is because it was my first time doing embedded system development. And I couldn't even figure out the logic behind set pin high or low. It is funny when I look back now and I am so glad that I went through these troubles eventually. I wish I had a clear mind when designing circuits and writing codes since it is really time-consuming when debugging. Last but not least, this class is warm. Although it was a tough time in GM lab, classmates were always helping each other debugging and supporting each other. I am so delighted to have them around. :)

6.2 Retrospective from Yifan Yuan

The most important skill I learnt from this class working on the embedded system and doing mechanical design by laser cutting. From the first thing I soldered was the teensy board, I finally solder the circuit board of our final project together with Miaoyan. The process of soldering give me lessons that don't be to hurry or you will always make mistakes. Just be calm and patient to do everything step by step. The best part of this class is that the previous labs taught us a lot and give a good prepare for the final project. And we are able to build our own car from nothing to everything. The most hard thing in this lab we have ever met is to debug the circuit. Since sometimes, you didn't do anything but the car just doesn't work. At this time, you will feel very annoyed and also unable to find the problem, especially when you are running out of time. Since we have almost rebuild the car after the check off day including the mechanical design and the electrical design to make the car smaller and more robust, We are running out of time but things should be running again from the start. Thus, we spent almost two to three days and nights in the lab to make our car work. It was a really hard time. The thing need to improve is that robustness of the components and earlier to prepare the vive and the field. And the TAs are really helpful and the class is really fun at all.

6.2 Retrospective from Miaoyan Sun

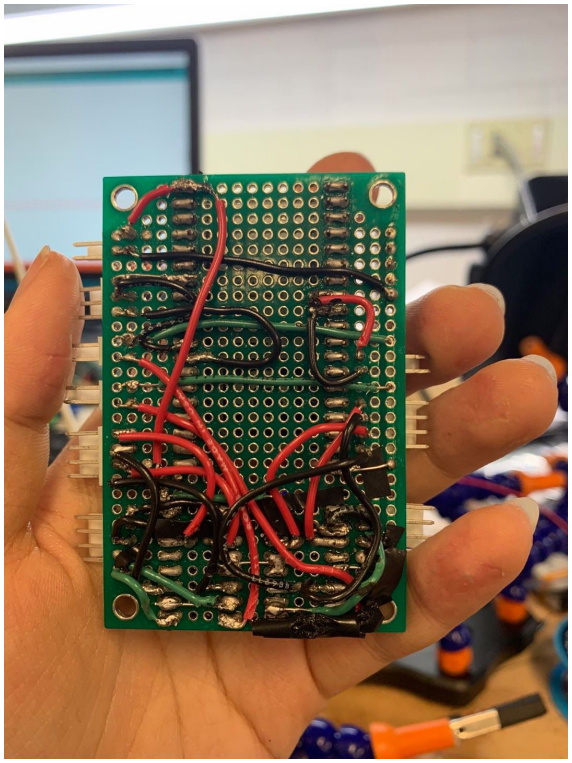
The most important thing that I learnt from this course is how to use microcontroller and electrical components. Before that, I had little knowledge about control. Though I have a background in mechanical engineering and I did take some control courses in my undergraduate years, I had very little practical experiences. This course is awesome in the way that it allows me to make something from scratch. A lot of my friends have warned me of this course regarding how time-consuming and tough it is. However, to be honest, I must say that I did not find it hard nor time-consuming at all. Is it because I picked up C++ before the semester started? May or may not. Indeed, I feel I love this course so that I think everything I did and every bug I fixed was meaningful. I learnt soldering and I found I love it. I used arrays to catch the x- and y- signals for VIVE, though not as straightforward as measuring their peak widths, it worked stably almost everywhere. Maybe the hardest part is sending numbers through Wifi using ESP. Next time I wish there will be better boards and electronic components for us to try. Moreover, it was the first time in my entire school years that I stayed up all night, but I feel energized when happily debugging the little robot car. Last but not the least, I am so grateful that I have two very awesome teammates, Yifan and Jiaxiao, who are far more intelligent and devoted than me. We always cheer each other up during hard times. Now we are good friends! Thanks MEAM 510!

