Rowan University

Thai and Ardit + CS Majers

ECE 09.342: Intro to Embedded Systems Section 2

Sumobot Final Project

Authors:
Ardit Pranvoku, Austin Huang, Thai Nghiem, Matthew Rodriguez

Professor:
Dr. Gina Tang

Handed in: December 20, 2017
Due: December 20, 2017

1 Introduction

Our objective for this project was to design a sumobot that would communicate with our MSP430F5529 using the built in ADC to read values from sensor and control two motors using an H-bridge. Another objective was using UART to communicate with the sumobot for testing purposes. In order to implement these subsystems together in the final bot, we had to first build and test each subsystem before putting all together. We were successful in accomplishing these objectives and building a functioning sumobot from scratch.

1.1 Background

Some of the necessary material to accomplish this project was knowledge of how to implement a timer, pulse width modulation, and analog to digital conversion using the MSP430. These were useful for processing signals from the sensors on our bot, and generating signals to the motors. Aldo, the knowledge on PCB designing using Diptrace is critical for the operations of the robot. In order to properly construct a sumobot, a lot of support structures were needed. These were generally made using SolidWorks, and others were readily found on Thingiverse, a 3d printing database.

1.2 Equations

In order to calculate the baud rate we need to choose low frequency (8 bit) or high frequency mode (16 bit). In this application, we used low frequency mode, so the formula used to calculate baud rate is the following.

$$baudrate = \frac{8}{(UBAxBR0 + 1) * UCBRS_a + (8 - UCBRS_a) * UBAxBR0}(Hz) \tag{1}$$

Formula to calculate the duty cycle of the fan. TA0CCR0 remains constant at 1000 while TA0CCR1 is set by the control algorithm.

$$duty = \frac{TA0CCR1}{TA0CCR0} * 100 \tag{2}$$

2 Design Specification

The requirements of the project required sensing, actuating, and communicating between the MSP430 development boards and the outside world. The sumo bot project fulfills all of the project requirements. The sensing component will be fulfilled by the IR sensors that the sumo bot will need to use to detect foreign objects. The actuating component is completed by putting together all the subsystems involving code, electronics, and mechanical design. Lastly, communication is fulfilled by having the sumo bot have a function where it can be controlled via UART similar to a remote controller.

A general overview of the functioning of the system can be seen in Figure(1). A 12V energy sources powers a 5V regulator, a 3.3V regulator, and the motors that are connected to the H-Bridge. The 3.3V regulator powers the infrared LEDs, which reflect off surfaces and into the nearby infrared sensors. The sensors send an analog voltage back into the MSP430, which converts it into an analog signal and compares it thresholds that drive the logic of the H-bridge. Finally, the H-bridge controls the motors of the sumobot. The code is explained more in detail in the design approach.

3 Design Approach

Three versions of the PCB was designed for the sumo-robot, but only the second version was printed because of the time-constraint. In the last version of the PCB, there are 4 main sections. The first section is the break-out board for the MSP430 to be mounted on. The second section is the H-bridge IC and the bulk and bypass capacitors that control the behavior of the 2 motors. The third section, which is on the top, is the two voltage regulators (5V and 3.3V). These are both used to supply power for the H-bridge and the MSP430. Finally, the last section, which is right next to the voltage regulator, is used for the IR sensors. In this section, the holes for the IR sensors are not connected to the pins of the MSP430, as to allow changes when actually building the robot.

There were many challenges when finally implementing the sensor design of the scratch sumo bot. Initially, the infrared sensor (TSSP4056) picked out needed a casing to be more accurate. A 3d printed casing was designed

and printed as seen in Figure(14). The rectangular slot in the figure would contain the IR sensor, while the circular slot would contain the infrared or IR LED.

The initial IR sensor selected worked, but did not fit as well into the mechanical design of the bot, since it had to use a casing. Additionally, the team was not satisfied with the performance of the TSSP4056.

Upon further research of the old sumo bots used as references, the PRP220 IR sensor was discovered. This sensor was then ordered, since it was previously used in successful sumo bot designs.The sensor was tested by the built circuit as seen in Figure(7) . This circuit is then implemented in a ProtoBoard. Ultimately, the sensor performed better than the previous (TSSP4056) sensor and the team decided to use the PRP220. The sensor was able to detect changes between black and white as well as objects within 77cm which is the diameter of the sumo bot ring where the bots will battle.

Another design issue faced was creating an initial design that did not properly accommodate for the dimensions of the sumo bot plow (the metal bumper in the front of the bot). The plow would not fit with the original zumo reflectance sensor array. The sensor array impeded where the plow needed to be screwed into the chassis. Fortunately, the new IR sensor (PRP220) could theoretically replace the zumo reflectance sensor array and be easily implemented into the designed PCB as seen in Figure (3). The use of this sensor also eliminates the element (reflectance sensor array) impeding the plow from being properly connected.

