Final Report

1. Functionality

- *describe minimal functionality approach*
- *describe extra components*

Our team's initial strategy was to capture the nexus in the lower part of the field by navigating to that area during autonomous. We believed this to be the most logical strategy since the spinning arms of death would be inactive during the autonomous period. As a result, we focused most of our design efforts on a strong autonomous program guided by ultrasonic sensors and the vive sensing circuit.

We decided to utilize three ultrasonic sensors as our extra components on our robot, one in the front and one on each side. When the robot encounters an obstacle directly in front, it can turn and rely on the side sensors to determine when the obstacle has been passed. We believe that this will ensure the robot does not waste any time by having to guess its distance to the end of the obstacle.

Since our robot hoped to capture the nexus, we wanted to focus on defense rather than an attack so that we could ensure that we could live long enough to capture it. Thus, we designed rotating arms on our flanges to use to push other robots away. We thought that this met the creativity criteria. Additionally, our controller creatively included laser cut bumper buttons utilizing limit switches.

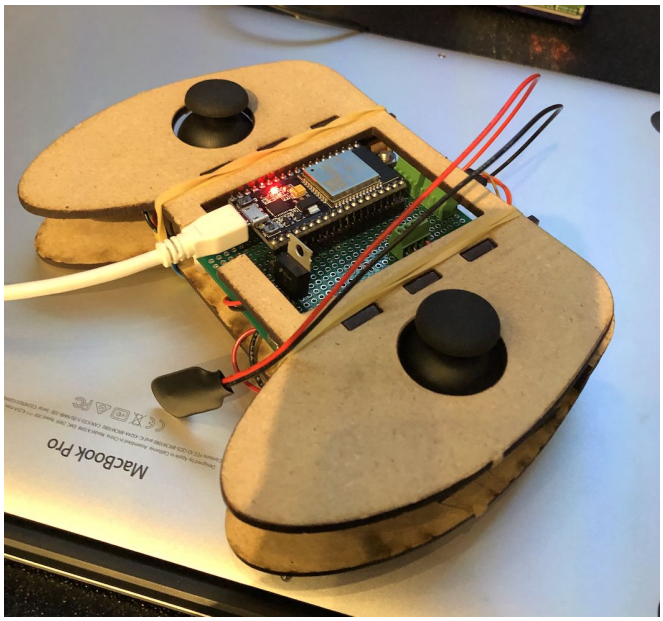

Figure 1. Final controller.

However, by demo day, our robot did not meet all of our goals. We were unfortunately not able to successfully implement an autonomous mode or a reliable defense mechanism. We instead acted as a defensive push bot and helped to gain our meta team some extra time during our last match.

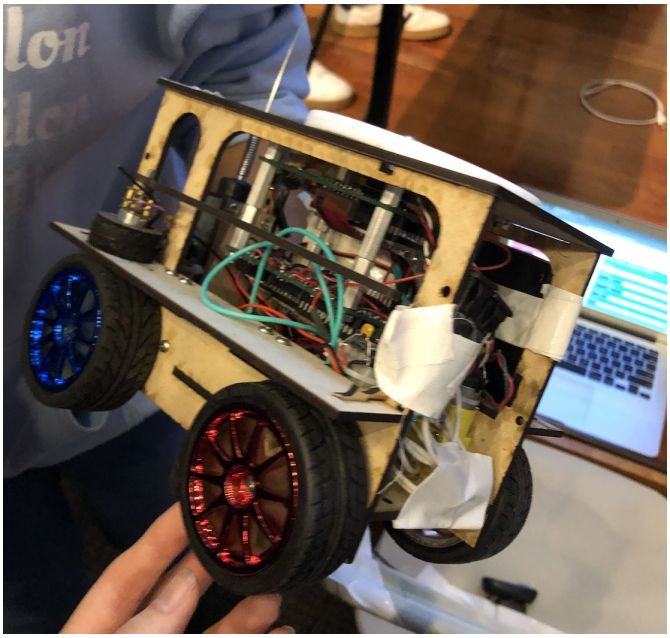

Figure 2. Final robot.

2. Mechanical Design

- *describe intended approach and actual performance*
- *include things that you tried but failed (and thus learned from)*

2.1 Drive Mechanism

We chose to use a four wheel drive system so that the robot would definitely be able to make it up the slope even with a significant amount of extra weight. The combination of the whisker switch, top hat, and additional boards added about 300 g to our weight requirements from the previous lab, bringing us to a mass of 836 g.

We chose to use the provided adafruit motors since they provided enough torque to maneuver around the field in the previous lab. Additionally we knew that we would be able to purchase additional motors from the TAs in the event that one of our motors malfunctioned unexpectedly.

The drive structure held up well for the most part during competition, though near the end of our match, the front right wheel fell off of its axle.

2.2 Defense/Offense Mechanism

The attack mechanism consists of a large arm mounted to a small DC motor. At first we planned to use servos, but their range of motion and overall force were too low for our application. The DC motors have a high gear ratio which means that they will have a great deal of torque and move slowly. This is ideal for our attack mechanism since it means that we will have enough force to activate the whisker switch while also being able to maintain control over the

slow-moving arm. However, since we are not using servos we need to consider the possibility that the arms might over-rotate and cause damage to the robot or themselves. To prevent this situation we added limit switches at the two ends of the motor's range. If the switch makes contact with either of these switches, the arm will be forced to rotate the opposite direction until there is no longer pressure on the switch. We included two of these arms at opposite ends of the robot so that we would have the greatest chance of dealing damage to our opponent.

This was later changed to a defense mechanism along the two flanges of our robot for defense. We reasoned that we should select from one of the two mechanisms above because if we are using arms to push other robots away, we bring our own robot out of range to attack. We also were limited in the number of pins remaining on our microcontroller to allow such actuation.

We never really got to try out our defense mechanism because during competition, our arm control was not working properly. Complications aside, if we were to redesign this mechanism, we would make the mount more sturdy because had our arm been used, we think the force from a robot on the arm onto the motor would have sheared the super glue connections within the mount, disabling our defense.

2.3 Weight

The car was designed to be very light overall with a final weight around 1kg. We wanted to be able to make it up the ramp with no issues and minimize the amount of power needed to drive around the field. We used ⅛'' MDF for all of the vehicle's surfaces, which was more than strong enough when properly fastened.

Our robot structure was designed to optimize space while keeping the center of mass of our robot low. We tried to keep our center of gravity as low to the ground as possible. In our race car, we were able to achieve a stable distribution of weight by placing the batteries on the underside of the vehicle. We repeated this on our new robot and secured two large LiPo batteries to bottom of the chassis. We also mounting our motors low and centralized our perforated boards and circuitry. The top of the car was also kept as light as possible with fewer fasteners and more cutouts in the MDF. During testing the car did not lift when performing turns or driving up the slope. The robot's final dimensions were 6" x 7" x 6".

Although by demo day our robot was definitely not the lightest robot out there, we still successfully kept it relatively light.