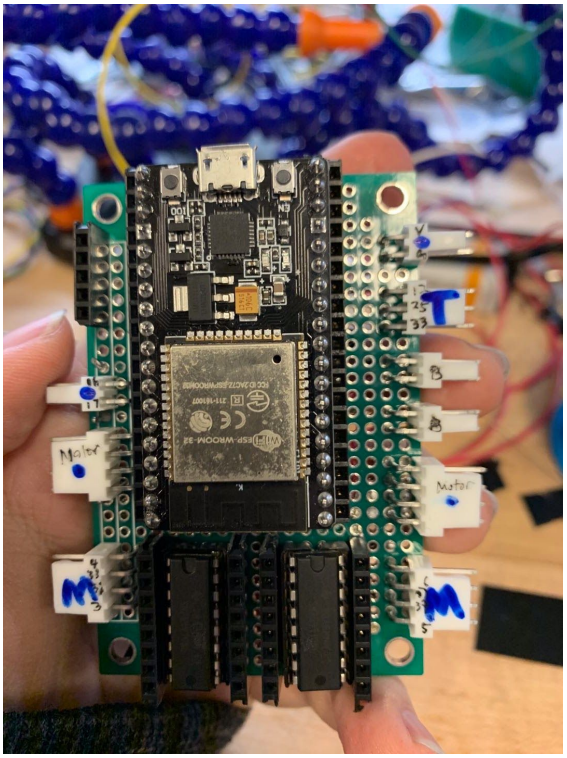
Appendix

Bill of Materials

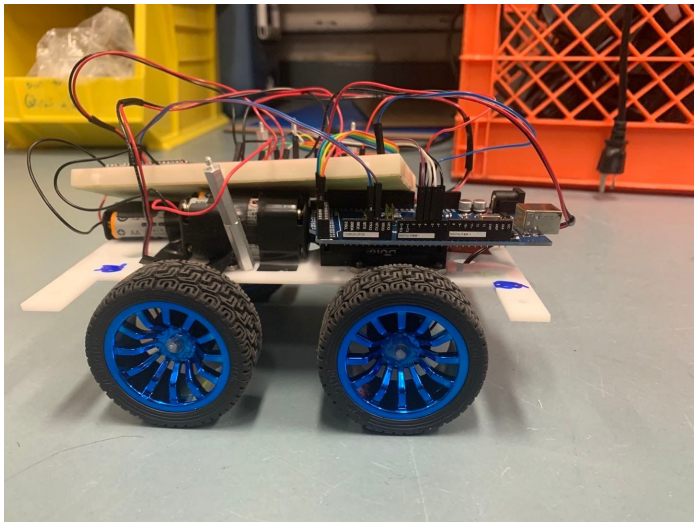
4 x wheel & motor set	$ 54.98
2 x Mega 2560 Board	$ 31.78
Ultrasonic Sensor	$ 9.26
4 x Continuous Servo (FT90R)	$ 15.98
100 x switch	$ 9.71
3 Color LEDs	$ 7.54
Batteries	$ 73.06
ESP 32	$ 26.74
Battery Connector	$ 3.18
Permanent Tap	$ 6.01
4 x SG90 Servo	$ 10.59
speaker	$ 2.31
Total:	$ 251.14