Initially, a multiplexer was used to cycle through four different sensors to take their readings. The circuit designed in Figure (6) was used to test the multiplexer. The LEDs were used to show the status signals of the enable 1, input 1, input 2, enable 2, input 3, input 4 for the H-bridge. By creating this design, the LEDs would allow for a simple visual displaying the functionality of the circuit. Additionally, the circuit was used to test the ADC values of the sensor (TSSP4056). The MSP430 connected to the board read the values of the sensor, which needed to be converted to a digital signal that the MSP430 could handle. The values were then read from the MSP430 to confirm that the ADC values were being converted properly.

The external multiplexer was no longer needed upon discovering further functionality of the MSP430. There are multiple ADC pins on the MSP430, so the code to initialize and use these pins was discovered which removed the need of an external circuit element (multiplexer).

It was necessary to adapt to what was on hand when putting the robot together. This caused for many electronics challenges. One of which included an large amount of external wires and the necessity of a breadboard to accommodate for lacking circuit elements. In an effort to improve the aesthetics of the sumo bot, a housing was created in SolidWorks and 3d printed. The model can be seen in Figure (17).

3.1 Code

The first section of code in Figure (18) determines when certain flags will be called. If the sensor reading coming from the analog to digital converter(ADC) is greater than the sensors corresponding threshold a flag will be raised and the robot will act accordingly. The second section in Figure (18) Indicates the maximum speed or timer CCR value that controls the duty cycle that controls the motor, the turn speed and the duration the robot will turn when it sees a white line. The third section initializes a variable that the code can access and is used to interpret the values coming from the sensors through the ADC. The final section Initializes variables used to control the robot through UART.

The function in Figure (19) is used to check the value of each sensor and will return a number flag which shows which sensor is above the threshold. If none of the sensors are it will return a -1 indicating that the sensors view nothing significant. Each sensor has priority viewing the the two if statements at the top prioritize the two bottom sensors on the robot and will call their flags before checking the top sensors.

Figure (20) shows part of the the function used to control the motors. This function does this by setting pins on the MSP430 to high or low that are connected to the H-bridge on the PCB that controls the motors. The TA2CCR2 controls the duty cycles that drives the speed of the motors.

The block of code in Figure (21) is found inside of the UART interrupt.

it takes in a value from the RX bus and sets the direction that the bot will move accordingly.

This block of code in Figure(22) is found inside of the ADC interrupt. It sets the values inside of a fixed size array. The values inside of this array are be read in a different function.

4 Results and Conclusion

The sensors needed to be tested to ensure proper performance inside the bot design. The Tina-Ti schematic of the designed circuit to test the sensors can be seen in Figure (8). A spare sumo bot was used to test the sensors. The output of the IR sensor(PRP220) was measured at varying distances from the sumo bot. The results are seen on the graph in Figure (9). The results show the successful detection of an object within 77cm (diameter of the sumo bot battle ring).

Testing the IR sensors allowed for calculation of threshold voltages. The sensors output a higher voltage the closer an object is to it. Relatively high threshold voltages were chosen for testing purposes. The voltage outputted from the sensors are converted to ADC values, which the program can use to behave accordingly. Once the ADC value reaches a certain threshold, the bot will react.

The circuit in Figure (11) was built to test the H-bridge while simultaneously testing the motors. The H-bridge was seen to be successfully working by being able to successfully switch the direction of the motors. This can be seen in the video attached to the Github repository. The video demonstrates the wheels turning a different direction due to the H-bridges ability to reverse a the voltage polarity. Additionally, it shows that the motors are properly working.

The H-bridge is controlled by four inputs which control two motors. A truth table made for the the inputs displays how the inputs control the H-bridge functionality seen in Figure (12). The values of the inputs were adjusted to properly control the motor's direction. By experimenting with the

H-bridge, the movement of the sumo bot can be controlled by the behavior of the H-bridge.

The sumobot was able to reach the group's expectations. The H-bridge used performed very well and was easily able to control both of the motors. The IR sensor module used however was not as sensitive as desired, and as a result the sumobot was only able to detect objects up to approximately three inches in front of it. This could be fixed either with better calibration of the sumobot or more sensitive sensors. UART transmission to the robot worked flawlessly, and opens up space for more communication possibilities in the future such as bluetooth or other forms of wireless communication. The battery configuration we used did not perform very steadily, with sudden drainages crippling the motors, so other options such as rechargeable batteries is a strong consideration in future projects. Overall, the sumobot was a success but there is still much room for improvement.