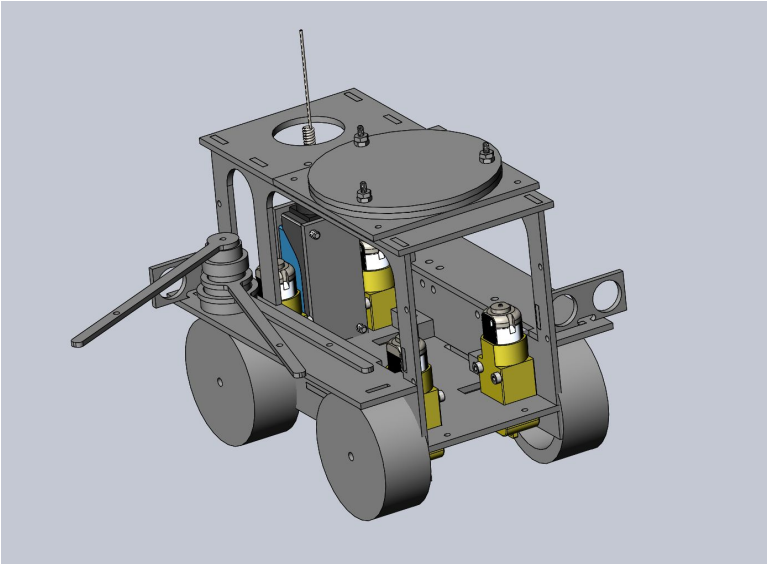

Figure 3. Final CAD of robot: isometric view highlighting unique defense mechanism and ultrasonic sensor mounts.

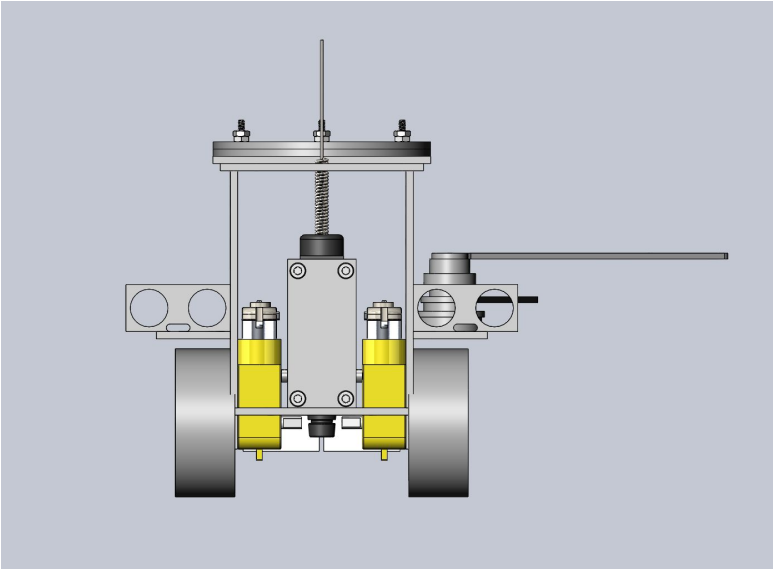

Figure 4. Final CAD of robot: view of back.

2.4 Controller

Our controller is comprised of two switches and two joysticks that can each only move in the vertical direction. We originally chose to control each side of the car with one joystick so that we would have better control over the direction of the car. Ideally, if one motor is slightly slower than the other we can compensate by moving the joysticks different amounts. This design worked well for our team at first with the joysticks controlling both the speed and direction of each wheel. However, as the motors began to develop lag from the Wifi we decided to set the wheel speed to a constant 50% once it was clear the joystick had been intentionally moved. This made the car slower overall but it became much easier to control turns without

overshooting. The push buttons were included to control the attack mechanisms. Pushing one moves the mechanisms clockwise while pushing the other moves them counterclockwise.

3. Electrical Design

- *describe intended approach and actual performance*
- *include things that you tried but failed (and thus learned from)*

When designing our electrical components we tried to use as few microprocessors as possible. This choice allowed us to both save space and prevent the communication lines between boards from becoming crowded. We planned to control our robot with a total of three ESP 32 microprocessors: the main control board, a dedicated drive train board, and the top hat/ whisker switch board provided by the TAs. The circuit was designed this way specifically for autonomous mode. We could use the vive photo sensing circuit to determine our coordinates and ultrasonic sensors to stop the robot when there is an obstacle. By placing both of these components on the main board we could calculate our next move without waiting for data from additional microprocessors. Then we could send commands directly to the drivetrain circuit through I2C. This layout also would have ensured that we only needed to communicate in one direction between any two boards.

However we had to reduce the circuit to a single ESP controlling all of our components just a few days before the competition. At that point the I2C between the drive board and main board still had several issues to be resolved, but there wasn't enough time to finish. Three days before the competition, our drive board inexplicably broke down and we were forced to redirect our efforts towards the driving functionality. With very little time left we decided to simplify to one ESP to recover time it would have taken to finish the I2C. Unfortunately this meant that we couldn't include several components due to the smaller number of pins, but the final robot's driving ability was noticeably faster to react than tests with our three board setup.

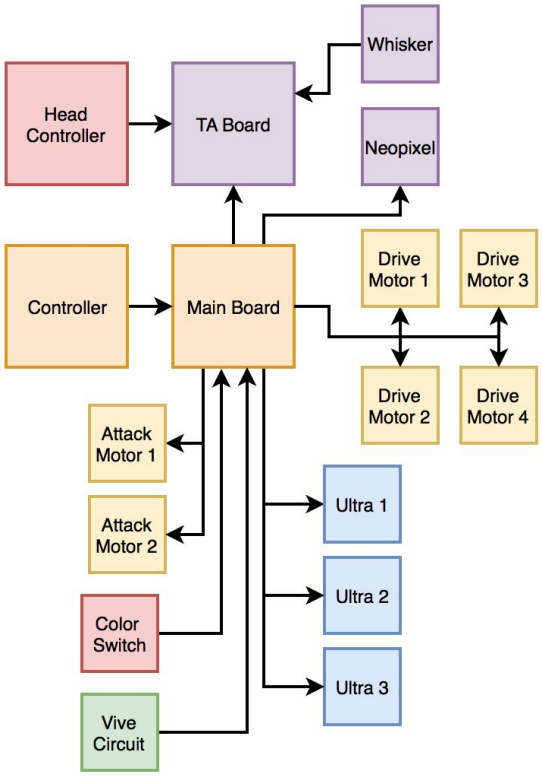

Figure 5. CPU Architecture for our robot.

3.1 Main Control Board:

The main board is responsible for controlling all of our robot's actions. It powers the ultrasonic sensors and vive circuit for autonomous as well as the servos for the remote control period. The board also contains a switch for determining team color so that we don't need to reflash the code before every match. Most importantly the board provides PWM and direction signals for the robot's four Adafruit TT motors. The board is controlled by one ESP 32 microprocessor which uses every available pin that does not cause problems with the I2C or Wifi.

The drive train requires two PWM signals and four analog signals for direction. Since the wheels on any given side will never need to oppose each other, we use the same PWM and direction signals for each pair of motors. In our original designs we planned to use an inverter so that the ESP would only need to provide two direction signals (one for each side). However, the inverter caused the motors to run uncontrollably when the system lost connection or experienced a reboot. By providing the signals directly from the ESP we can ensure that this issue will not occur.

There are many components connected to the board we don't need to worry about the microprocessor handling too many functions since only half of the components will function during autonomous (ultrasonic and vive) and half will operate during autonomous (servo). In practice the motors did have a small degree of lag from the wifi (0.5-1s) but very little of that could be contributed to the number of components being controlled simultaneously.

3.2 Vive Circuitry

The Vive circuit connects to the ESP on the main control board, but the components are soldered onto a separate perfboard. During tests where the components were squeezed onto the main board we had a great deal of difficulty reading a clean signal. With so many components on one board, signals from the drivetrain and other heavy current lines were interfering with the signals generated by the Vive sweeps.