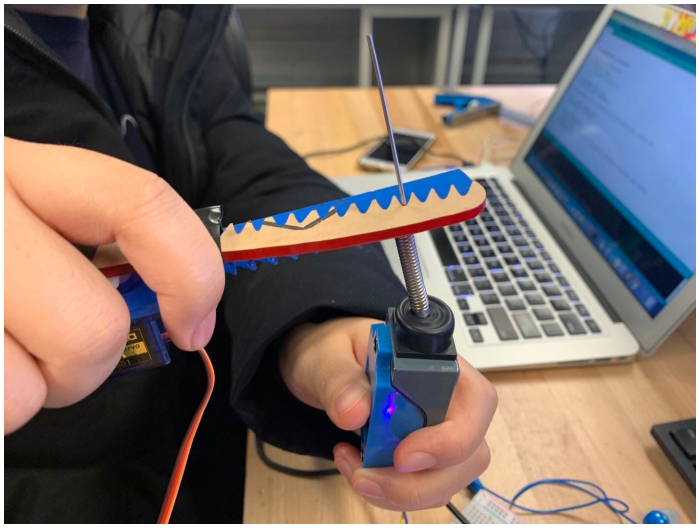
Physical electrical design photo

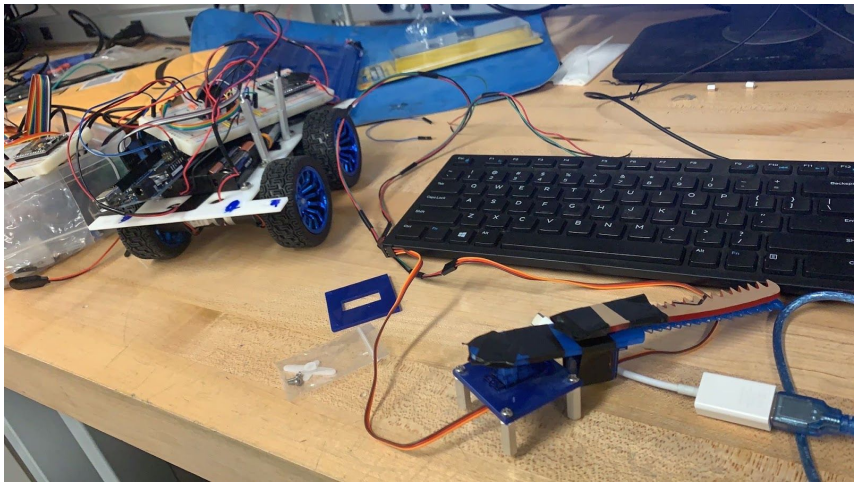


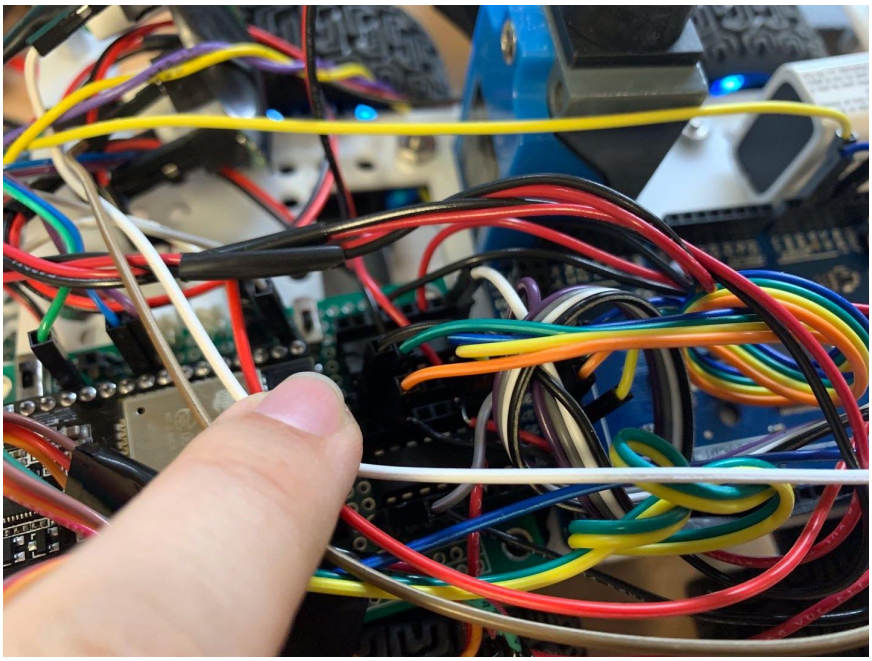


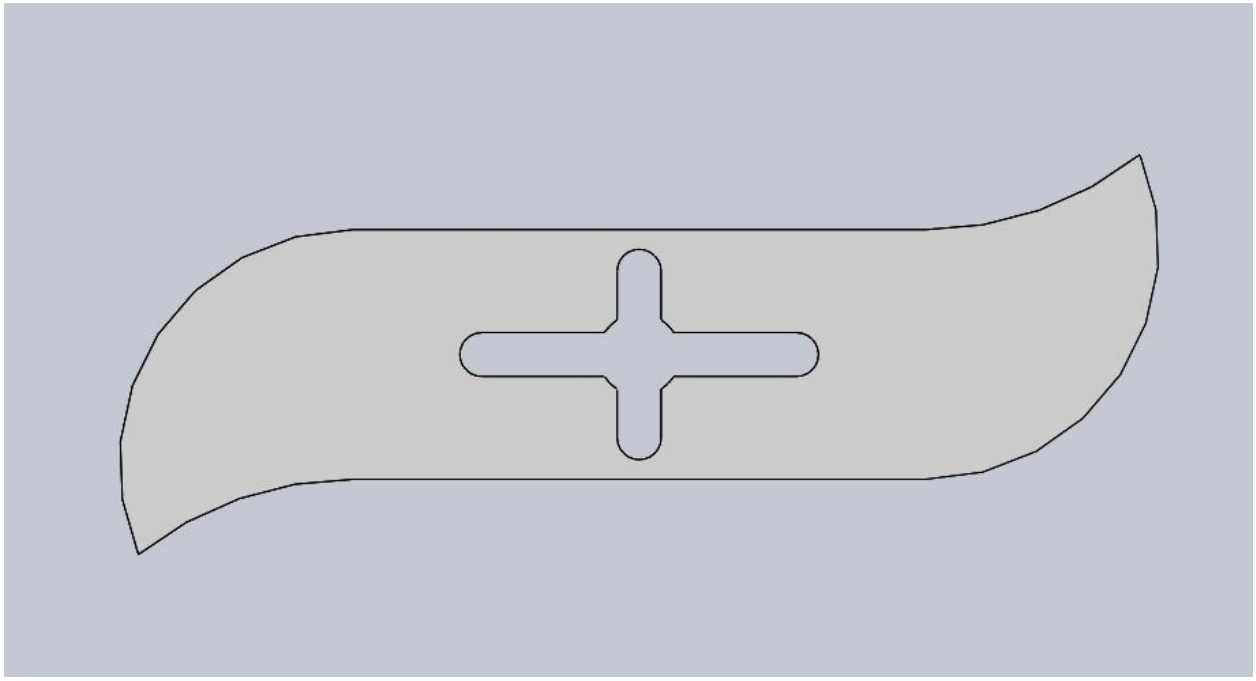
Try but failed car, weapons, and jumper wire circuit