5 Appendix

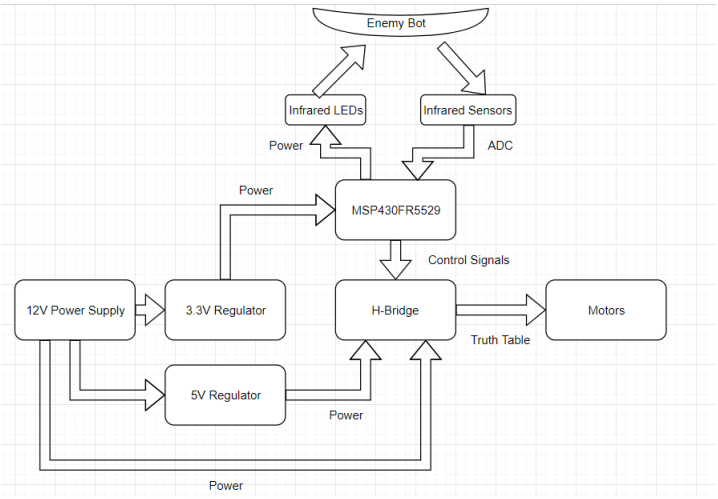

Figure 1: Diagram of inputs and outputs to the sumobot.

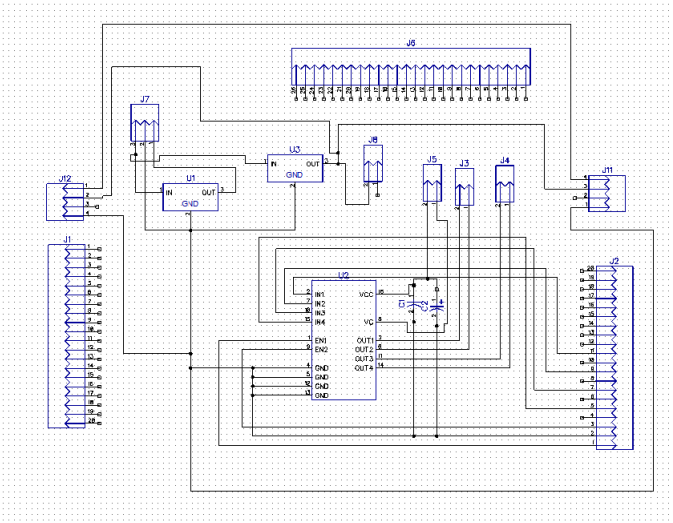

Figure 2: Schematic of breakout board used to power components.

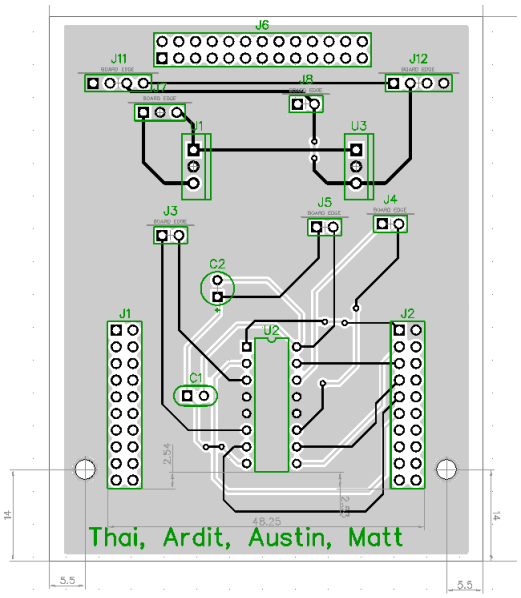

Figure 3: Last version of PCB design.

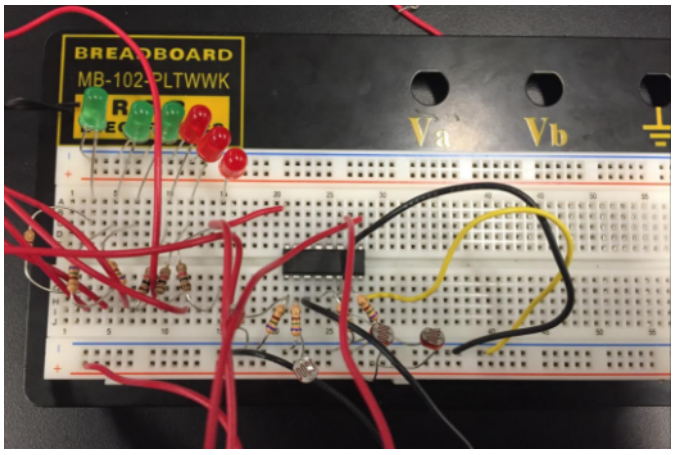

Figure 4: Breadboard picture of LEDs used for testing the onboard ADC and photoresistors.

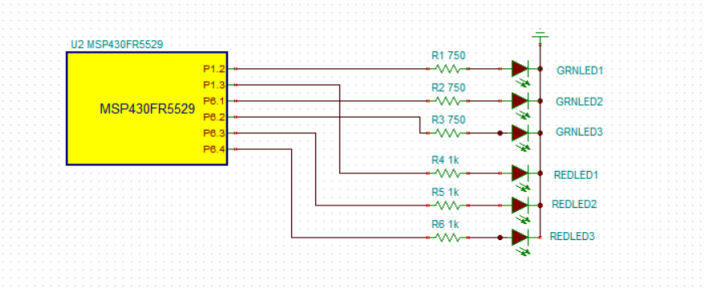

Figure 5: Schematic of LEDs used for testing the onboard ADC.