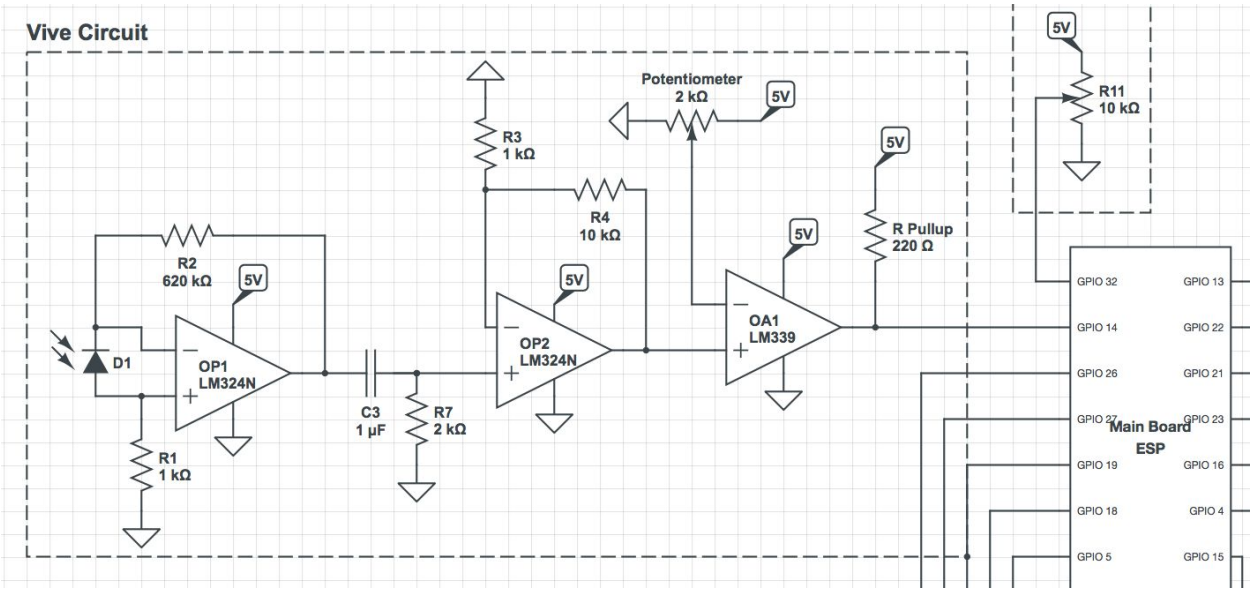

Figure 6. Vive circuit schematic.

The board requires a dual op-amp and comparator to read the signal clearly. The signal first goes through a special kind of op-amp circuit called a current to voltage converter. Then we use a high-pass filter to remove any signals below 50Hz. Most indoor lights operate at a frequency of about 40Hz, so this step will ideally remove the ambient noise from visible light. Then the signal passes through a non-inverting op amp with a gain of 10. We chose to increase the signal after filtering the noise since our gain is low enough that there won't be too much noise created by the op-amp itself. Testing showed that this order produces the cleanest signal for our circuit. Finally the signal is increased to 5V with a comparator. With the Vive station in the lab we are able to obtain clear coordinates from up to 10 feet away.

3.3 Power Supplies

The robot requires multiple batteries to power the microprocessors, motors, and peripherals. The drive motors require the largest amount of power since they draw almost 1A from the battery. We chose to power them with their own LiPo battery, since LiPos have a capacity of 5000 mAh. This is also necessary to prevent the current draw on the battery from affecting any other sensitive components. Most of the other components don't draw anywhere near as much current as the drivetrain, so the TA/tophat board, servo motors, and peripherals are all powered by a second LiPo battery. Finally the main microprocessor requires a 5-6V input which we supply through an AA battery pack. When wiring the batteries together we made sure to avoid ground loops by always connecting lines directly to the battery. This setup worked well in competition and we did not observe any issues arising from our setup.

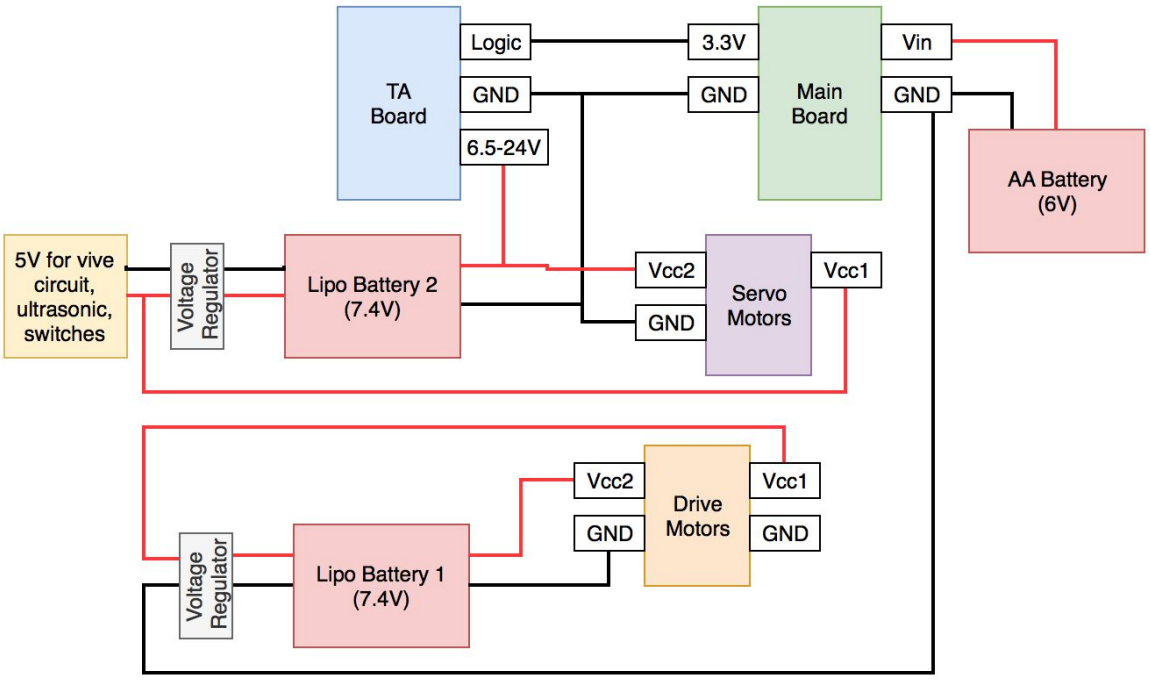

Figure 7. Battery connections on the robot.

3.4 Remote Control

Our controller used a single ESP connected to the robot through Wifi. It reads values from the two joysticks and two push buttons with digital read. The entire controller is powered by a 9V battery since the components receive power from the ESP's 3.3V line.

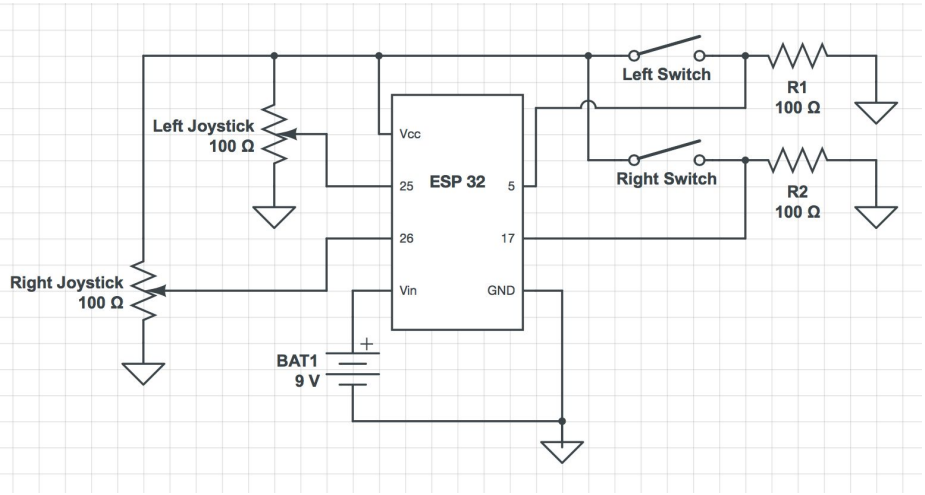

Figure 8. Remote control schematic.

