Final Project Report

Brian Grimaldi

Functionality:

In terms of winning the game, I decided to allow for the car to be robust enough to withstand collisions with other cars, blocking them from taking cans from my team's side of the field. However, despite the defensive capabilities of the car, I also added two servo motors to grip cans from our side and the opponent's side. Two servos would more than account for the weight of the cans, being able to hold onto them with ease and slowly carry them to a desired location. In terms of autonomy, I have added 3 sensors to my robot. The first sensor was a simple switch attached to the right claw. This switch would be depressed when a can hits it in autonomous mode, signaling to the car there is a can next to it. The car would then steer into the can and close its claws to capture it. It would then exit the autonomous mode to be driven back to our side of the field. The second sensor is a short flex sensor on the left claw. This sensor is long enough that it would give the program installed on the car ample time to detect when it hits a wall in the autonomous wall following mode. When it is bent slightly, the car will turn 90 degrees to continue the wall following. Finally, attached to the right-hand side of the car is a VL53L0X Time of Flight sensor. This sensor detects how far the car is from the right-hand side wall, adjusting the PWM signal on the left and right motor using a PID controller to drive alongside it in a relatively straight manner.

Regarding performance, the car can drive very smoothly, however it tends to skid slightly when reversing. This could be due to the wheels not aligning as perfectly as desired and the heavy weight of the car exacerbating this slight design flaw. Initially, there were plans to use a rotary encoder for both DC motors and a touch sensor instead of a switch. Regarding the rotary encoders, they were unable to properly read the RPM of the motor and with there being many issues regarding using these with the ToF sensor, I decided the payoff of controlling the motor speed using these would be miniscule compared to the grand scheme of its autonomous driving. Regarding the touch sensor, the values that the ESP32 reads on the designated TOUCH pin were very unreliable. Due to the noise of 6 PWM signals and 5V outputs on other pins, the TOUCH pin gave many false reads, indicating that the exposed wire had been touched by a capacitive surface when it had not actually done so. As a result, I found that the same result could be achieved by putting a physical switch in its place, attached to a INPYT_PULLUP pin on the ESP32. This gave way more reliable results when detecting cans.

In the evaluation, the car worked as expected. First it was able to grab cans and move them to the doubling square with much ease. For wall following, the car was able to travel 50% of the circuit but failed on the 2nd turn/third side of the field. This is due to the constant parameters not being tuned properly for the GM lab field. At home, the car tended to not have enough traction, making a much longer turn in the GM lab. For the future, having a much quicker turn and a smaller proportional constant so the car doesn't go too fast into the wall when the error is large. Another issue was with the autonomous can grabbing. As we will talk about in the software section, the car did not have the proper length to the wall that it should be maintaining, leaving it at a larger distance and was unable to touch the can one inch from the wall, just passing it as a result. In the future, shortening this desired distance and possibly having more complex sensors to detect the can would result in better detection.

Mechanical Design:

For this competition, I decided to build upon my original design from Lab4. I found my mobile base was quite agile and very sturdy due to the rigidity of PLA plastic. It also allowed me to design new parts, making it a very modular design as I could iterate very easily if certain parts needed tweaking. An aspect that I felt needed to be improved on were the motor harnesses. They were very flimsy due to the peg holding the motor in place being fragile and breaking frequently. As a result, a new design was made, allowing for bolts to be placed through the harness and motor, and nuts to hold everything in place. This gave a very sturdy motor harness that would allow for the car to drive very reliably.

The intended approach was to tweak the design from Lab4 of the mobile base to make the print faster to materialize and give it more modularity, assembling pieces for the base rather than have one large piece with little flexibility. However, due to many issues and replacement parts being ordered for my 3D printer, I was able to retrofit the mobile base from Lab4 relatively well for the purposes of the final project. As stated above, the performance of the car is particularly good, driving in a relatively straight line and sturdy enough to withstand collisions with other obstacles.

Pictures and CAD files of the car can be found in the appendix.