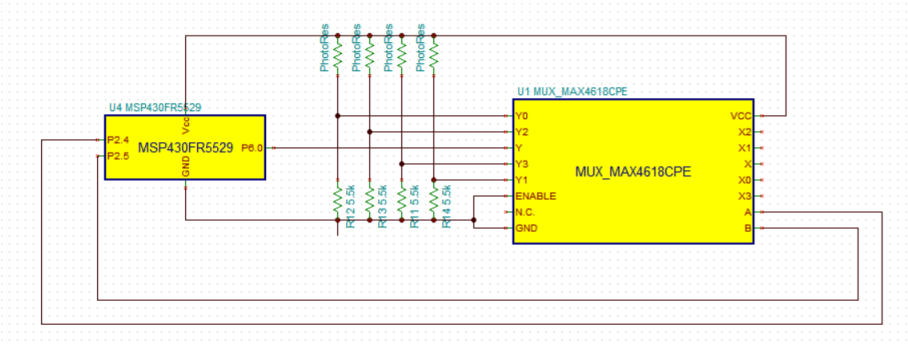

Figure 6: Schematic of Photoresistors used to simulate IR LEDs and IR sensors with MUX.

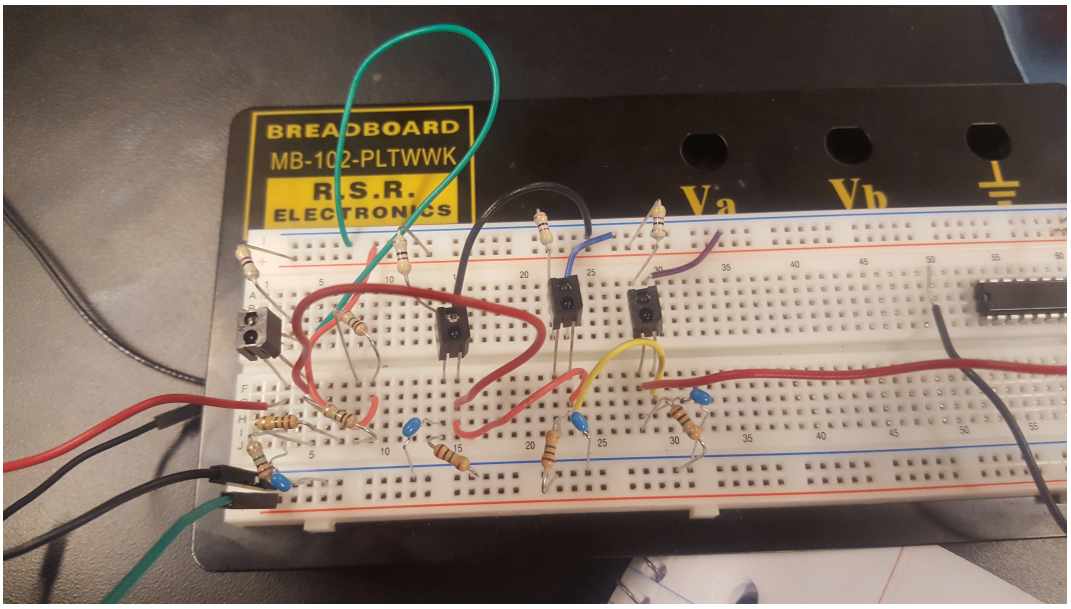

Figure 7: Picture of breadboard test setup with IR sensors.