4. Processor Architecture and Code Architecture

- *include a block diagram of how MCU's are logically connected*
- *describe software approach*
- *include things that you tried but failed (and thus learned from)*

4.1 Drivetrain and Remote Control

The robot receives drive commands from our remote controller. The remote controller has two joysticks that each use analog read to determine their voltage. The left joystick maps its values from 2-127 and the right joystick maps its values from 128-254. The left and right buttons will continuously send 1 and 255 respectively when pressed. As a result we never have to send numbers that are over 1 byte in size. This helps to reduce our lag and makes it easy to determine which control the command is coming from.

4.2 Vive

In order to obtain coordinates from the vive circuit we needed a way to distinguish the X and Y pulses from the sync sweeps. We accomplished this by always storing the pulse widths of the last three pulses. We know that X,Y pulses will have a width around 4-12 microseconds and that sweep pulses have a width around 120 microseconds. So we can easily determine if a pulse is X or Y if its width is less than 30 microseconds. However we still need to know which of these are specifically X and which are Y. By storing the last three pulses we can determine that the signal is an X pulse if we have just read three sync pulses in a row. Otherwise the signal must be a Y pulse. Finally, we can convert this into a coordinate value by comparing the time between the X,Y pulse and the preceding sync pulse to the time between two sync pulses. To ensure that we are getting the most accurate coordinates, the function returns the average of five X,Y pulses each.

4.3 Autonomous

Our plans for autonomous were not fulfilled by the end of the competition, as our team focused more on producing a reliably driving robot before taking on the task of creating a fully functioning autonomous protocol. Our early ideas and attempts to implement autonomous included the use of both the Vive station and a series of ultrasonic sensors.

Since the robot could begin in any position and orientation on the field, we begin our autonomous section by determining both of these values. We use the vive coordinate function to measure our starting position. Then we move the robot approximately 6 inches forward and measure again. By comparing these two numbers we calculate our direction about the vive station.

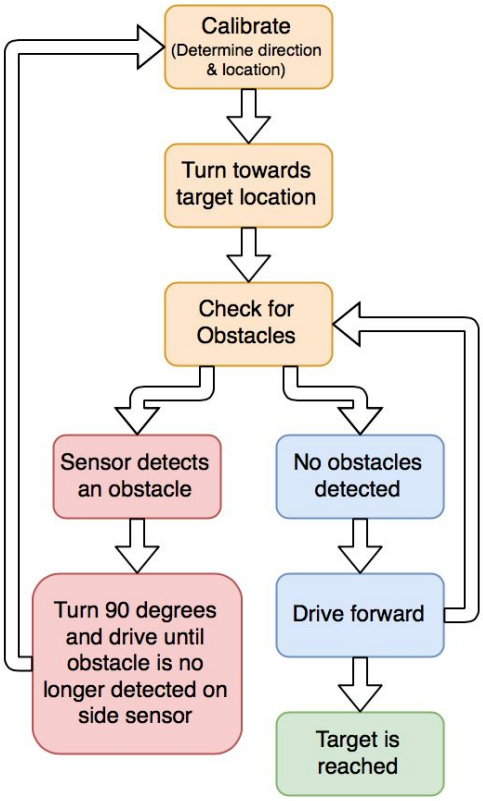

Figure 9. Logic structure governing our autonomous program.

Then the robot drives towards the lower nexus, marked by a hardcoded (X, Y) position. The bot uses its ultrasonic sensors to navigate the obstacles it encounters, taking the following steps to reach its destination: knowing its orientation, the robot will turn towards its target, and drive forward until it detects a nearby obstacle within 5 cm of its rangefinders. If it detects an object in front of either forward range finders, it will turn in place in the direction opposite the activated side until it senses a clear path for at least 6 inches. It will then move forward for 6 inches and once again reorient itself towards its goal, ready to drive towards its target again.

4.4 Communication

The robot uses I2C lines to communicate between the TA board and the main control board.In our setup, the main board acts as the master while the TA board is defined as a slave. The TA board both reads and writes to the master board. When the TA board acquires new information (either from the whisker switch or the head control tower), it writes to the master board that it needs to communicate. The main board will then interrupt its current process in order to read the new data from the TA board.

5. Videos

https://drive.google.com/file/d/1BXGs2miQxdB9AUqnrFPG4WRIwFcXSQge/view?usp=sharing

6. Retrospective

- *What you feel was most important that you learned*
- *What was the best parts of the class*
- *What you had the most trouble with*
- *What you wish was improved*
- *Anything else about class*
- *Each member's thoughts on the gameplay*
- *Each member's thoughts on what you learned*
- *Each member's thoughts on what could be better*

Toni-Ann

This final lab was a good amalgamation of the topics covered over the year. From troubleshooting a second Vive sensing circuit to setting up wireless communications between multiple microcontrollers, the final lab allowed me to understand the concepts taught in previous labs more deeply. I was able to better understand how hardware filters worked, as we spent hours trying to properly implement a high pass filter to remove ambient noise from our IR sensor. I learned cleaner, more efficient methods of soldering circuits, making use of as many sockets as possible to allow for interchangeable components. At the same time, I learned entirely new concepts associated with the integration of our various systems into one robot. Things like controlling the I2C lines of microcontrollers and using different types of connectors and switches to connect lines from different boards were relatively new, though derived from the topics of previous labs.

Out of all the experiences of this class, my favorite engagements were centered around the design and creation of circuit boards for the bot. I had little previous experience with actually soldering boards, especially as a mechanical engineer. This lab helped to push me out of my comfort zone in mechanical design and rapid prototyping, and challenged me to learn to build and troubleshoot increasingly complex circuits.

Though we began work on the robot early, our main issues stemmed not from developing individual systems, but from the integration of them. Though we seemed to be making good progress on the project in the weeks before Thanksgiving, we did not truly begin to connect the different boards into the same system until the final week, leading to problems in control and unseen errors to rear their head when we tried to integrate our code. As we combined each of our independent systems, connecting boards to each other and attempting to implement I2C communications between each board, we redesigned our main controller board, eventually to the point of phasing our I2C lines altogether. Had we attempted integration earlier, we would have had the insight to attempt these redesigns earlier as well, making the development of a finished robot much easier near the end of the project.

Overall, this lab, and this class as a whole, was an enlightening and enjoyable experience. Though I spent many hours working in GM lab, trying to push out the best work that I could and

and trying to understand the concepts at work in each situation, I feel that these labs were worth the effort.

Zoey Flynn

I think that our robot could have been improved had we decided to control all of our components with one ESP earlier in the project. Connecting everything to one board meant that we didn't have to worry about getting two ESPs to communicate via I2C. Additionally I think we could have improved our design by designing a robot with horizontal platforms that could be easily removed layer by layer. Otherwise I think the robot was designed extremely well in almost all other areas. Most of our problems during the competition arose from the fact that the drive board completely malfunctioned several days before the competition forcing us to rebuild the piece in a very short time.