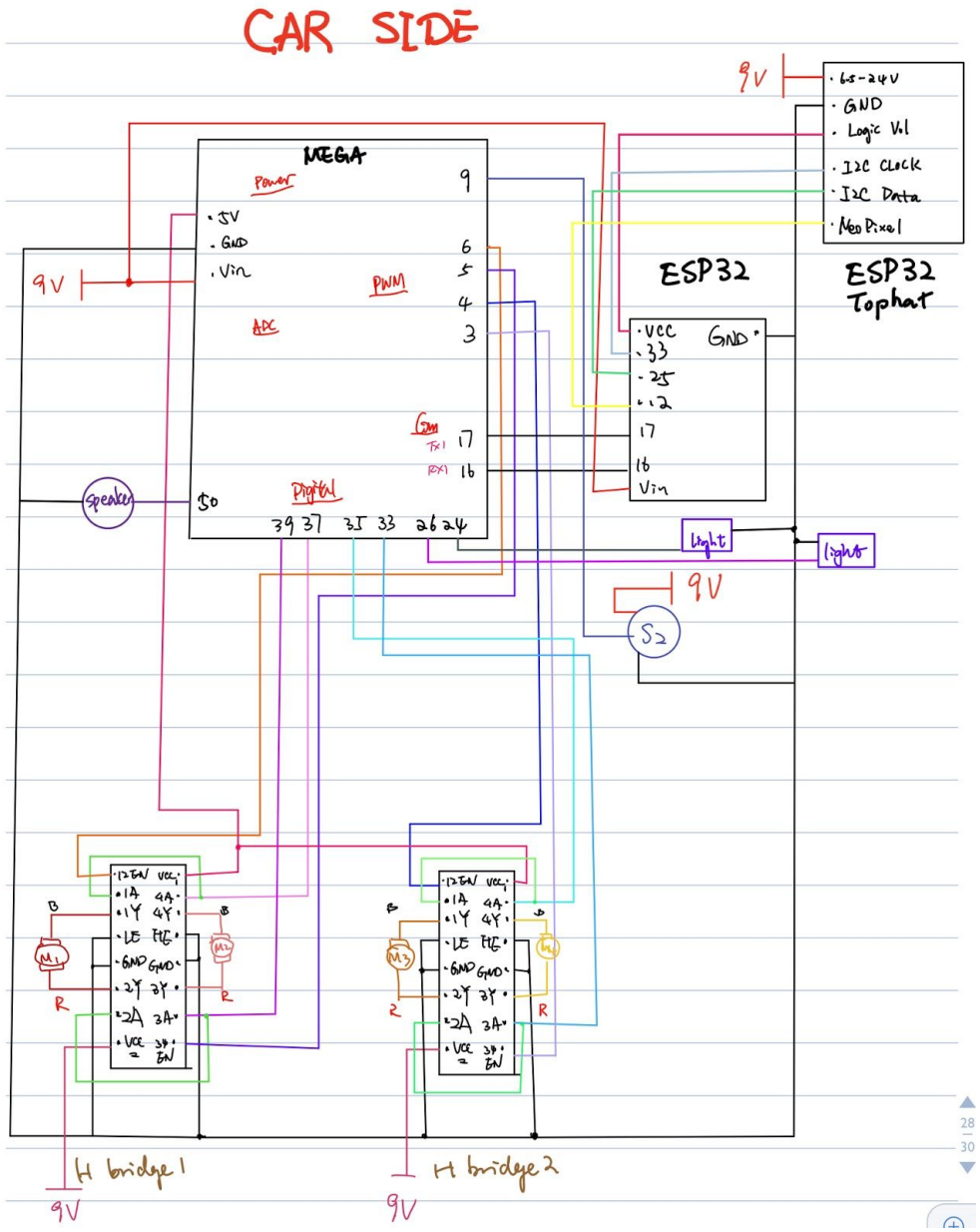




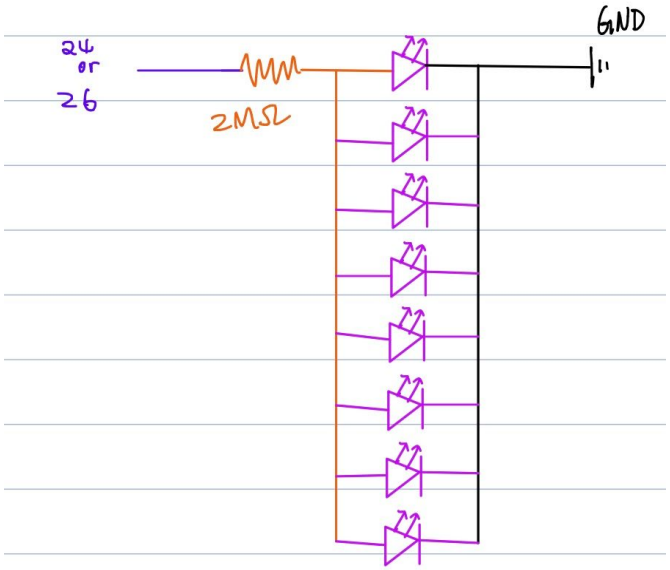




Schematic Diagram

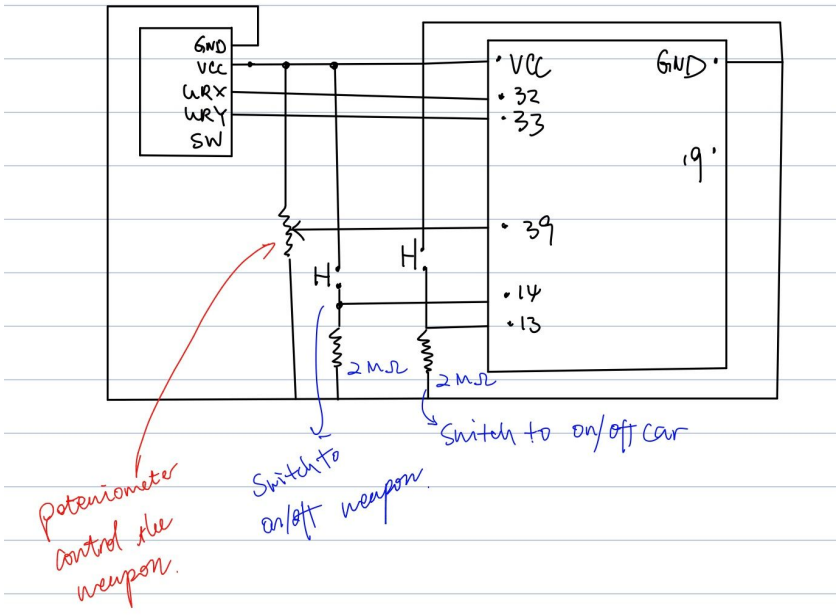