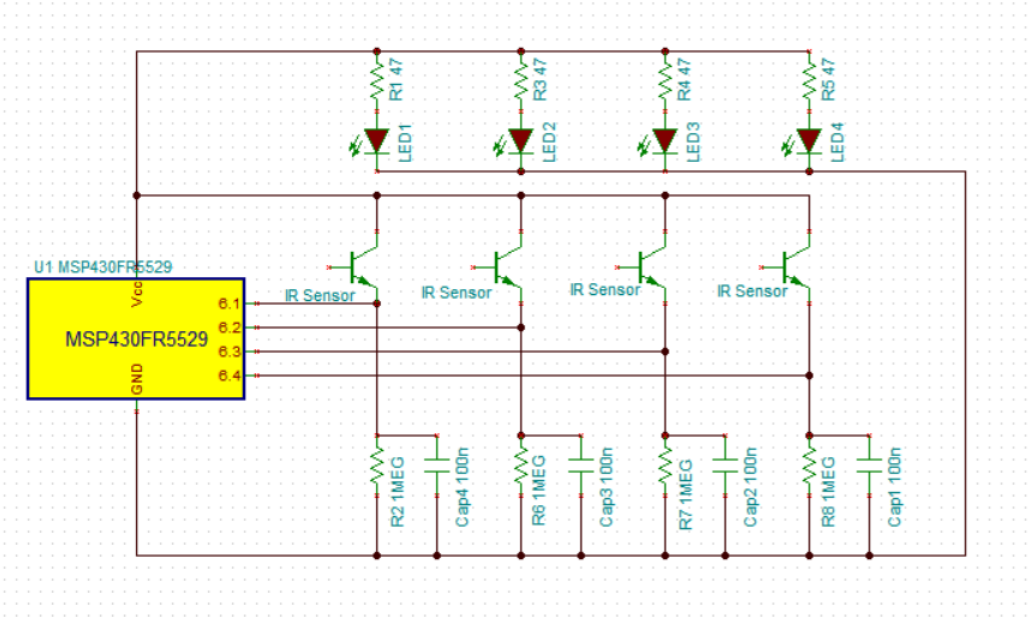

Figure 8: Schematic of breadboard test setup with IR sensors.

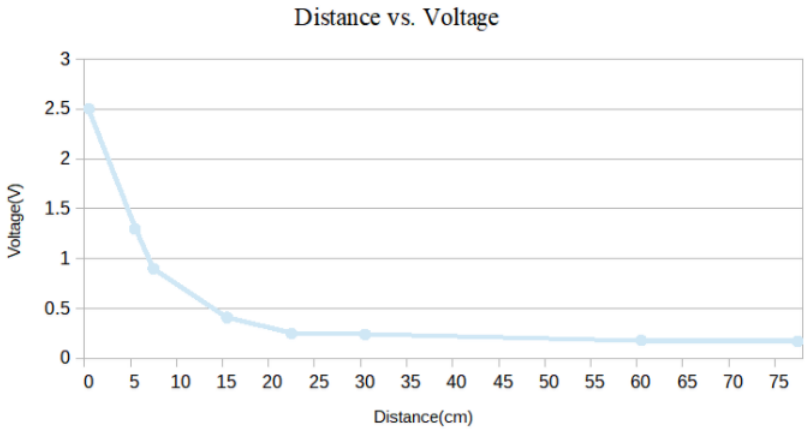

Figure 9: Graph of distance vs. voltage use to calibrate IR sensors.

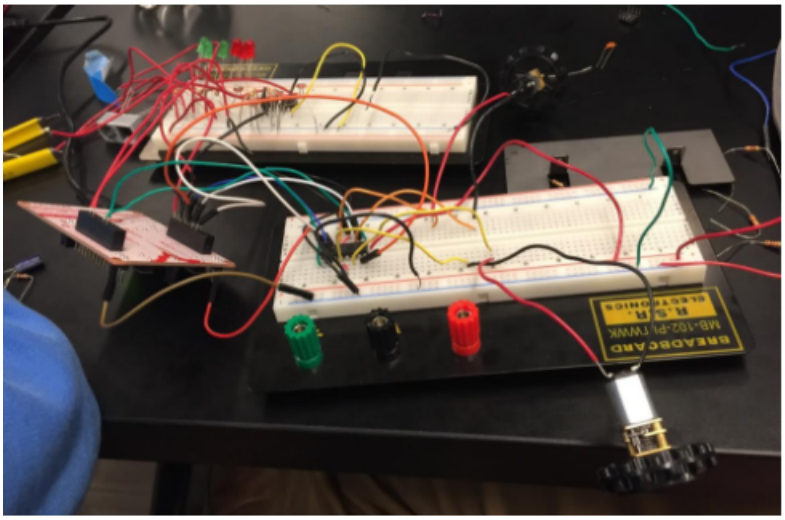

Figure 10: Picture of breadboard test setup of H-bridge with motors.