Prior to this course I had no experience with electronics or mechatronics. This class struck a perfect balance between hands on learning and traditional lectures. I really enjoyed most of the labs and can genuinely say that I now have a solid understanding of the subject. I think that the most important thing I learned is how to best diagnose and deal with electronics issues. At the beginning of the class I found it very difficult to debug the hardware on my own. However, through the trial and error of each lab, I have developed a strong sense of how to use the electronic equipment to evaluate the circuit and locate the cause of a problem.

Throughout this course I had the most trouble with the Vive circuit. I found it very difficult to debug the circuit since everyone had to share one station. There was often no room to work near the station due to overcrowding so some of the signals were very small and nearly impossible to read. I also wish that the Teensy software was available on the school computers. I lost the ability to use either of my computer ports early on due to power draw issues with my circuits. Had I been able to use a school computer I would have been able to work on teensy code more easily without having to ask my peers to upload for me. I also wish that we had learned a little more about how to actually use sensors and to implement them in our designs.

Shaylin Marn

I feel like the most important thing I learned from Mechatronics was circuitry. Before this class, things like wires, resistors, and capacitors were all wizardry to me. It was nice to finally get to apply some of the things I learned in my intro physics courses. I knew nothing about mechatronics before coming to this class, and I struggled a lot with most of the labs. However, I often got a helpful push from my good friends and classmates (thanks, Zoey Flynn and Martin Yang) and when I finally got my things to work, those were perhaps the best moments of the class. I was probably the most proud of my waldo because when it finally came together, even though it was a little wonky, it did what I envisioned it would do and was laid out with the design I had in mind. On the other hand, I think I had the most trouble with the ESP32 in general. I feel like I was pretty confident with my abilities with the teensy, but as soon as we started using ESP32, I lost all momentum. Perhaps it's just wifi that messes me up the most. Either way, I was not very good at figuring out my bug in that lab.

If there is one thing I could change about the class, I wish there was more time for the final project. I feel like with many people gone over Thanksgiving break, we really did not have all that much time to produce a good robot, especially since we ran into complications that took days to fix or weren't fixed altogether. As a result, we had to stay in very late on many nights leading up to the deadline. Good integration of all the systems itself takes a while. Maybe in the future, there could be checkpoints, and if your team successfully gets that task done in that time, you are given full credit. If you were mostly there or showed a lot of effort, you are given a pass. If you did not show much effort or progress, you are given half credit. At some point, there is example code and set up that is revealed so that students can move forward with this to avoid falling behind from some type of issue. Alternatively, there could be code released in the last week with errors in it, and you could use to get that part of your robot working.

Overall, mechatronics was a very positive experience for me. I feel like I learned a lot and am excited to apply this knowledge in the future. This knowledge is especially powerful. Only some people know how to program and wire real world devices. Now I can say that I'm (sort of) one of them.

Mohamed Mohamed Shahul Hameed

The important elements which I learnt in this lab was interfacing with microcontrollers and circuits, especially the Vive and the Waldo. Integrating all them into a big project helped me to learn about team work, planning, time management and patience.

The best parts in this class was the Race Car which was the first time in the lab where we combined multiple modules to design a working product with an output.

Building up the Vive was a difficult task as I couldn't achieve a proper output at the end of the lab.

I feel that working with ESP32 for the whole course can be more interesting and we can build in bigger projects using it as interfacing with Teensy had problems, particularly the Serial port which wasn't working time to time for half of the people.

This course was one of the interesting courses I've took, and it was a fun-filled one with practical applications. The course was designed perfectly to teach everything about mechatronics right from scratch and is a very relevant course for people, especially for beginners who are aspiring to work in the field of robotics.

7. Appendix

- *Bill of Materials*
- *Schematics of all electronic circuits*

- *Photo or rendering of full robot*
- *CAD drawings highlighting anything special*
- *All data sheets for all components (either links or copies of data sheets)*
- *Upload all code to canvas (separately)*
- *Optional if you have fun competition video links please add*

Bill of materials

	Component	Manufacturer	Part Number	Quantity
Large Electronics	Adafruit TT motor	Adafruit	377	4
	Ultrasonic Sensors	Sparkfun	HC-SR04	3
	Neopixel LEDs	Adafruit	1507	1
	Micro DC Motor	Pololu	3050	2
	Whisker Switch		ME-8169	1
Small Electronics	Op Amp	Texas Instruments	LM 324	1
	Comparator	Texas Instruments	LM 339	1
	Photodiode	Texas Instruments	GT CSHPM1.13	4
	Level Shifter	Texas Instruments	BOB-12009	4
	H Bridge	Texas Instruments	SN-7544	3
	Voltage Regulator	Texas Instruments	LM 7805	6
Batteries	Lipo			2
	9 Volt			1
Mechanical	Wheels	DFRobot		4
	Hex Shafts	DFRobot		4
	M2.5 Screws	McMaster Carr		25
Materials	MDF			5

Table 1. Bill of materials.

Data Sheets

TT Motor	https://www.adafruit.com/product/3777
Ultrasonic Sensors	https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
Neopixel	https://cdn-shop.adafruit.com/datasheets/WS2812.pdf https://cdn-shop.adafruit.com/product-files/1138/SK6812+LED+datasheet+.pdf
Micro Motor	https://www.pololu.com/file/0J1487/pololu-micro-metal-gearmotors-rev-4-1.pdf
Op Amp	http://www.ti.com/lit/ds/symlink/lm324-n.pdf
Comparator	http://www.ti.com/lit/ds/symlink/lm139.pdf
Photodiode	Datasheet
Level Shifter	https://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf https://cdn.sparkfun.com/datasheets/BreakoutBoards/BSS138.pdf
H Bridge	http://www.ti.com/lit/ds/symlink/sn754410.pdf
Voltage Regulator	https://www.sparkfun.com/datasheets/Components/LM7805.pdf

Table 2. Datasheets for electrical components.

