Xinyuan Zhao
MEAM 510 Final Project Report

Game and Functionality Approach:
In the game “GTA 2021”, participants form teams of 2 and compete against each other in scoring points. The points can be scored by staying at the opposite side, putting cans into the doubling square, and stealing cans from the opposite side. My robot is relatively heavy and slow, but it is very reliable in grabbing the can and pushing it to the doubling square. Since my robot is slow and has limited sensing capabilities, I plan to mainly play on the defensive side and push as many cans as possible to the doubling square. My robot will also protect the cans on our side. I will also go to the opposite side occasionally and try to steal their cans in the scoring square.

The main functionalities of my robot are Wall Following and Grabbing Cans. It has ultrasonic sensors on the right and front side to read the distance. A gripper with a heavy duty servo is mounted on the front of my robot for grabbing the cans. There are two lever switches that act as touch sensors. One of them is mounted on the forearm of the gripper to detect the collision of the gripper with the wall, and the other is placed at the inner middle of the gripper to detect the can that is inside the gripper. So the wall following is controlled by two ultrasonic sensors and the lever switch at the front of the gripper. The grasping is controlled by buttons on the webpage and the lever switch between the gripper arms. I also developed a nice webpage display to continuously track the distance readings, wall following and gripper status.

During the evaluation, my robot can efficiently pick up the cans and drop off at the scoring area. It is also capable of following the wall and making full circuits within 2 minutes. However, I only used a simple control method for wall following instead of PID control. So the right distance with the wall is not stable. Since the friction of the field at GMLab is very different from my home, it is hard to optimize the parameters. For those reasons, my robot often fails in stealing the can from the opposite side. It has about 50% success rate in following the wall and grabbing the can automatically when I tested it at home. During evaluation in the GMLab, my robot successfully stole one can from the opposite side, but failed to steal more since the distance with the wall could not be maintained very well especially after turning 90 degrees. If I had more time, I would use the encodes and develop PID controllers for the motor speeds and wall following. The PID control would give a much more reliable performance in any field (The Demo videos of functionalities when I tested at home are included in the Appendix).

Mechanical Design:
For the mechanical structure of my robot, I built up based on the mobile robot I used in Lab 4. The dimensions of my robot are 8” in length and 4” in width. In Lab 4, I found that the motor and wheels I used could not provide enough torque and traction with ground. So in the project, I decided to replace the motor with one with larger torque and replace the wheels with thicker ones.

For the motors, I initially tried the 6V, 1:200 plastic gear motor from Pololu which provides 7.2 kg cm torque at stall and draws 800 mA current at stall. However, When I sent commands to this pair of motors, they were running at different speeds even with the same voltage. One motor always responds slower than the other. They are also not compatible with encoders. So I decided to just switch to another type of motor. The new motor I chose was the 6V, 1:90 gear motor from Adafruit. This motor provides about 2 kg cm of torque at stall and draws 1A current at stall. Although the new choice of motor has much less torque, they are enough for moving the cans and run faster than the previous ones. After testing, the new pair of motors have the similar speed response to voltage and so are much easier to control. The maximum torque and current draw are very important features to look at when choosing motors. And I also learned that the motors can be more reliably controlled by encoders and PID controls. If I had more time, I would use encoders and PID control to make the motor commands more robust.

In order to grab the cans in the field, I also need to attach a gripper in front of the robot. I found a gripper kit online which is compatible with my mechanical structure, and it can grab items up to 4.2" wide. So I decided to purchase this gripper kit and use it in this project. This gripper kit uses a HS-322HD heavy duty servo which has a 3.7 kg cm torque and draws 800 mA current at stall. The gripper kit with this heavy duty servo is installed with four 6-32 screws at the front of the robot. And the Beston Battery is placed at the back of the robot to balance the weight. Photos and CAD Drawings of my robot are included in the appendix.

Electrical Design:
The electrical design of my robot can be divided into power supplies, motor control, gripper control and touch sensor, and other sensors.

- Power Supply: My initial design was to use 6V AA batteries to drive motors and gripper servo, and use a small 5V, 2A power bank to drive the ESP32s and sensors. Since the Beston Battery is large and heavy, a smaller power bank can decrease the load. However, I found that the gripper servo could not be controlled when it is in the same circuit with the motors. The servo on the gripper would unstably wiggle around when motors are running. I suspect that this is caused by voltage noise in the circuit when motors are running. The servo angle is sensitive to voltage fluctuations, so I decided to move the servo motor to a separate power circuit driven by Port 2 of the Beston Battery. So I ended up still using the Beston Battery and the new power circuits design is: Beston Battery Port 1 drives ESP32s and sensors, Beston Battery Port 2 drives gripper servo, 6V AA Battery drives motors.

- Motor Control: Same as in Lab 4. I used an H-bridge to drive the motors. To avoid short circuits, NAND Gates are used to make sure voltages on the same side of the H-bridge are logically inverted. A circuit diagram of motor driver circuit is shown in Appendix Figure 1.

- Gripper Control and Touch Sensor: One function of my robot is to grab the can automatically when it senses that this can is inside the gripper. So I need a kind of

sensor at the middle of the gripper so that the gripper will close whenever the can touches this sensor. The easiest way to do this is to use a SPDT Lever Switch. The lever switch is attached to the inner middle of the gripper. When the robot is going forward in autonomous mode and the can is inside the gripper opening, then the can will be guided to the inner middle by the gripper arms and press the lever switch. The lever switch has 3 ports: COM, NO and NC. COM is connected to 5V, NC is connected to GND, and NO is connected to a pull-up PIN in ESP32. When the switch is pressed, COM switches from NO to NC and so NO will be LOW. So the command of closing the gripper is sent whenever the pull-up PIN connected to NO port is LOW. A circuit diagram of gripper and switch is shown in Appendix Figure 2.