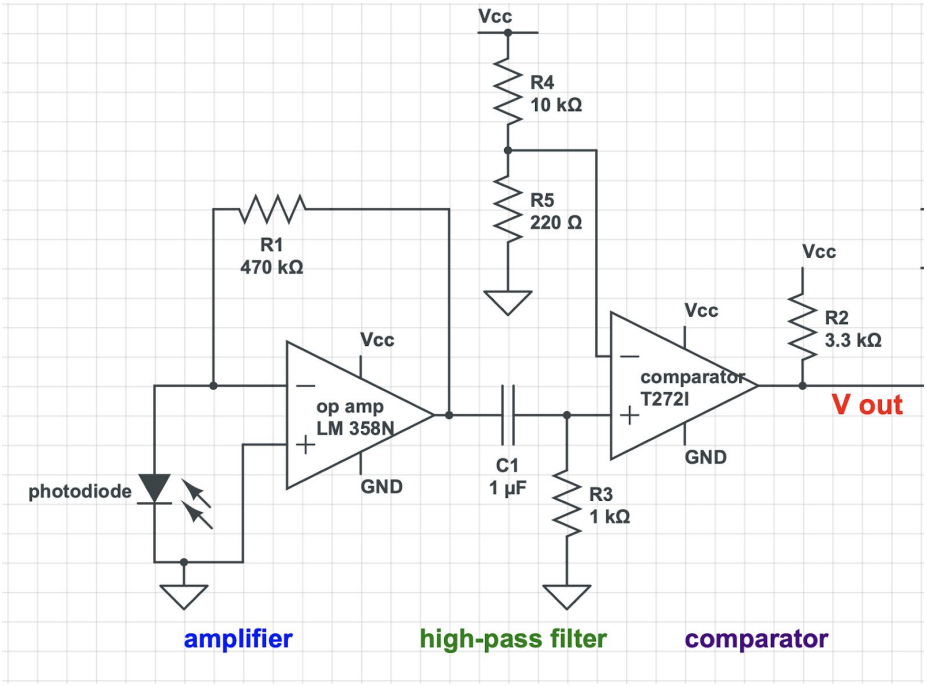
lights:



tri color LEDs.

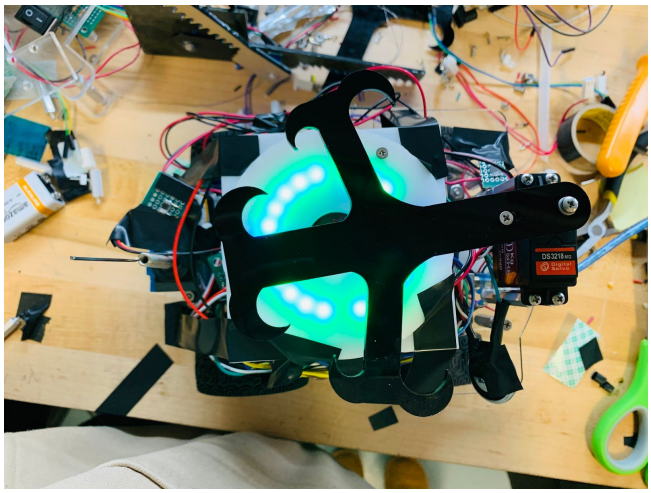
Remote Control Parts.