Electrical Design:

The intended approach was to use 3 main boards: the ESP32 as the microcontroller, a DRV8833 two motor driver breakout, and a 5V breakout. The DRV8833 would drive the two DC motors and would be powered by 4 AA batteries. This breakout board would take in 4 PWM signals, connect the ground of the AA batteries with the ESP32 and supply power/signal to both motors simultaneously. The 5V breakout board would be used to power the two servo motors, connected to a BESTON phone battery along with the ESP32. The ESP32 would be the hub for sensors (INPUT) and for PWM/3.3V logic (OUTPUT). This design would prove to work very well for most of the project life. However, due to a mishap in reading the instructions, the VM pin on the DRV8833 was accidentally connected to the 3.3V pin on the ESP32. This effectively destroyed the ESP32 output/input capabilities, servo motor functionality, and DRV8833 functionality, but was easily replaced due to leftover personal parts. The oscillasorta ESP32 was reprogrammed with ease since it shared the same model/version as the destroyed ESP32.

Processor Architecture:

This section will be written in text to better see where each pin is going to. Only one microprocessor is used, being the ESP32 Pico board.

- Motors
  - Motor Left: Pins 23 and 18
  - Motor Right: Pins 5 and 10
  - These pins output a PWM signal to one of two inputs on the motor driver breakout, the other pin being set to ground depending on the direction that the wheel is going in.
- Servos
  - Servo Left: Pin 4
  - Servo Right: Pin 25
  - These two pins use a PWM signal sent to the control wire (orange) of the servos, moving the servo horn to a desired angle.
- Time of Flight Sensor
  - SCL: Pin 22
  - SDA: Pin 21
  - These two pins allow the ToF sensor to communicate with the ESP 32.
- Flex Resistor
  - Output: Pin 32
  - Analog Read: Pin 37
  - Reads voltage on pin 37 as analog signal.
- Switch
  - Input pullup: Pin 27
  - Reads a 3.3V or 0V signal on pin 27
- Sleep
  - Output: Pin 14
  - 3.3V output to keep motor driver running.

<u>Code Architecture:</u>

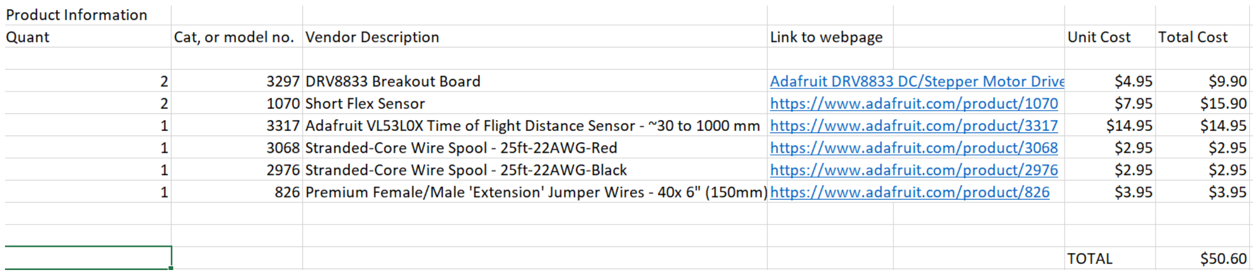
The code for this project built heavily upon the edited lab4demo code. This code allowed for the car to be driven through the web interface. On the driving side of the software, an additional two PWM pins were added since lab4 used NAND logic to control the direction of the motors, while the DRV8833 relies on two PWM/GROUND signals. New additions to the code included a gripper button and autonomous driving button. The gripper button allowed for the servo motors to open and close the claws, effectively grabbing/releasing the can in front of it. The autonomous button put the car into a wall driving mode. It would set a flag for the loop function to not respond to any inputs, allowing for the mode to be uninterrupted unless the autonomous button was pressed again. In this mode and handler function, there are 3 conditional statements. The first conditional statement checks to see if the right-hand switch is pressed, indicating the car has hit a can. If it has, it will drive right into the can, placing it within the grippers and grab it. It will then exit the mode so the driver can manually bring the car back around to the other side of the field. The second conditional statement checks to see if the flex sensor decreases below the 2600 value, indicating it has bent. If it has, this means that the car has most likely hit the wall, allowing for it to turn 90 degrees to continue wall following. The last conditional statement is the wall riding PID controller. In this statement, a simple controller is used to change the PWM signal of the two motors, turning the car slightly left or right to keep the car following the wall. Mostly, the software approach has been straightforward. There have been some bugs that prevented progress, such as sensors not being properly soldered or failing to initialize. Once these issues were fixed, the new code that was added was straightforward to implement. The only time-consuming aspect was tuning the car to drive straight with the PID constants. In the actual performance evaluation, there were issues with both the wall turning and can grabbing. For the wall turning, while the car was able to make consecutive turns at home, it was not tuned for the courser surface at the GM lab. This caused it to turn too much, then drive into the wall too fast since the error from its desired spot was so great. While there was a derivative component, it still went straight into the wall, triggering the can grabbed sensor and stopping its autonomous drive all together. For the future, it would be beneficial to be able to tune the car in the GM lab to make it more competition ready. The other issue was the can grabbing feature. The car was too far off the wall detect a can, normally passing by it or getting stuck on it but not triggering the can grabbed sensor on the right gripper. Code wise, this can be fixed by shortening the distance between the ToF and wall so that the car will be closer to the wall. Mechanically, this can be fixed by having a larger right gripper, however this would then require for the wall detection to turn to be modified.