- Other Sensors: In order to perform autonomous wall following, we need sensors to detect right distance and front distance. For the right sensor, I chose the ultrasonic distance sensor because it is cheap, easy to implement, and fairly reliable in sensing distance from 2cm to 500cm. The ultrasonic sensor I chose has a resolution of 0.3cm and it is generally reliable and accurate for wall following at a distance to the right of 4 - 7cm. For the front sensor, I initially planned to use an PSD sensor, but it was not working well with the wall following functionality. The PSD sensor I used was GP2Y0A02YK0F from Sharp, which has a range of 20cm - 150cm. During the test, it has a very low sensitivity for objects close to it. Considering that there are other IR LEDs in the field and I need to accurately detect cans in front of the robot as well as the walls, I decided to also use an ultrasonic sensor as my front range sensor. Since the ultrasonic sensor is relatively unreliable when it has an angle with the wall, I decided to add another lever switch at the very front of my robot to detect collisions with the wall. This lever switch acts as a front limit detector, so whenever the robot hits the wall at the front, the lever switch will be pressed and the robot will back up and turn. In this case, my robot will back up and turn when either the ultrasonic sensor at the front senses an object or the front lever switch is pressed. This guarantees that my robot will back up when hitting the wall at the front instead of stucking when ultrasonic sensors read wrong distances. Aftering testing the Ultrasonic and PSD sensors, I learned that we need to consider sensor range, budget and precision when choosing sensors. PSD sensors can achieve a very high precision and accuracy, but it generally does not work well for objects close to it. Ultrasonic sensors are less precise but are generally reliable for sensing flat objects like walls in a close distance from 2cm to 20cm. So in my case, the ultrasonic sensors satisfy my design requirements.

Processor and Software Architecture:

I am using 2 ESP32s as the processors. ESP32-1 is the pico kit MCU and it serves as the main hub for processing sensor inputs and motor controls. The second ESP32 is the Node32 MCU and it controls the gripper servo by reading inputs from lever switch and commands from ESP32-1. The ESP32-1 will send commands to the second ESP32 to control the closing and opening of the gripper servo. Since these commands are binary, I just used the digital I/O pins to connect outputs to the inputs of the other MCU to send commands. ESP32-1 is powered by USB-C from Beston Battery and its 5V output provides power to all the sensors and the second

ESP32. The schematic of the ports usage and processor architectures is shown in Appendix Figure 3.

My software part consists of 2 programs: one for the main ESP32-1 and the other for the ESP32-2.

- ESP32-1 Code: This code contains the web server, motor control, wall following control and sensor readings. For web server and motor control, I continued to use the “Joystick and Tank Mode Example” which was used in Lab 4. For the wall following, I firstly added two buttons on the web page to start and stop the wall following mode. These two buttons turn on and off a global variable “auto_state” in the ESP32 code. The Wall following function basically checks the right and front distance readings. If the right distance is less than a limit, increase the PWM for the right motor to turn slightly left. If the front distance is less than a limit, then back up and turn 90 degrees left. If the front lever switch is pressed, then it means that the right gripper arm hits the wall and so the robot will back up and turn about 30 degrees left. Otherwise the robot will continue to go slightly towards right. This control is not very robust and sometimes the right distance is not maintained correctly. So I would develop a PID controller with encoders on the motors if I had more time. In the main loop, the program checks right and front distance readings and updates the web server every 2 ms. Then the program checks if the wall following mode is turned on. If yes, run the wall following function, otherwise update the motors by manual controls on the web page.
  During testing, I found two problems for this code. First problem is that my robot will run into hysteresis when the front distance is right about the limit. The robot will go back and forth forever in this case because when the robot detects that (distance < limit), it backs up a little bit, and when it checks again after 2 ms, the (distance > limit) and it starts to turn. But when it is turning, the distance may go below limit again and it starts to back up again. In order to fix this problem, I need to make sure that the robot only checks the distance again when back up process is complete. So I separated the wall following function into two. And instead of checking distance every loop, it checks the “follow_state”. When distance goes below limit, it will change the “follow_state” so that another function for backing up and turn will be called and runed until the “follow_state” is changed back when it completes the 90 degrees turning.
  Another problem I found was that the robot will turn when the front sensor senses the can or the front lever switch touches the can. So I added another mode called stealing mode. When the stealing mode is turned on from the web, the robot will turn off the front sensors’ control and only maintain the right side distance with the wall. Switching mode is done by changing the value of the “auto_state” variable which is checked every main loop.

- ESP32-1 Web Page Code: Another part of this program is the display on the web page. The web page (Client) can request data from ESP32 (Server) using AJAX code. I used the ‘setInterval()’ function to request data from ESP32 every 0.5 second and post the data onto the web page using the function ‘getElementById()’. The data requested

contains front and right distance, gripper status, wall following status and gripper lever switch status. A screenshot of the web page display is shown in Appendix Figure 4.