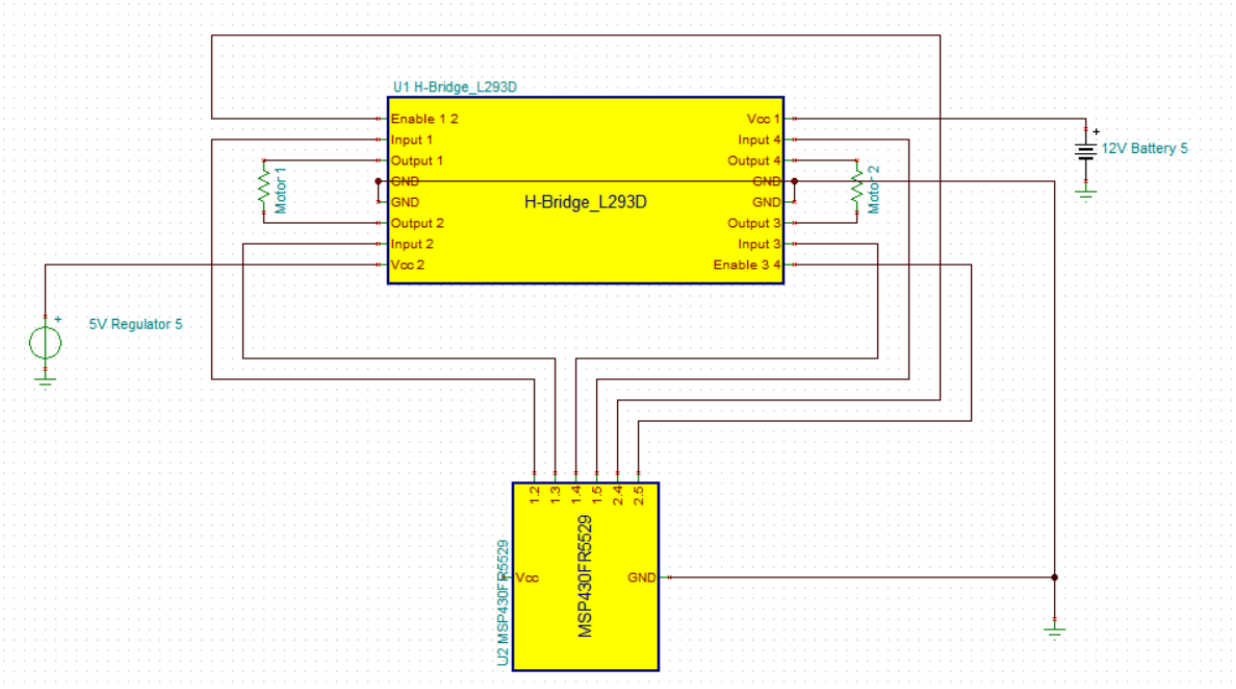

Figure 11: Schematic of breadboard test setup of H-bridge with motors.

Truth Table

A	B	Description
0	0	Motor stops
0	1	Motor runs clockwise
1	0	Motor runs anti-clockwise
1	1	Motor stops

Figure 12: Truth Table of H-Bridge.

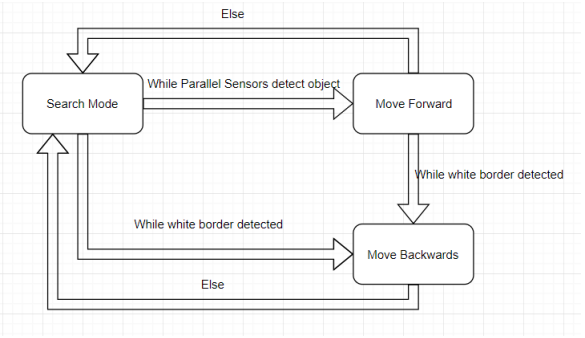

Figure 13: Graphical diagram of code flow.

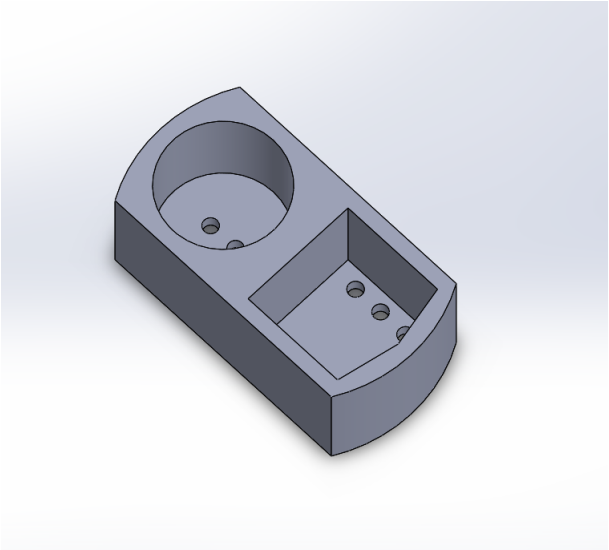

Figure 14: IR Casing.