[redacted]

[redacted]

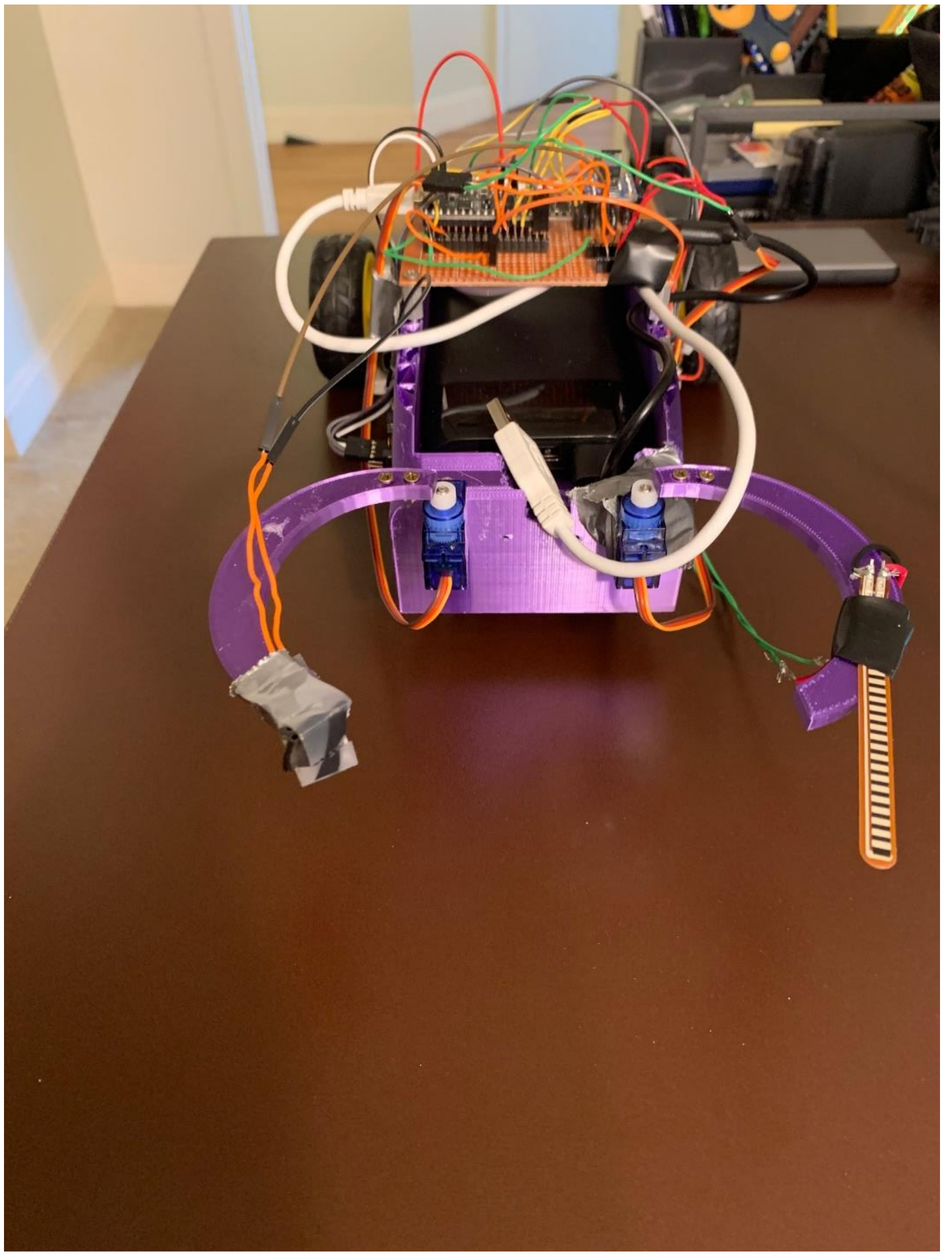