- ESP32-2 Code: The code for the second ESP32 controls the gripper servo by reading the commands from ESP32-1 and from the lever switch inside the gripper. Initially, I was just using 1 digital I/O to connect the two ESP32s. So the ESP32-1 will turn this digital pin HIGH when the grabbing command button on the web page is pressed. When the release button is pressed, the ESP32-1 will turn this digital pin LOW. But then I realized that this would cause conflicts when there is another lever switch which also commands the gripper. The logic here is a bit more complicated than I thought. In the main loop, the program checks the gripper command pin from ESP32-1 and also the lever switch pin. If the lever switch is pressed, the gripper will close automatically as there must be a can inside. However, the gripper command pin is still LOW as the grabbing button from the web is not pressed, so the two commands will be in conflict. The logic I intended was that: Close the gripper when either lever switch is pressed or web page grab button is pressed, and open the gripper whenever the release button is pressed. So this will need 2 digital I/O pins connecting the two MCUs. Furthermore, during the tests, the lever switch can be released when the gripper is closed and the can is inside the gripper but not pressing the switch. In order to make sure the gripper stays closed throughout the carrying process, I additionally set a global variable in the program. When either lever switch is pressed or the web page grab button is pressed, this global variable is turned on and the gripper will stay closed. This global variable is only turned off when the release button is pressed. Additionally, I also added a reset button on the web page to reset both grab and release I/O pins to LOW in order to prepare for the next grab and release process. From this experience, I learned that simple logics can be harder to implement in embedded systems programming. The event-service model is truly useful in this case. As many sensors are running simultaneously, the event-service model makes control logics clearer and easier to debug.

Retrospective:

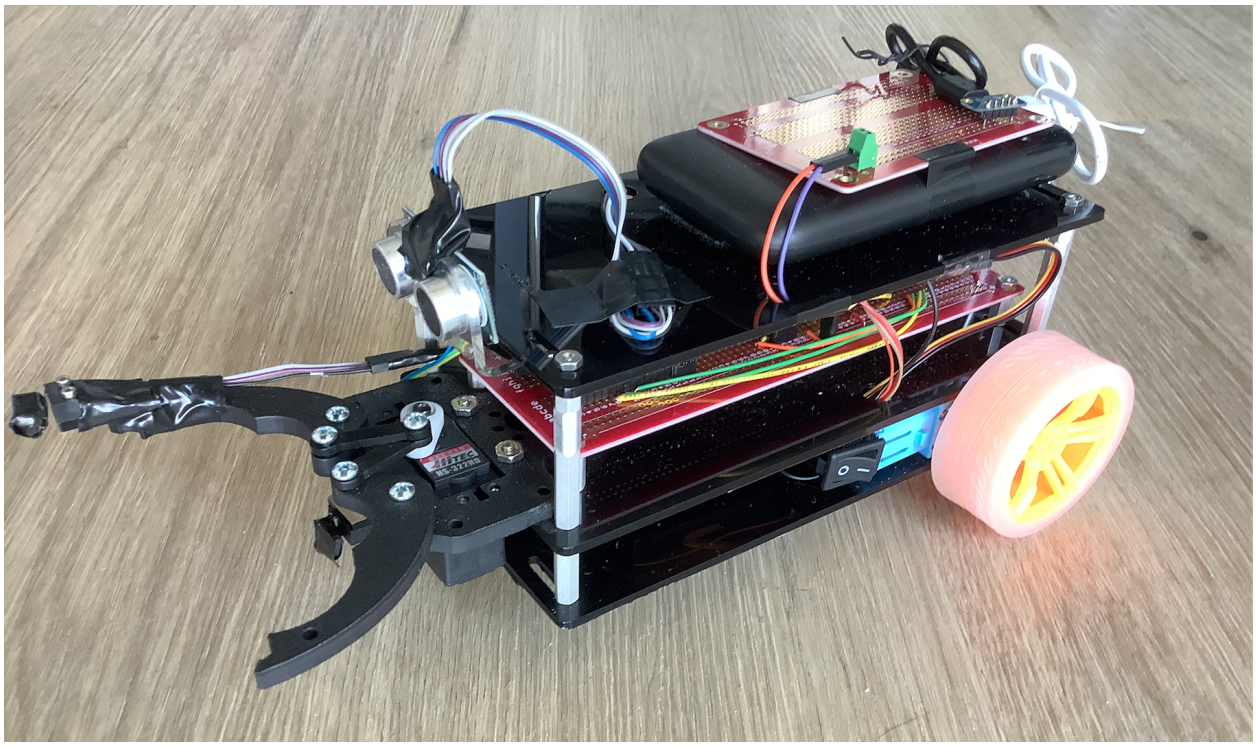
This is a great class! I think the most important knowledge I learned is embedded systems programming and hardware to software integrations. I gained much hands on experience with sensors, circuits design and programming. Topics like events and service models and wired communications are really useful for our future projects and research. The thing I was getting most trouble with is actually learning from home for this class. Taking the final project as an example, I ran into a great problem when I found that my wheels could not work with the field in GMLab because the friction is different from my home and I had to purchase and redesign the wheels within one week. Also it is hard to design mechanical structures when we do not have direct access to the tools and materials in the lab. If we were working in the lab, I would be able to work much more efficiently and learn more. So I would say that this course should not be offered online in the future. But for this semester, the class is still very enjoyable even though it is online. Professor Yim and the TAs are really really helpful. I really appreciate the time and effort the TAs and professor put into this class in this semester! Thank you for your help!