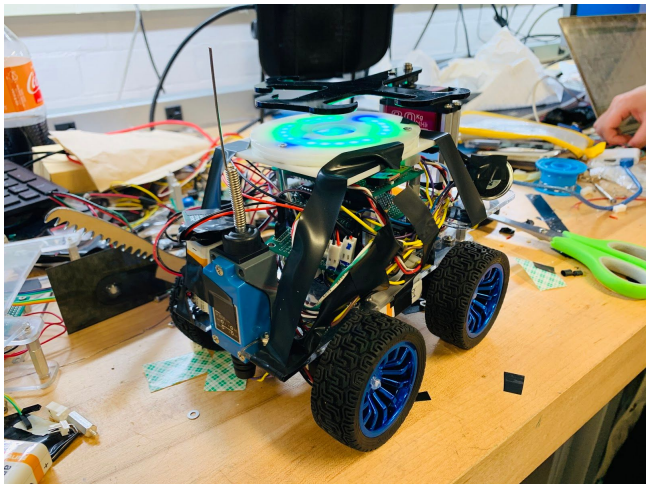


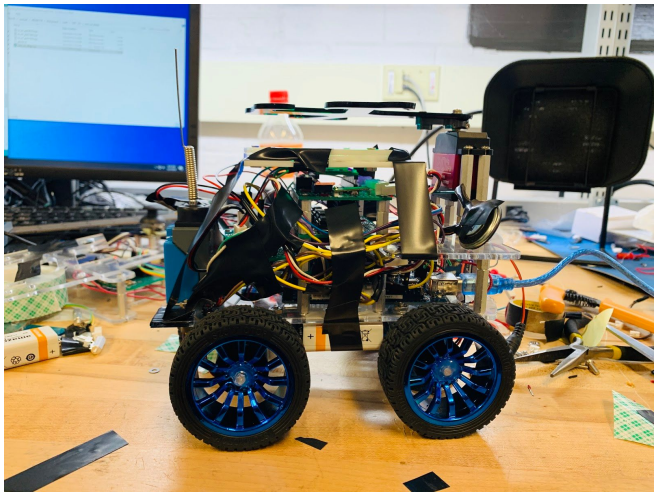


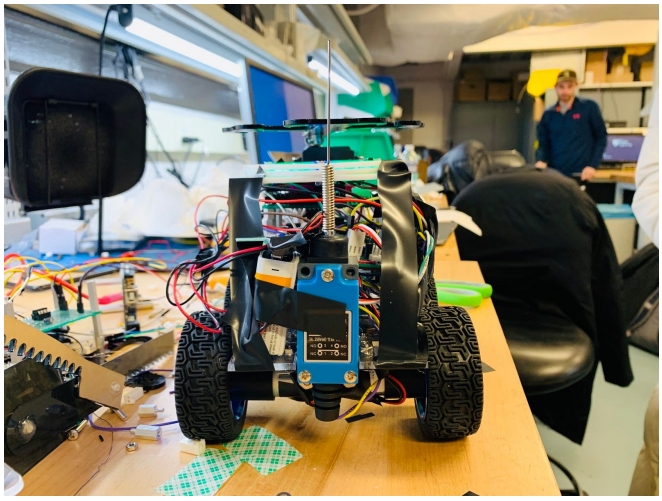
Circuit Diagram for VIVE

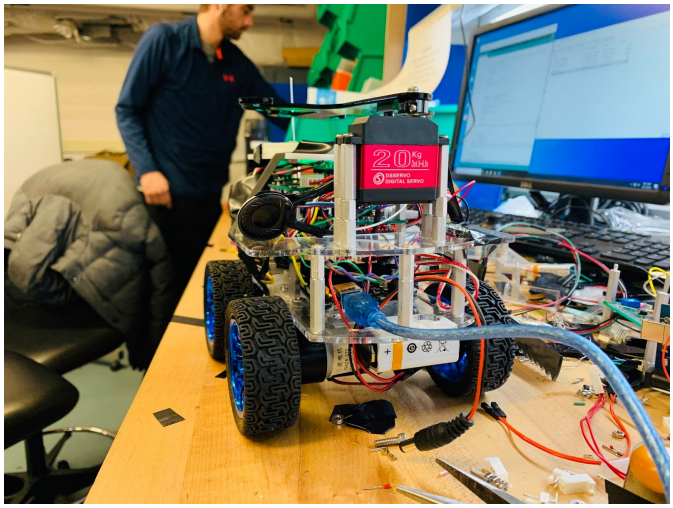
Photo of our car