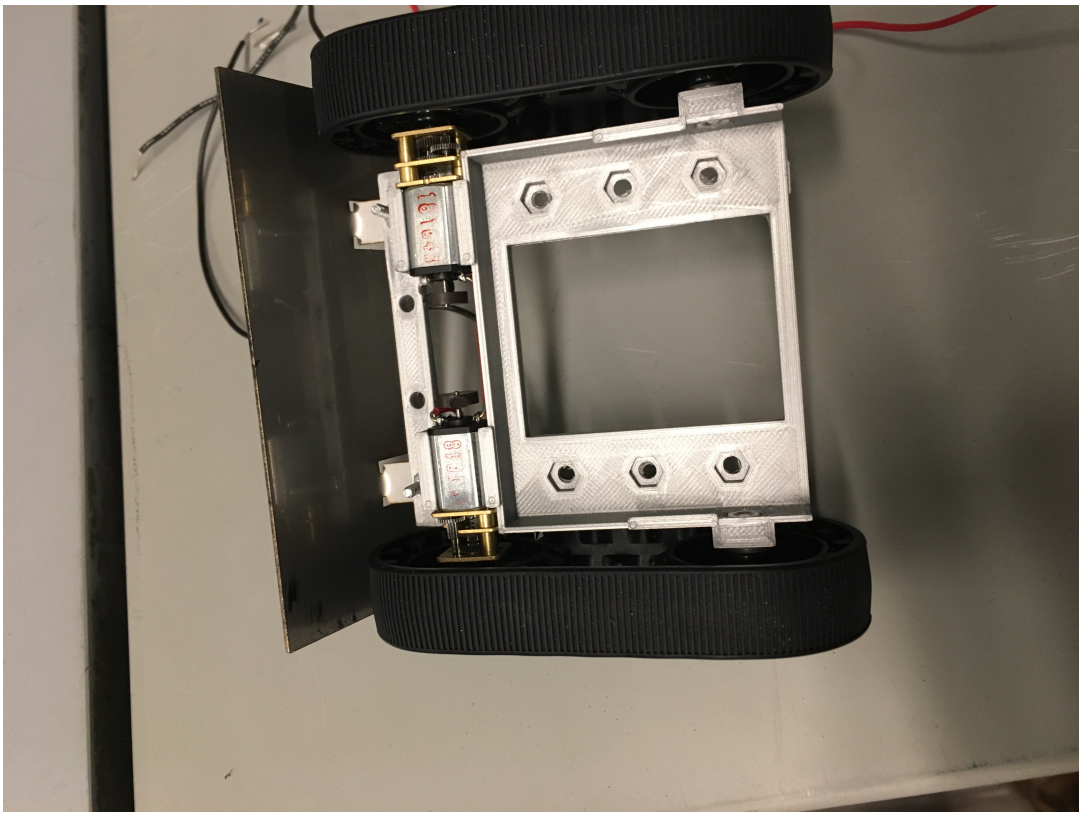

Figure 15: 3d printed chassis.

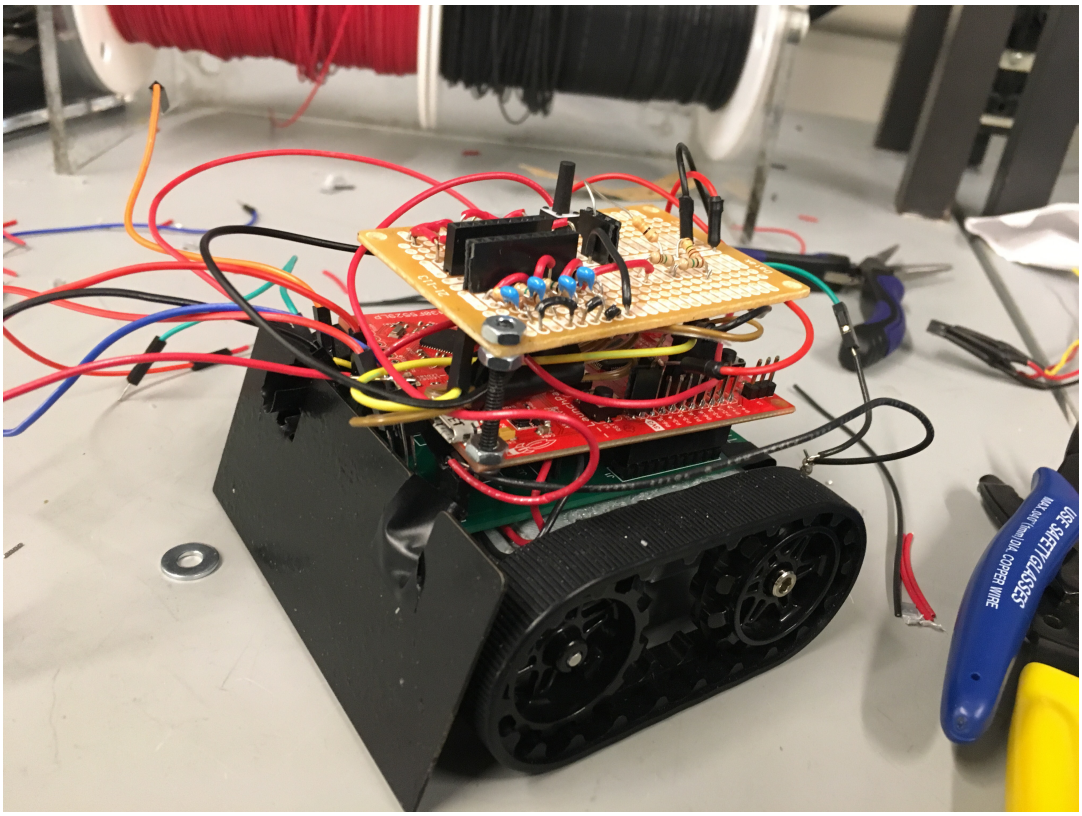

Figure 16: Sumobot without helmet.