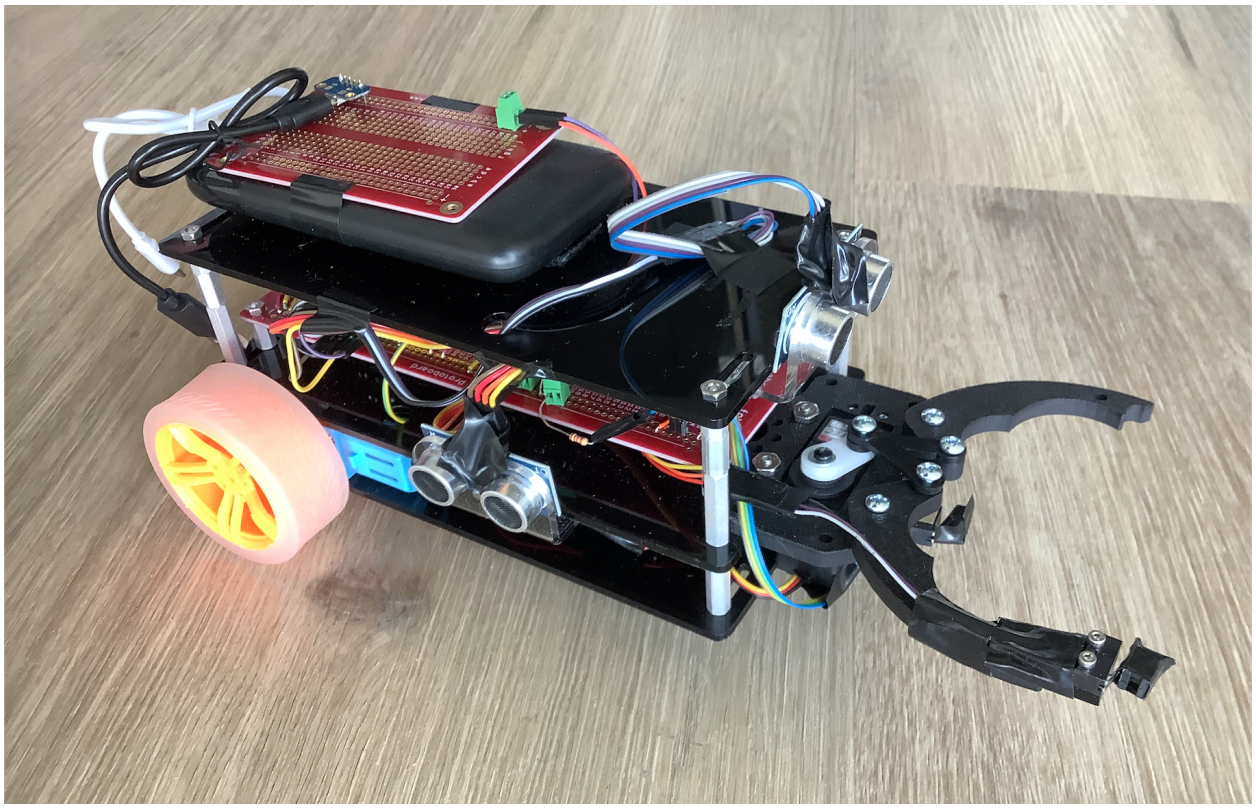
Appendix:

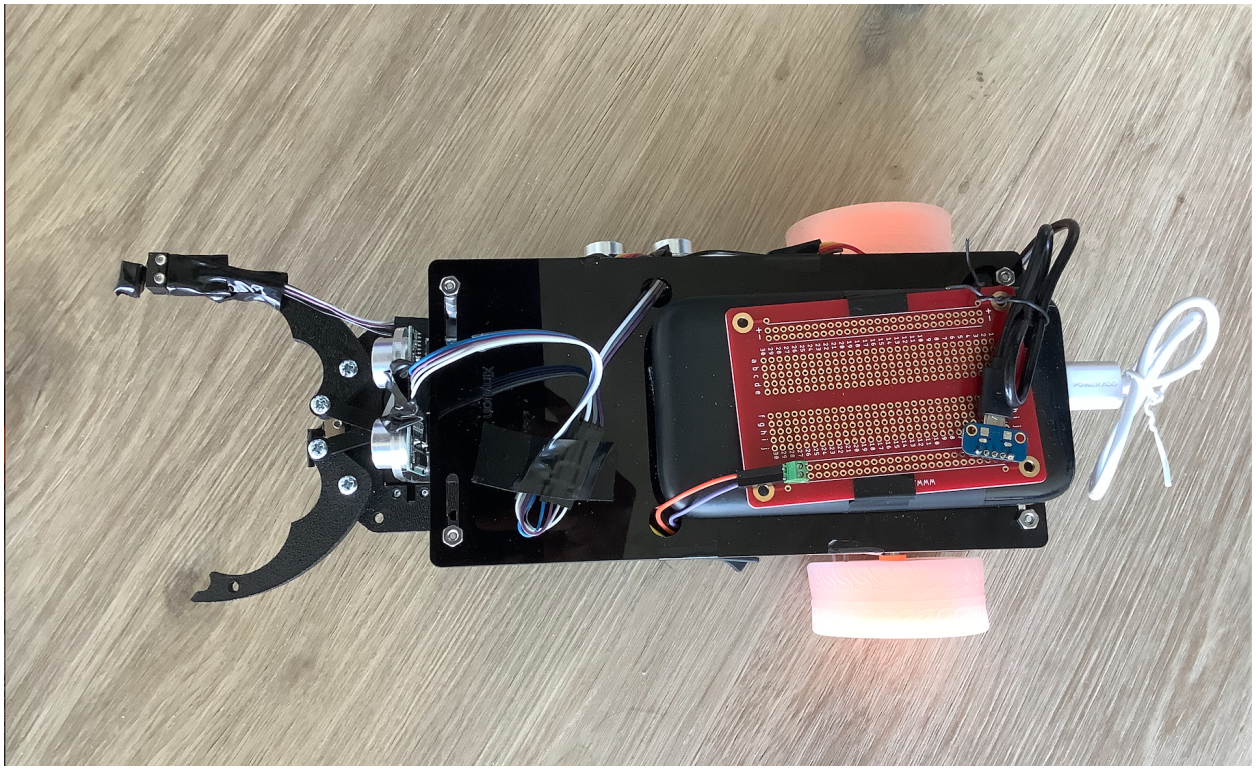
Bill of Materials:

Item	Quantity	Total Cost
1:90 Gear Motor	2	$11.90
2 types of Wheels	4	$6.00
Gripper and Servo Kit	1	$15.00 (self purchased)
Ultrasonic Sensors	1 pack	$9.59
AA Battery	4	From my home
Beston Battery	1	Given
AA Battery Holder	1	$2.95
1” 4-40 Standoff with screw and nuts	12	From GMLab
1/2” 4-40 Standoff with screw and nuts	8	From GMLab
SPDT Lever Switch	2	From GMLab
ESP32 MCU	2	From GMLab
SN754410 Dual H-Bridge	1	From GMLab
74HC00 NAND Gates	1	From GMLab
Solderable Breadboards	2	From my home
2-pos Screw Terminals	8	$4.72 from GMLab
Header Pins, IC Sockets and Wires	1	From GMLab
		Total Cost from Purchase: $50.16
		Budget Left: $12.48

Photos of My Robot:







Circuit Diagrams:

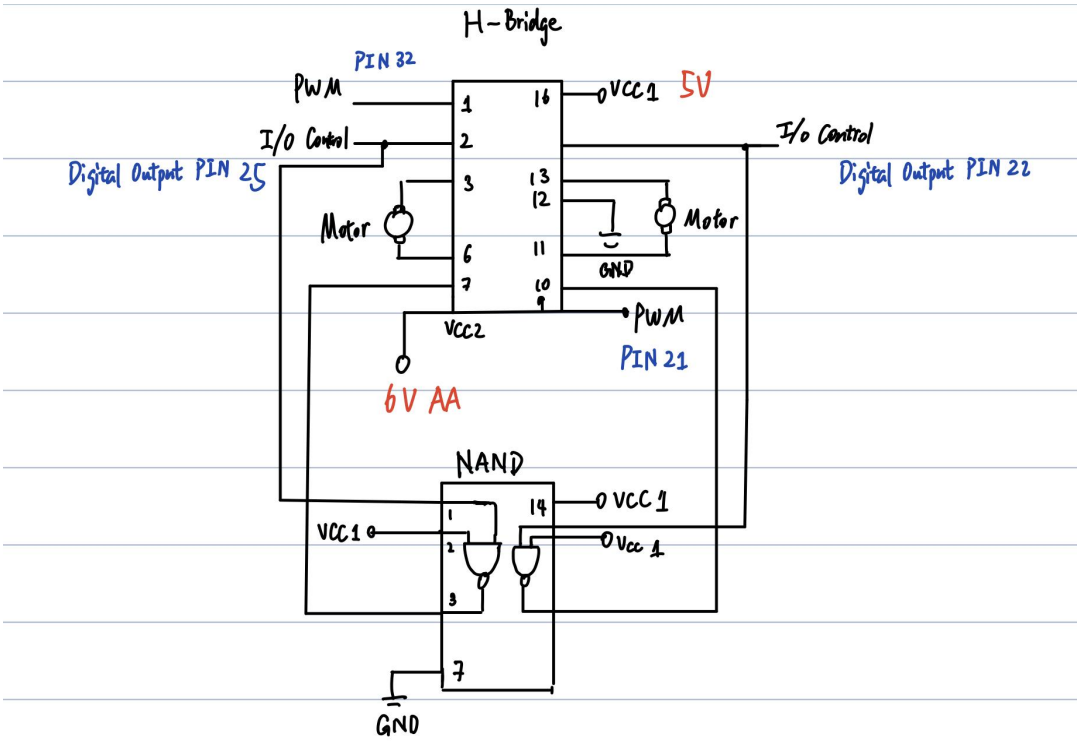