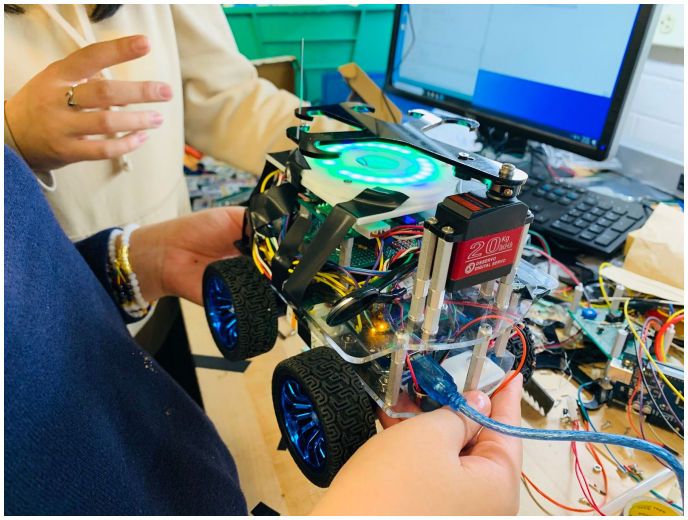




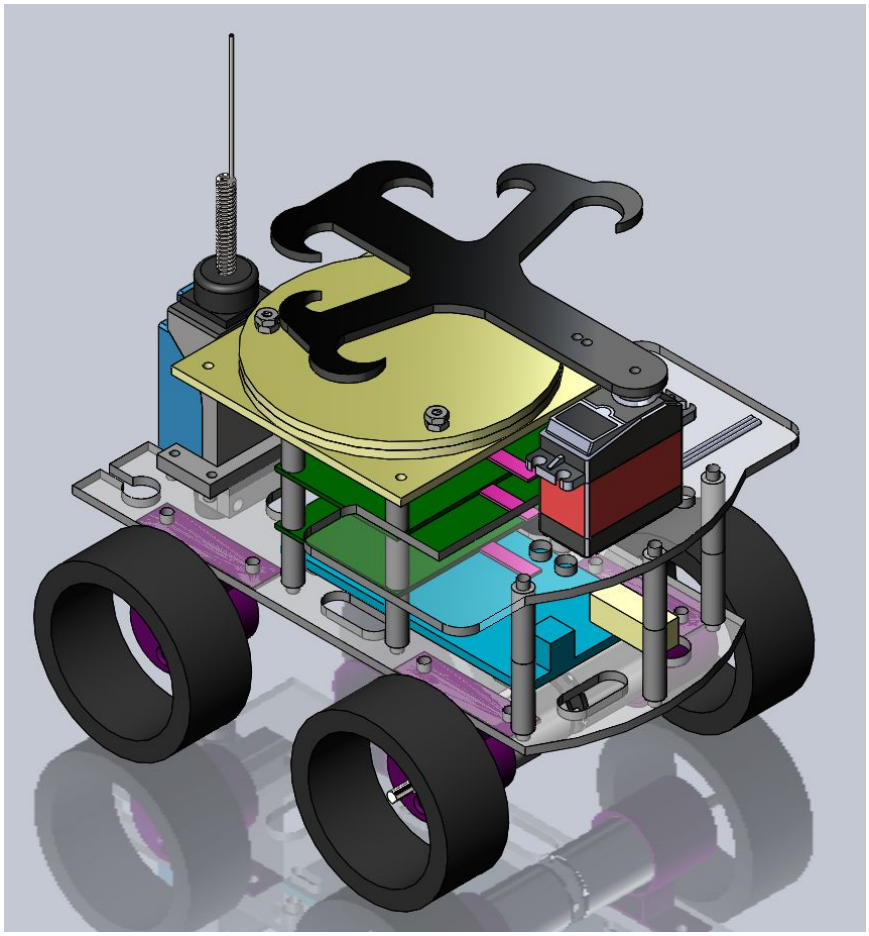


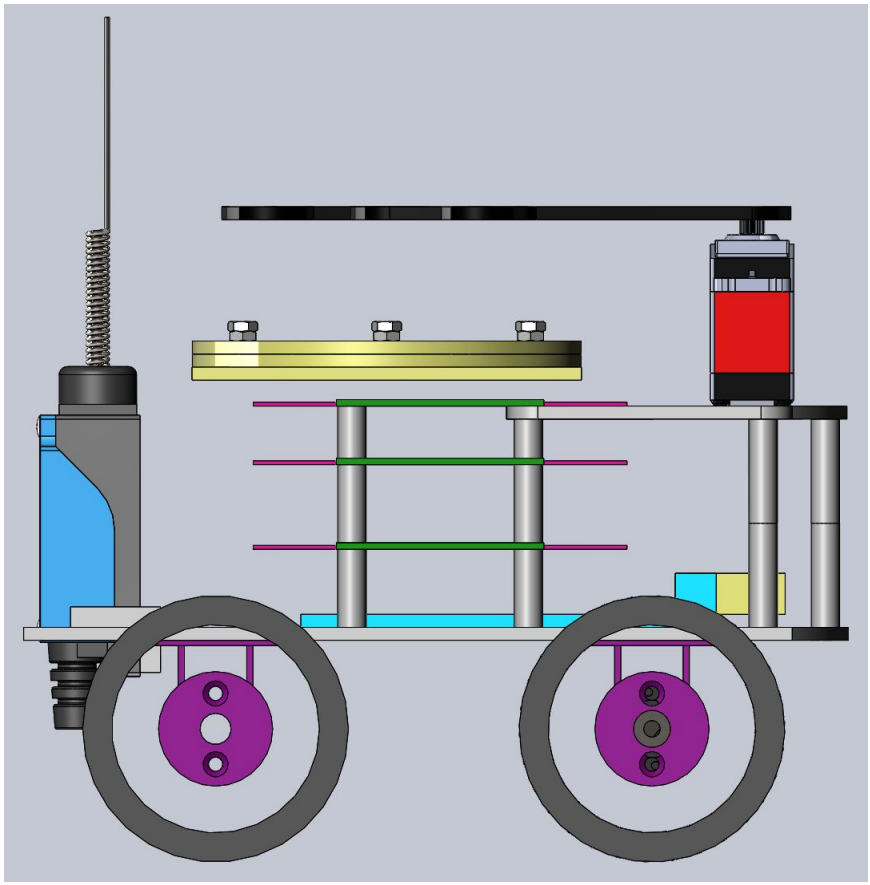


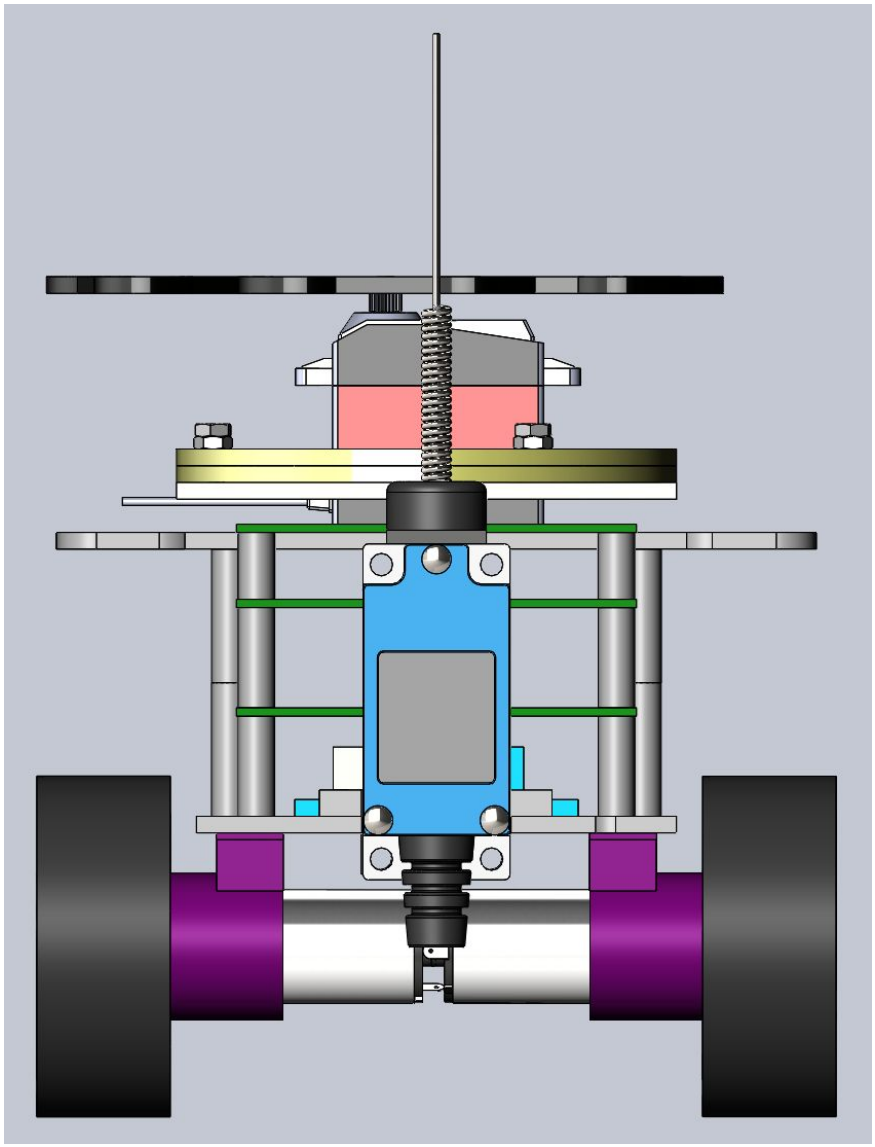

20Kg
DIGITAL SERVO



CAD of the car:

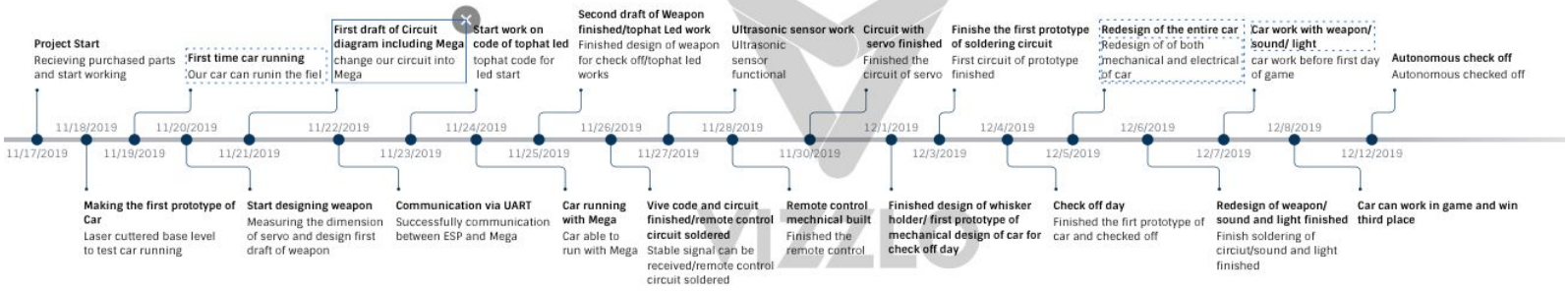






510 Final Project

Timeline



Data Sheet

DS 3218 Servo:

Features:
- High-precision metal gears with hard anodizing
- CNC aluminium middle Shell,
- water proof rubber seals screws

Sepcification:
Brand: DSSERVO
Item: DS3218MG
Stall Torque (5V): 19 kg/cm (263.8oz/in)
Stall Torque (6.8V): 21.5 kg/cm (298.5 oz/in)
Dead band: 3µs
Speed : 0.16 sec/60°(5V) / 0.14 sec/60°(6.8V)
Operating Voltage: 4.8 ~ 6.8 DC Volts
Weight: 60 g (2.12 oz)
Motor Type: DC Motor
Gear Type: Copper & Aluminum
Working frequence: 1520µs / 333hz
CE Certification: Yes
Size: 40 x 20 x 40.5 mm (1.58 x 0.79 x 1.60 in)

Package Included:
1 x DS3218MG Digital Servo
1 x 25T Adjustable metal servo arm

Chanzon 100 pcs 5mm RGB Multicolor Slow Flashing (Multi Color Changing) Dynamics LED Diode Lights (Blinking Round DC) Bright Lighting Bulb Lamps Electronics Components Filcker Light Emitting Diodes:

Specification:

- Quantity: 100 pieces / Pack
- Lens: 5mm Diameter / Transparent / Round
- Emitting Color: Red Green Blue Slow Flash Automatically
- Luminous Intensity: R:2000-3000mcd G:15000-18000mcd B:7000-8000mcd
- Viewing Angle: 30 Degree
- Forward Voltage / Current: 3V-3.2V | 20mA
- Polarity: Anode (Longer Part) | Cathode (Shorter Part)

DC motor and wheel:

Features:

100% brand new and high quality

This DC motor comes with mounting bracket and 65mm wheel This DC gear motor can be used in smart toy car, robot, printer. The motor is made of metal gear, wear-resisting, avoid tooth breaking, so with long serving life. This motor is adopted of pure copper wire coil, with low heat and low loss

Specifications:

Model: 25GA370

Rated power: 4 W

Product type: Brush DC motor

Rated voltage: 6V

Rated moment of force: 0.9 (NM)

Rated speed: 280 RPM

Motor size: as the picture

Wheel size: 65mm / 2.56in

Weight: 400g

Package included:

2 x motor

2 x bracket

2 x wheel

2 x wrench

2 x coupling

6 x black screw

1 set x bolt

DATASHEET of DC motor attached back

uxcell a15080600ux0275 Metal Shell Round Internal Magnet Speaker 2W 8 Ohm Pack of 4:

- Product Name : Magnetic Speaker;Power : 2W
- Resistance : 8ohm;Shape : Round
- Diameter : 28mm/ 1.1";Material : Metal, Plastic
- Weight : 29g
- Package Content : 4 x Magnet Speaker

ELEGOO 5PCS HC-SR04 Ultrasonic Module Distance Sensor for Arduino UNO MEGA2560 Nano Robot XBee ZigBee:
DATASHEET of Ultrasonic Sensors attached back

SG90 9g Micro Servos for RC Robot Helicopter Airplane Controls Car Boat:

Specification:

Model: SG90

Weight: 9g

Stall torque: 1.8kg/cm

Operating speed: 0.1sec/60degree

Operating voltage: 4.8V

Temperature range: 0℃ ~ +55℃

Dead band width: 10us

Maximum rotation angle: 180 degrees

Gear medium: nylon

Working mode: Simulatio

Circuit connection:

Yellow line (signal line)

Red line (power line)

Brown (ground)

DATASHEET of Servo attached back

DATASHEET of Mega attached back