Schematics

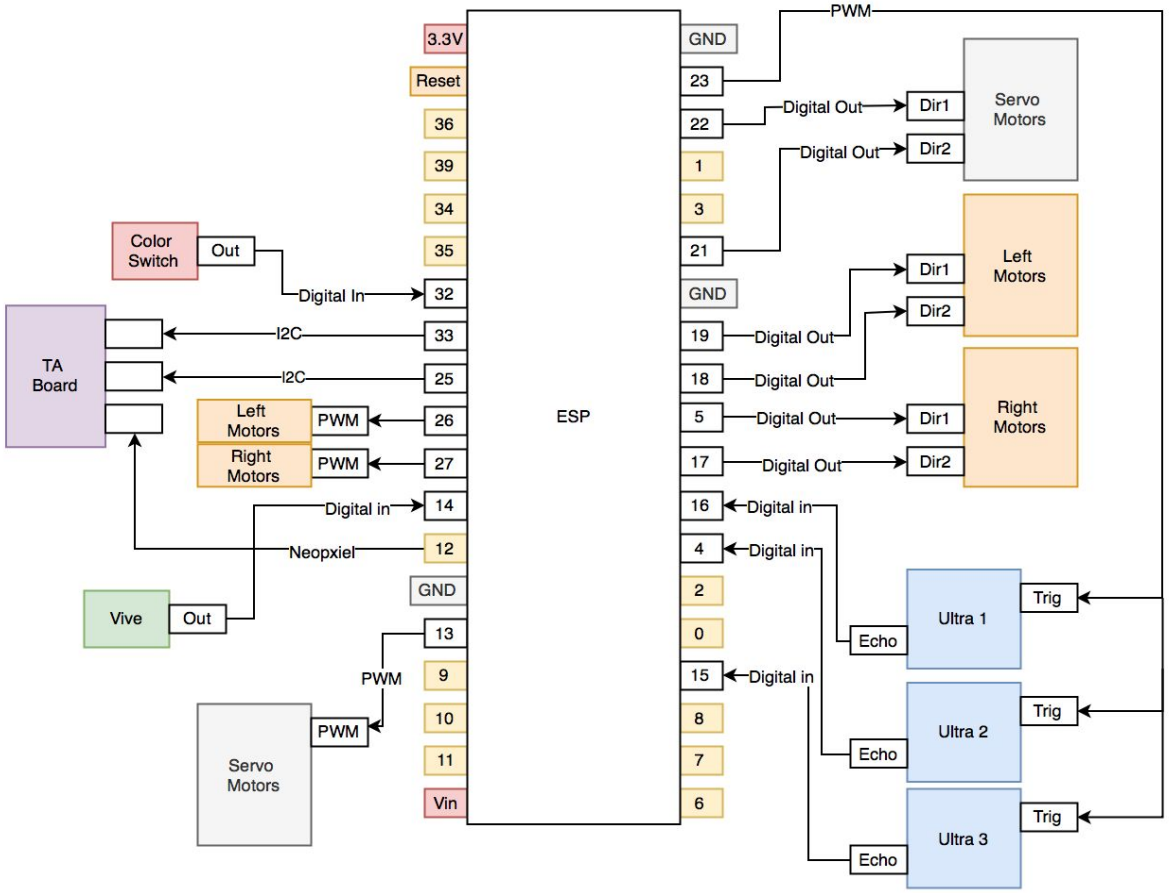

Figure 10. Main board pinout.

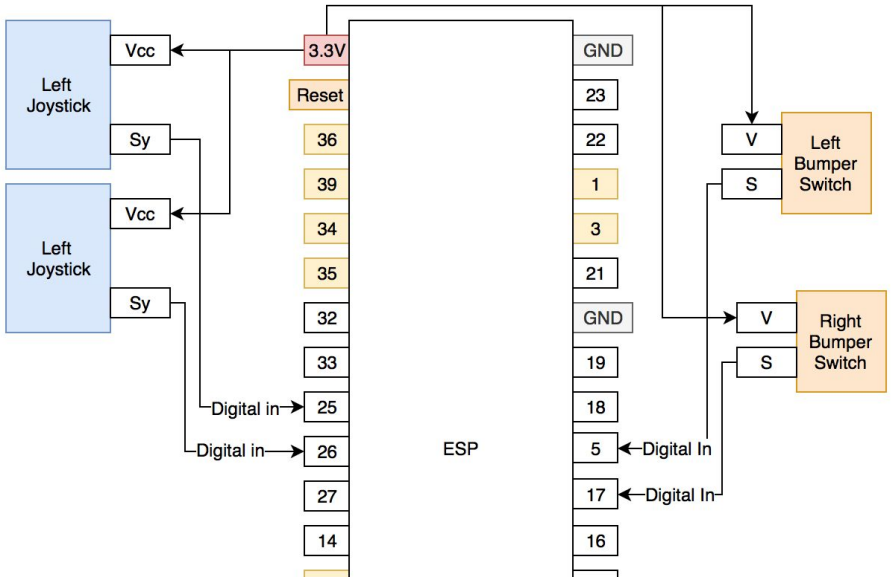

Figure 11. Controller pinout.

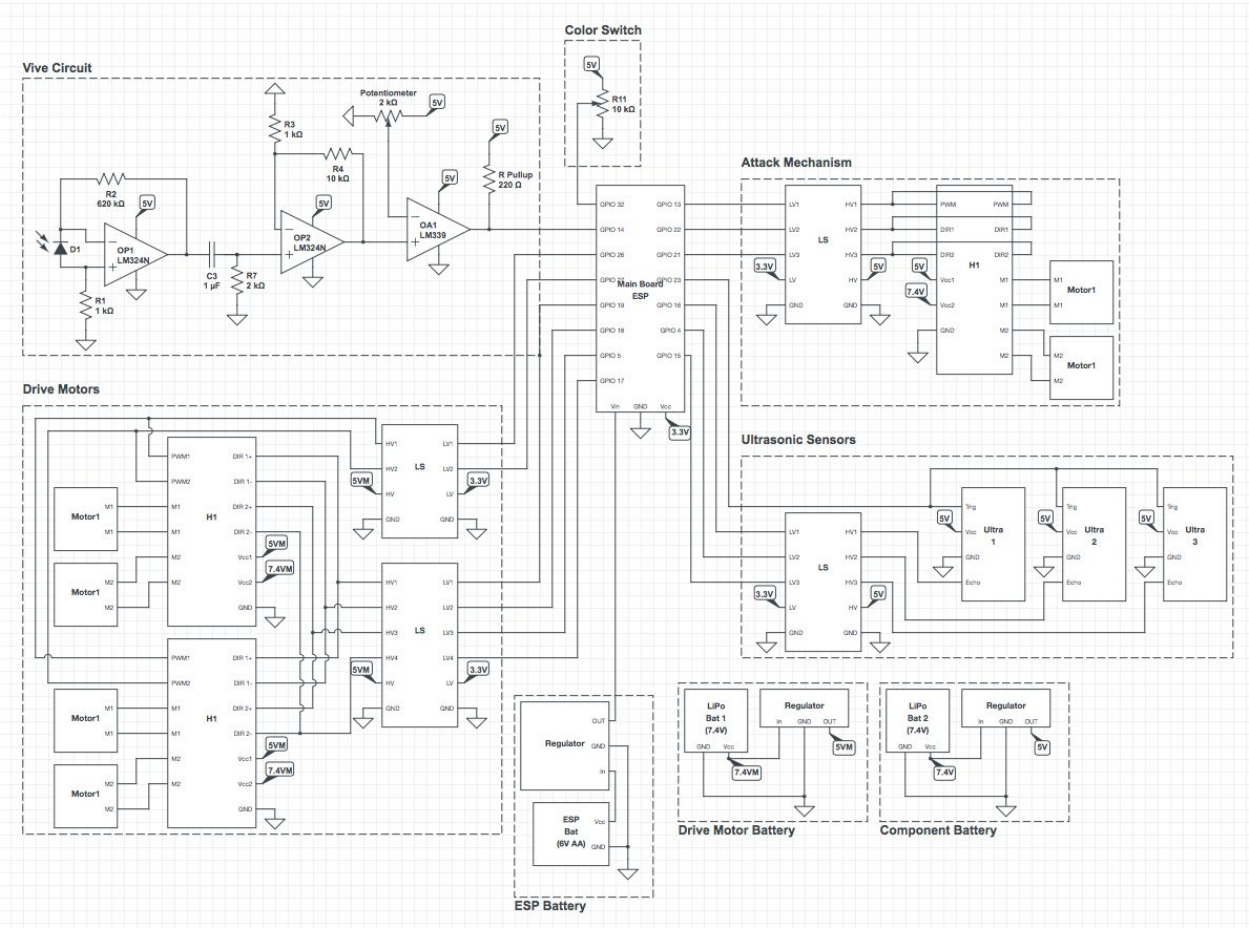

Figure 12. Full schematic for main drive board.

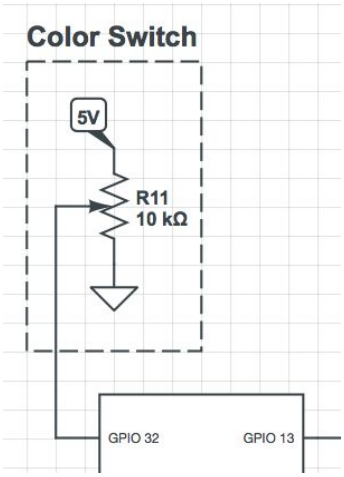

Figure 13. Color switch schematic.