Figure 1: Motor Driver Circuit

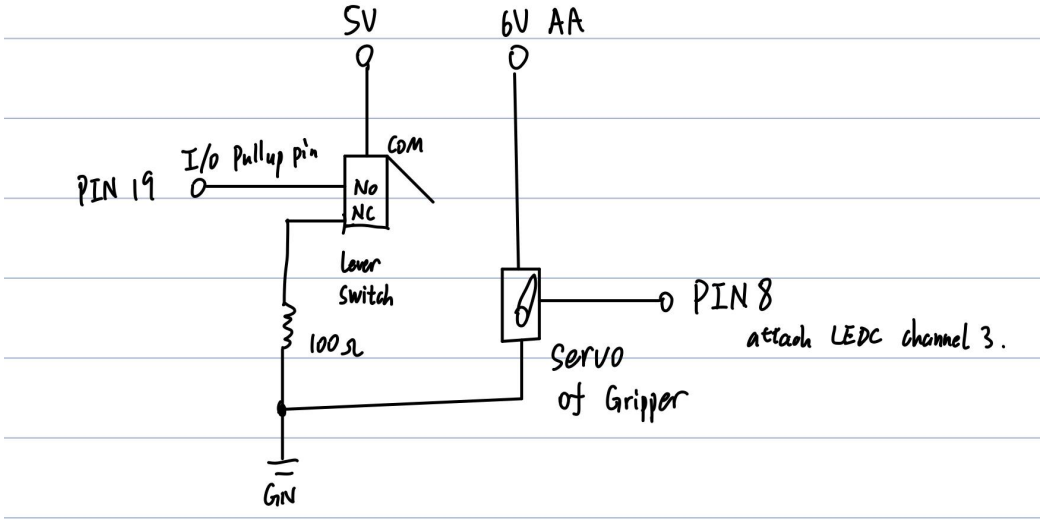


Figure 2: Gripper Servo and Lever Switch Circuit

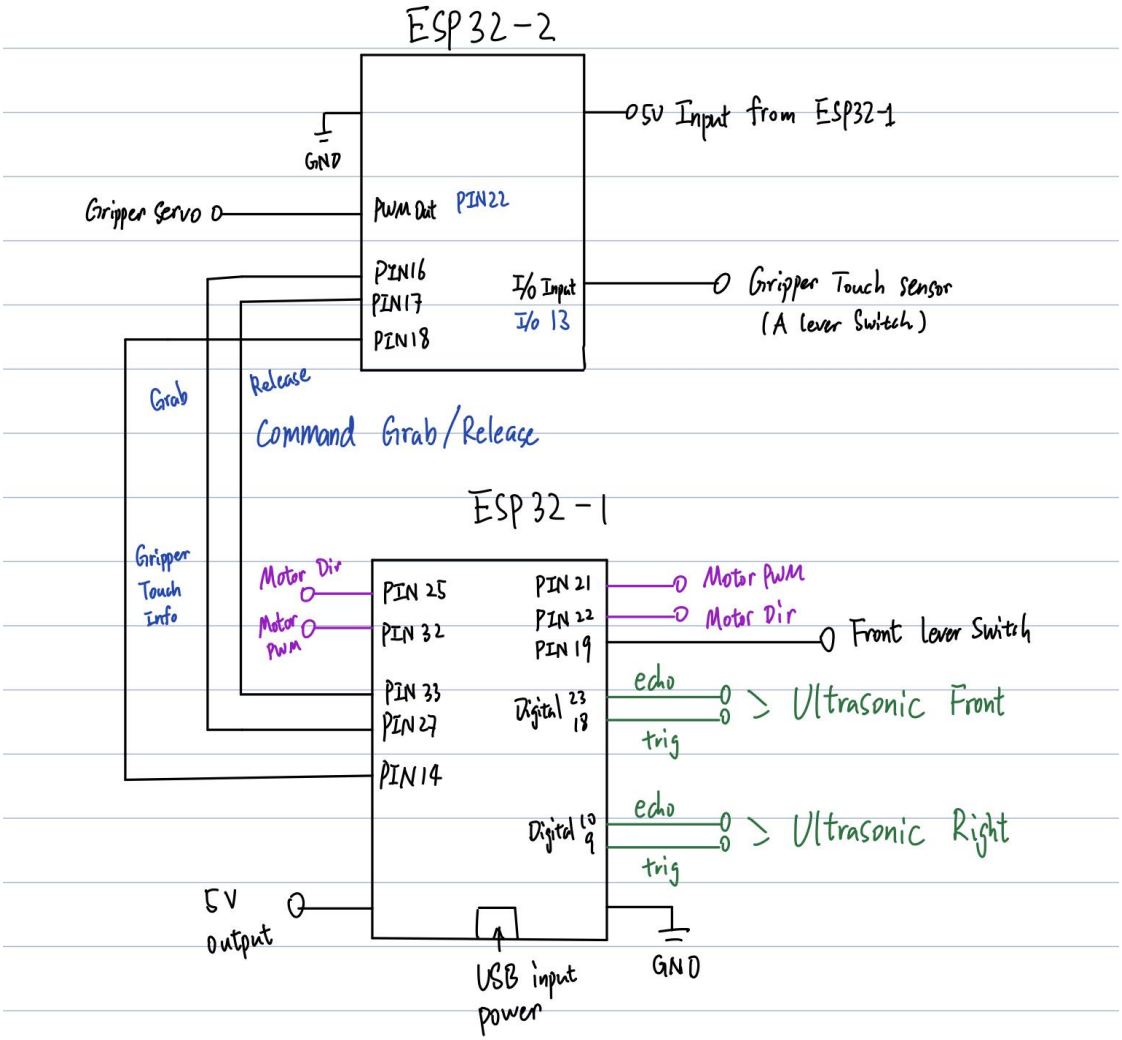


Figure 3: Schematic of MCUs Architecture and Port Usages

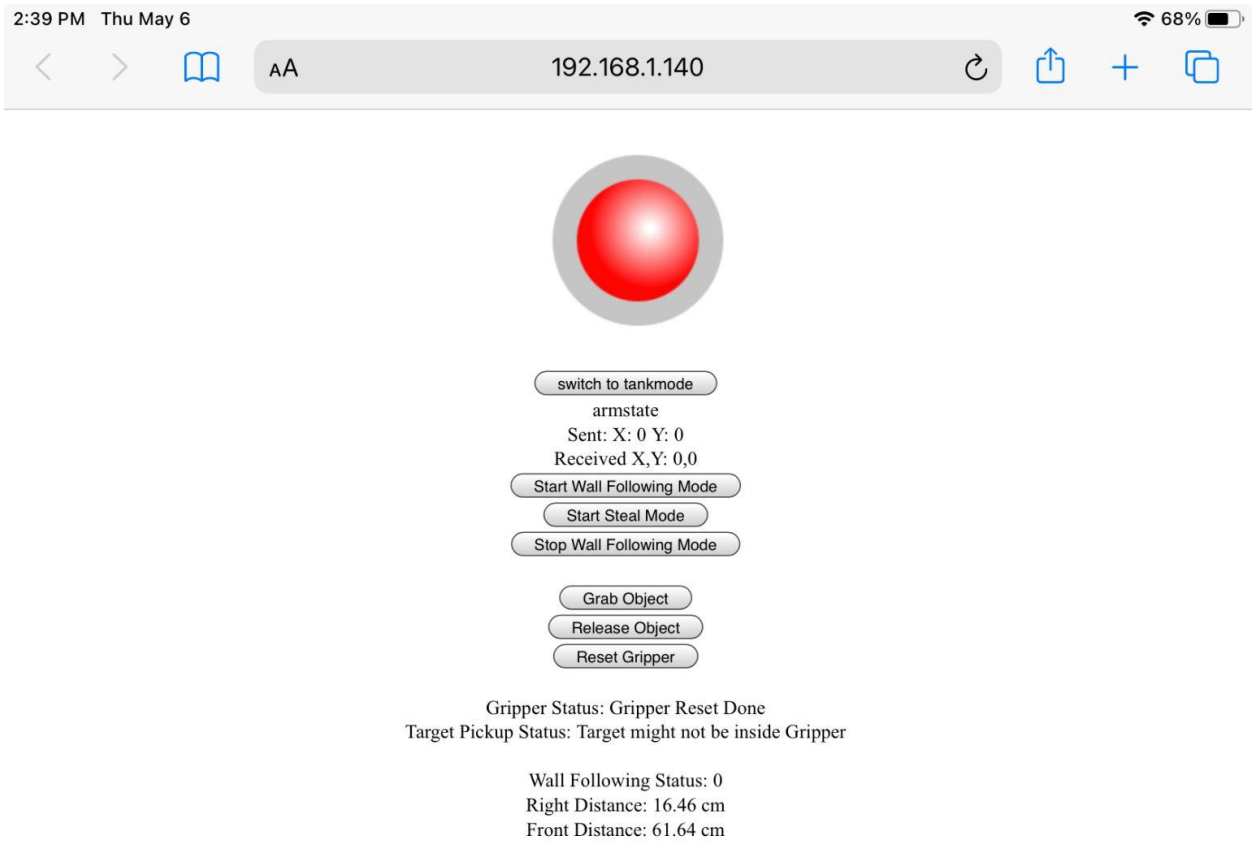


Figure 4: Screenshot of Webpage Display

Links to Datasheets:

- Gripper Servo (HS-322HD):
  https://www.servocity.com/hs-322hd-servo/

- 1:90 Gear Motor (not datasheets found, only description) :
  https://www.adafruit.com/product/3802#description

- Ultrasonic Sensor (HC-SR04):
  https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

- PSD Sensor I tested (Sharp):
  https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf

CAD Drawings:

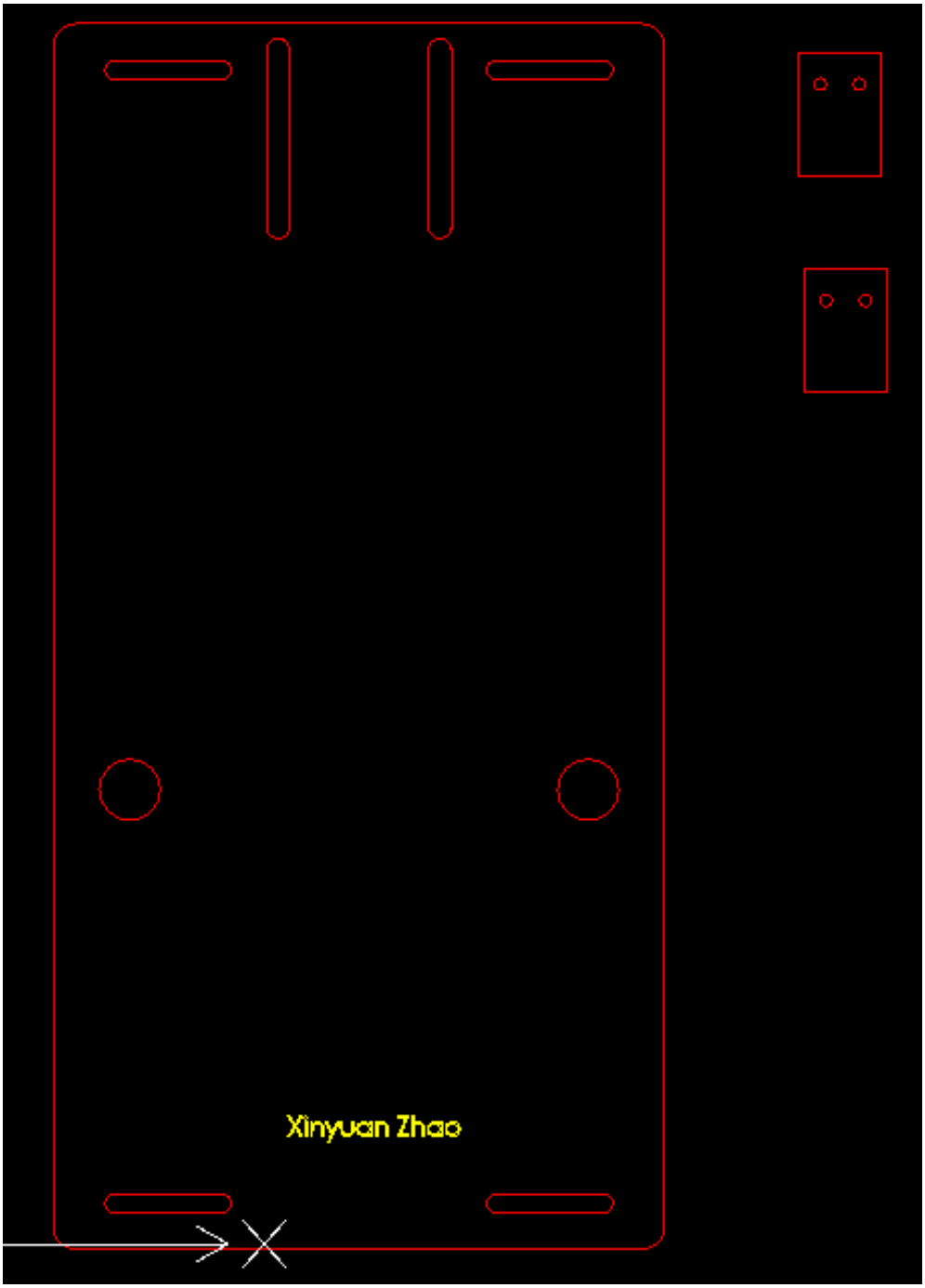


Figure 5: Left: Middle Layer holding Gripper and Circuits, Right: Lever Switch Holder

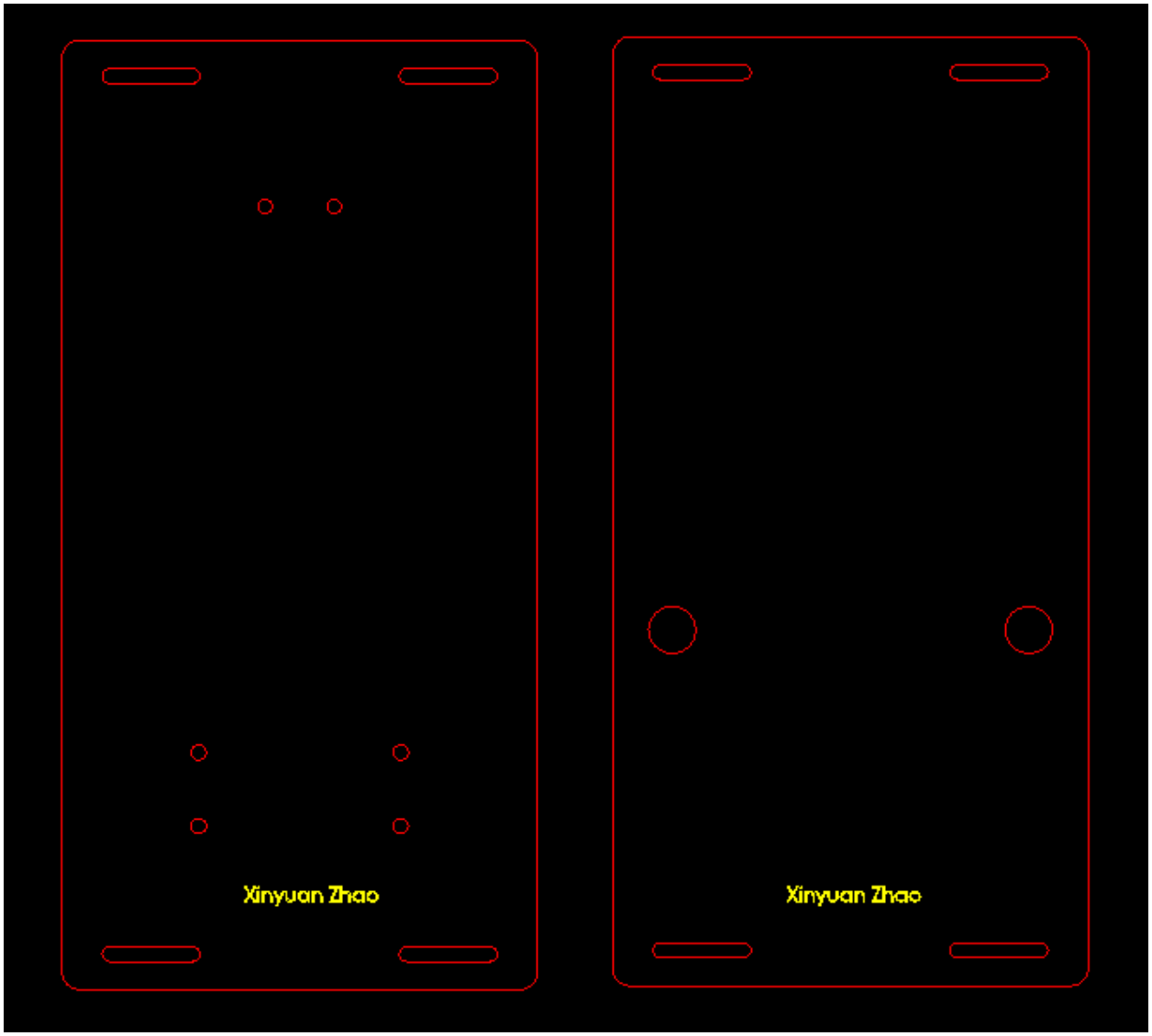


Figure 6: Left: Bottom Layer holding Motors and Wheels, Right: Top Layer holding Sensors and Battery

Links to the Additional Functionality Demo Videos:

1. Wall Following Demo at home:
   https://youtu.be/75_1uK2ZP-g

2. Stealing Cans Demo at home:
   https://youtu.be/Aom6hOYZl7s

3. Gripper Control Demo:
   https://youtu.be/jPeMM8FIv44

Fun Videos of my Robot Playing the Game:

- Almost stole a can from the opposite doubling square: https://youtu.be/xFULCZIHzoA?t=21514

- Defensive Plays from both sides and one of the highest scoring games in final rounds: https://youtu.be/rbyzACLpvsc?t=5441