<u>Appendix:</u>

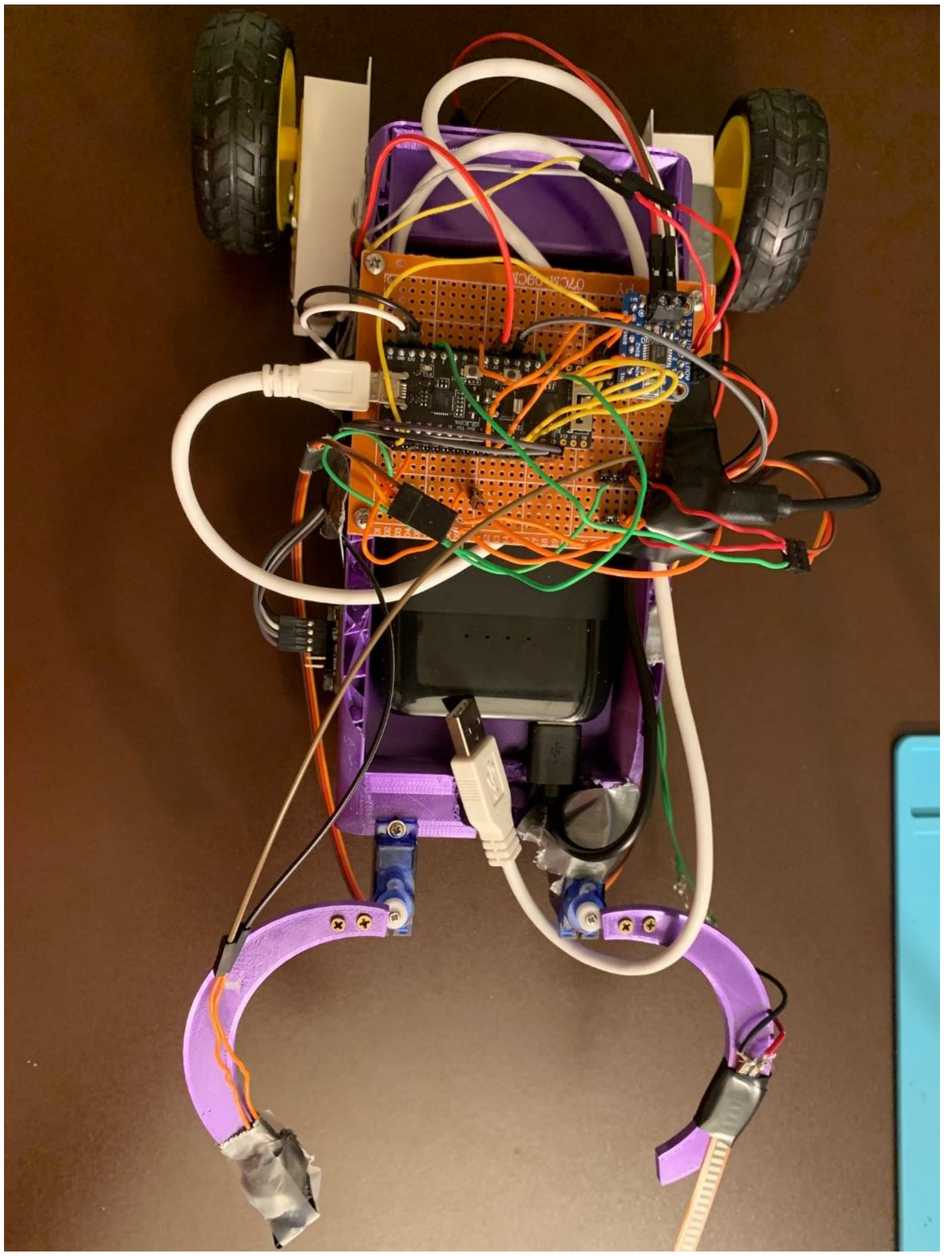
BOM:

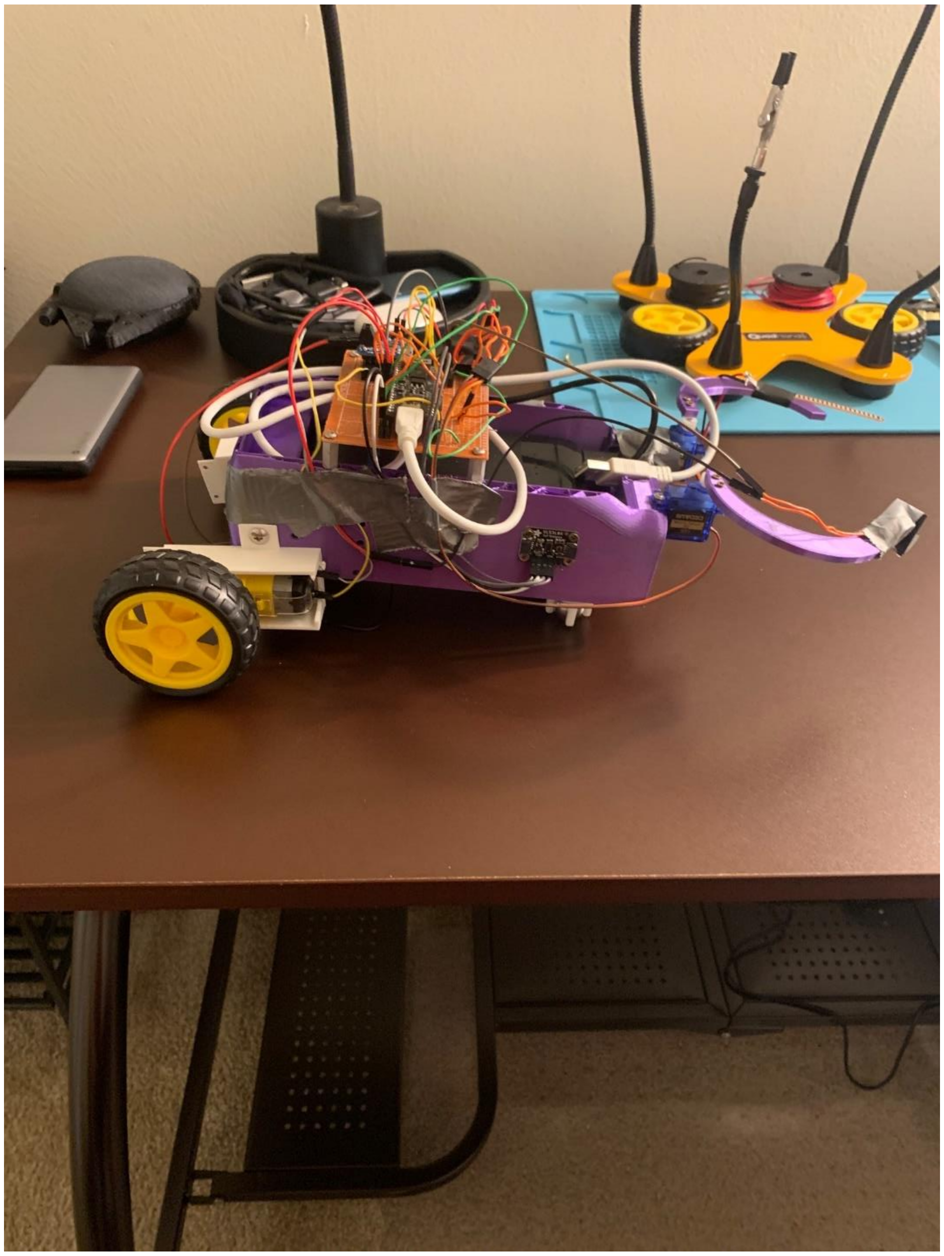


Product Information						
Quant	Cat, or model no.	Vendor Description	Link to webpage		Unit Cost	Total Cost
2	3297	DRV8833 Breakout Board	Adafruit DRV8833 DC/Stepper Motor Drive		$4.95	$9.90
2	1070	Short Flex Sensor	https://www.adafruit.com/product/1070		$7.95	$15.90
1	3317	Adafruit VL53L0X Time of Flight Distance Sensor - ~30 to 1000 mm	https://www.adafruit.com/product/3317		$14.95	$14.95
1	3068	Stranded-Core Wire Spool - 25ft-22AWG-Red	https://www.adafruit.com/product/3068		$2.95	$2.95
1	2976	Stranded-Core Wire Spool - 25ft-22AWG-Black	https://www.adafruit.com/product/2976		$2.95	$2.95
1	826	Premium Female/Male 'Extension' Jumper Wires - 40x 6" (150mm)	https://www.adafruit.com/product/826		$3.95	$3.95
					TOTAL	$50.60