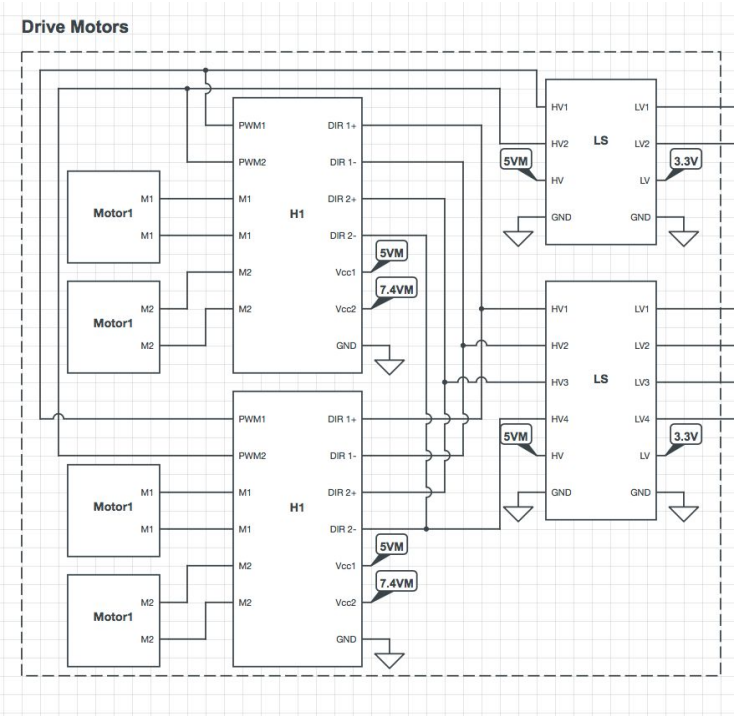

Figure 14. Drive train schematic.

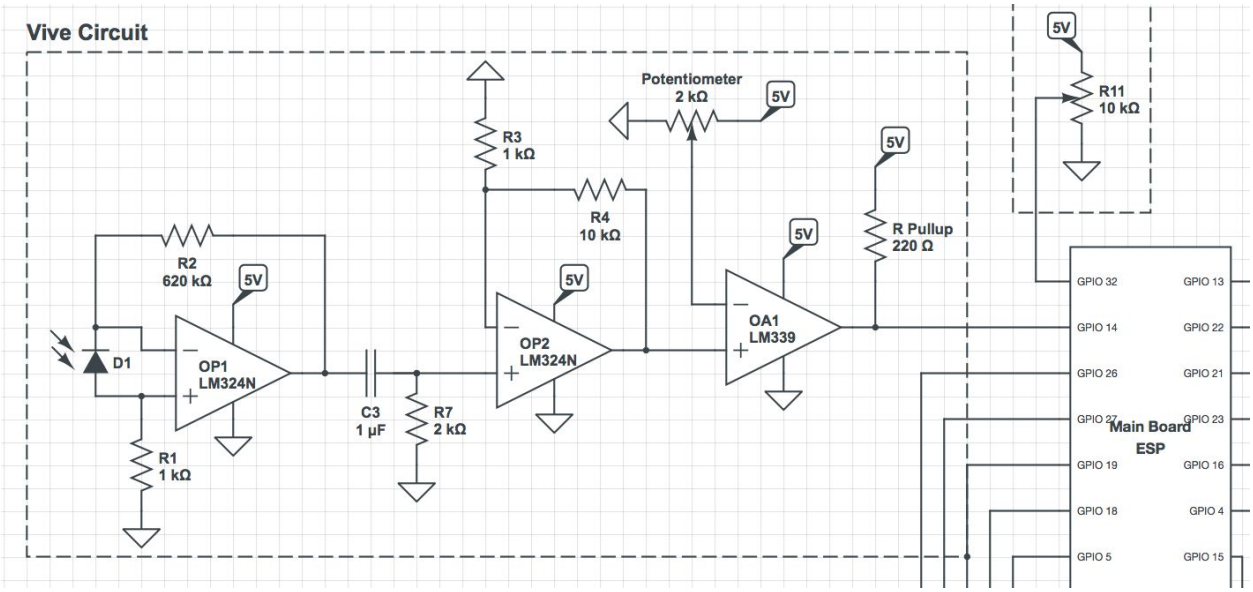

Figure 15. Vive circuit schematic.

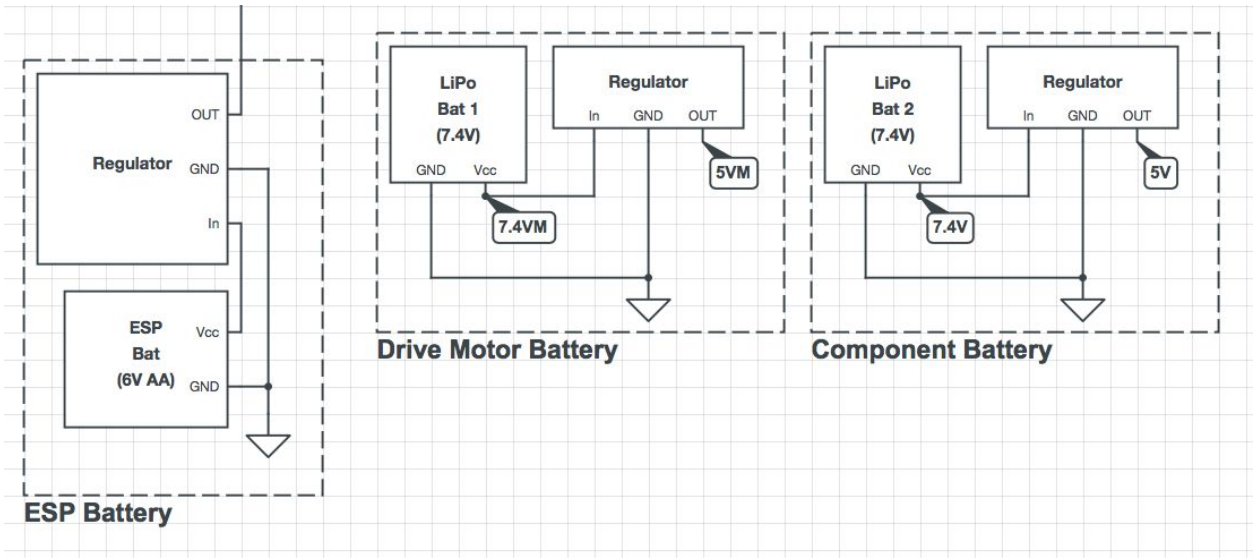

Figure 16. Battery schematic.