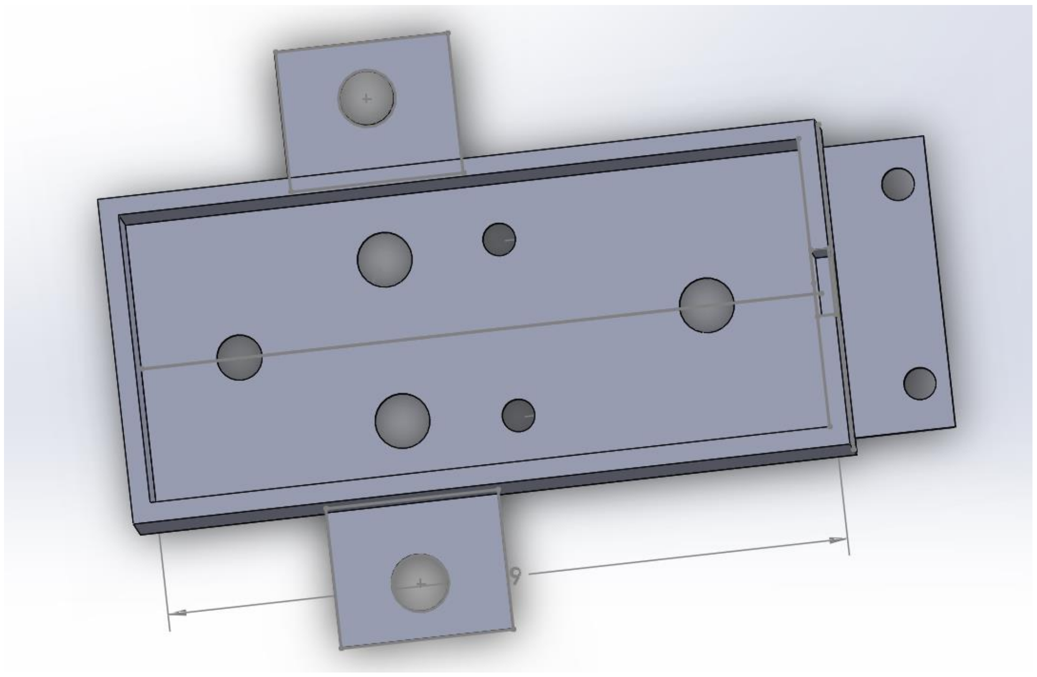
Car photos:







CAD of motor assembly:



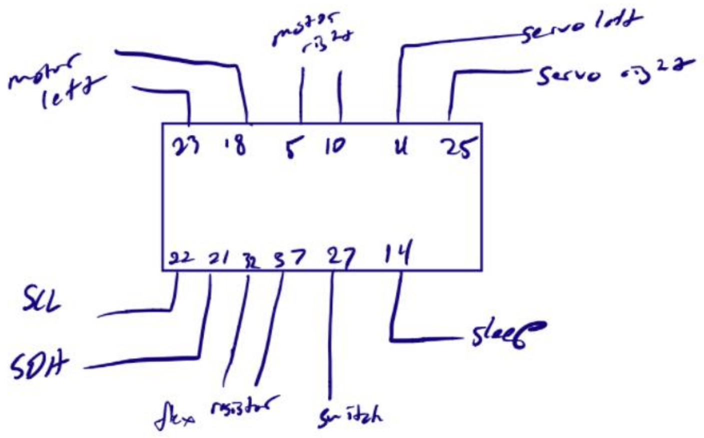
Data sheets:

DRV8833: [DRV8833 Dual H-Bridge Motor Driver (Rev. D) (adafruit.com)]

Flex Sensor: [FLEX SENSOR DATA SHEET '10 (SPARKFUN KIT).ai (adafruit.com)]

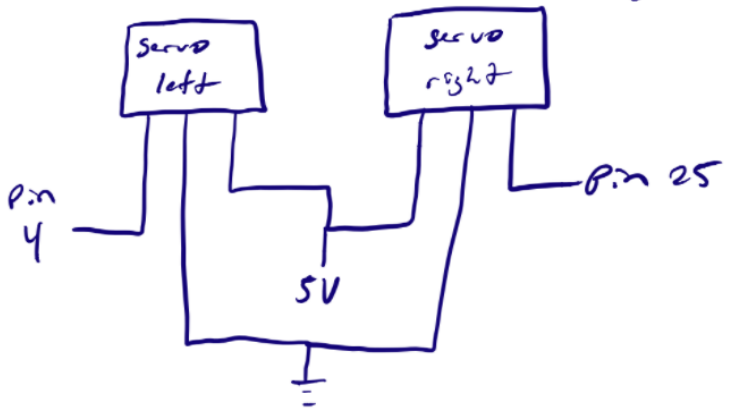
VL53L0X: [World's smallest Time-of-Flight ranging and gesture detection sensor]

<u>Electronic Circuit Schematics:</u>

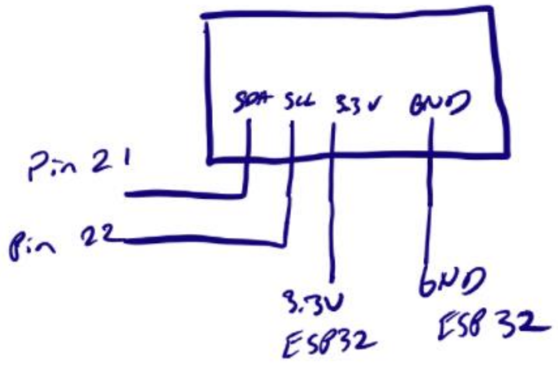


Servos

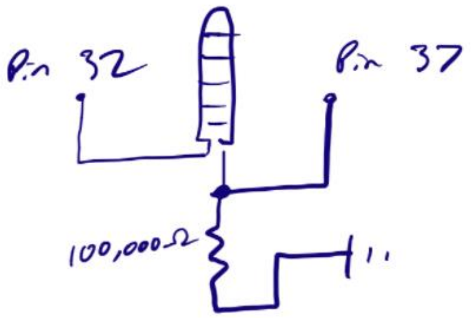
Powered by Boston battery



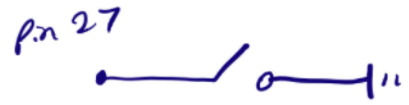
Time of Flight



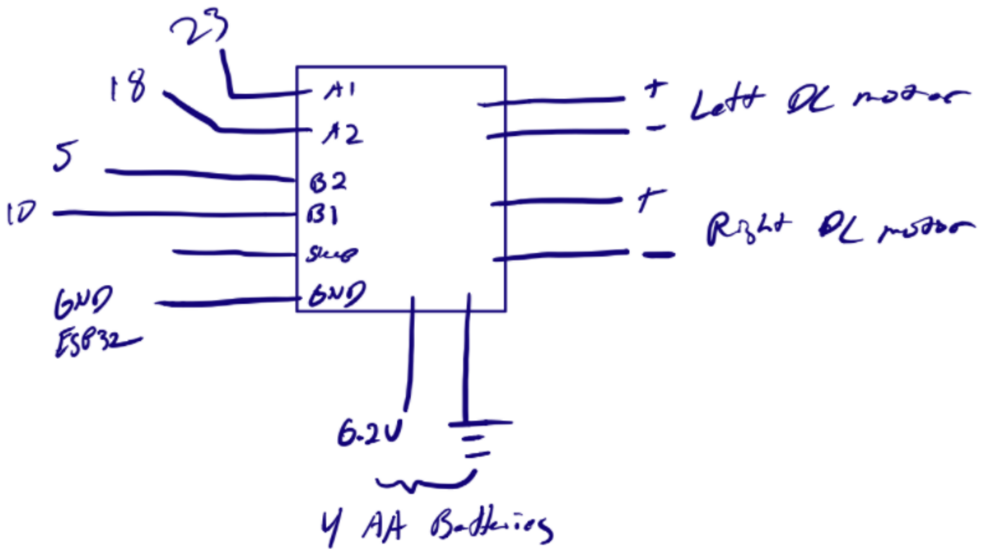
Flex Resistor



Switch



Motor Drive DRV8833



Links of functionaliry:

Wall Following and Turning: https://youtu.be/0e13hofVi5I

Grippers working: https://youtu.be/7CIP9Vw7Aj0

Autonomous Wall following: https://youtu.be/S2wu-Sbfxg0