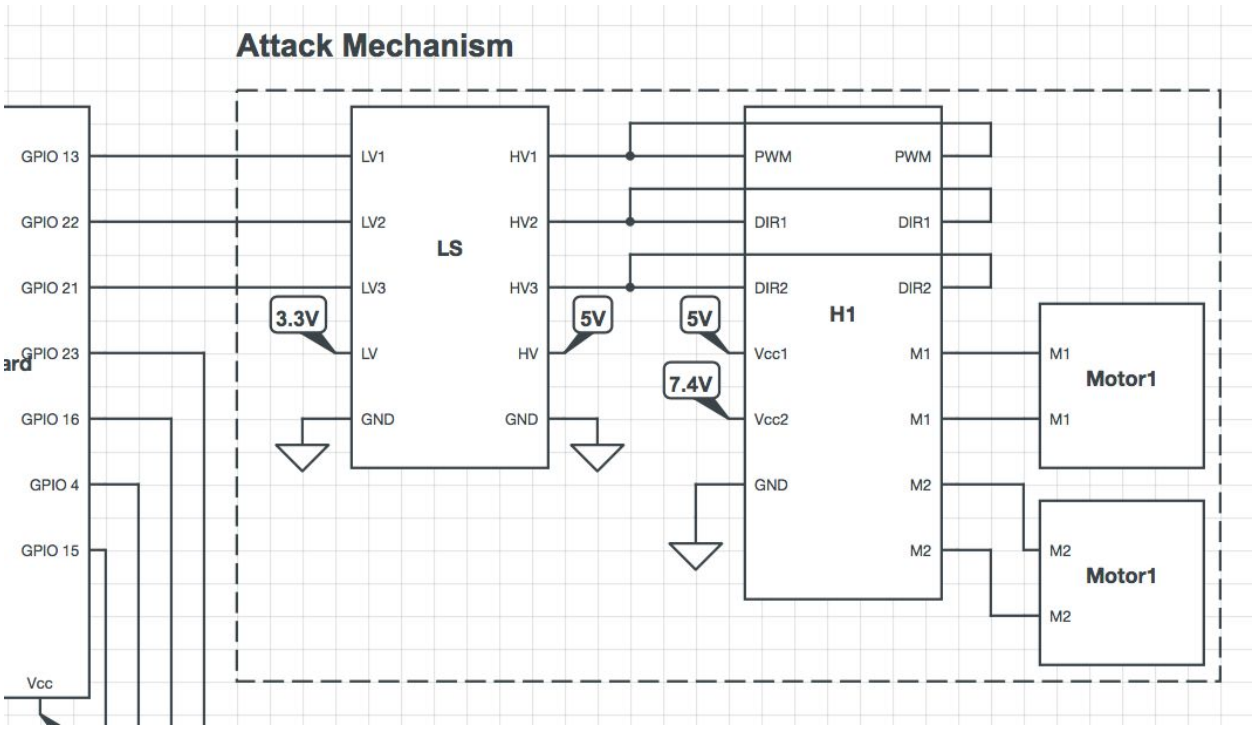

Figure 17. Attack mechanism schematic.

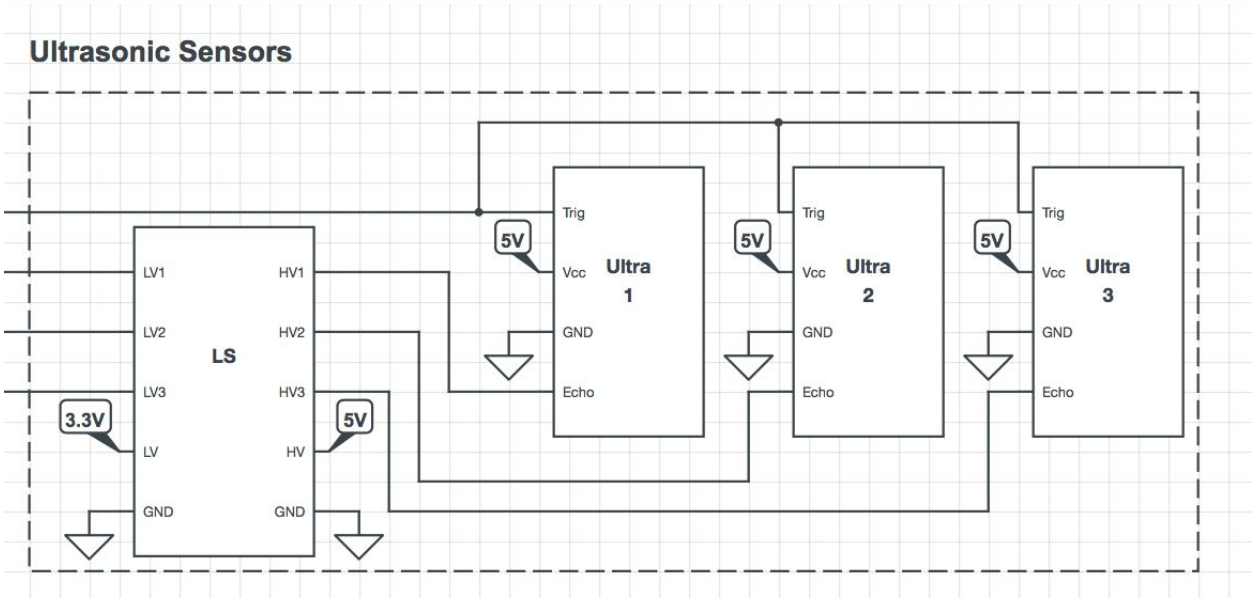

Figure 18. Ultrasonic sensors schematic.

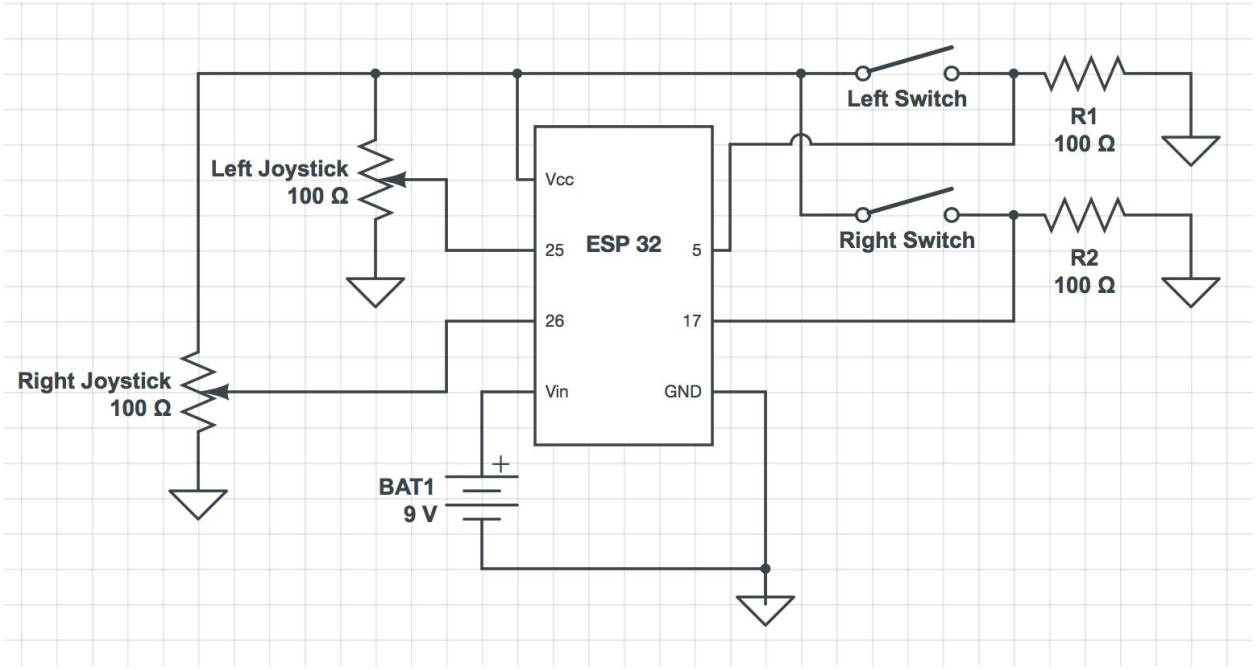

Figure 19. Remote control schematic.