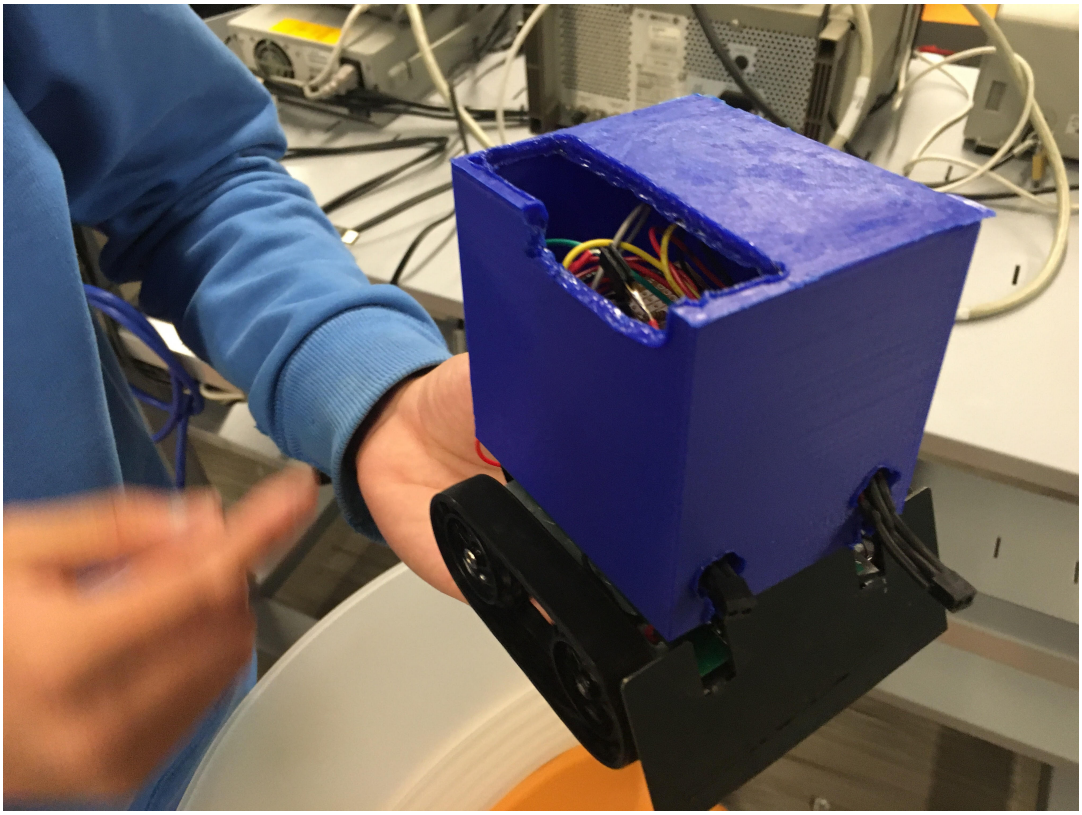

Figure 17: Sumobot with helmet.

```
4 //Initialize Sensor Thresholds.
5 const int PHOTORESISTOR_THRESH1 = 200;
6 const int PHOTORESISTOR_THRESH2 = 200;
7 const int IRSENSOR_THRESH1 = 40;
8 const int IRSENSOR_THRESH2 = 40;
9 int sensorThreshHolds[4] = {PHOTORESISTOR_THRESH1, PHOTORESISTOR_THRESH2, IRSENSOR_THRESH1, IRSENSOR_THRESH2};
10
11 //Initialize Speed constants and turn duration.
12 int MAX = 999;
13 int TURN_SPEED = 999;
14 unsigned int TURN_TIME = 500000;
15 int attackSpeedLeft, attackSpeedRight;
16
17 //Initialize variable that the program can read to see sensor values.
18 int adc_value[4] = {0,0,0,0};
19
20 //Define button and variables for UART remote control mode.
21 #define BUTTON BIT0;
22 int fightMode = -1;
23 int direction = 0;
```

Figure 18: Initializes thresholds and other important variables.

```
// Check the status of all 4 sensors
int sensorCheck()
{
    if(readSensor(0) > sensorThreshHolds[0])
    {
        return 0;
    }
    else if(readSensor(1) > sensorThreshHolds[1])
    {
        return 1;
    }
    else if(readSensor(2) > sensorThreshHolds[2])
    {
        return 2;
    }
    else if(readSensor(3) > sensorThreshHolds[3])
    {
        return 3;
    }

    return -1;
}
```

Figure 19: Checks sensor values vs thresholds.

```
void motorPWM(int left, int right)
//'left' and 'right' are the input speed
{
    if(left < 0) //backwards, clock-wise
    {
        P1OUT &= ~BIT4;
        P1OUT |= BIT5; // Sets direction for H-bridge
        TA2CCR2 = (left * -1);//Speed of motor CCR/1000
    }
    else if(left == 0) //stopping
    {
        P1OUT &= ~BIT4;
        P1OUT &= ~BIT5;// Sets direction for H-bridge
        TA2CCR2 = 0;//Speed of motor CCR/1000
    }
    else // forwards, counter clock-wise
    {
        P1OUT |= BIT4;
        P1OUT &= ~BIT5;// Sets direction for H-bridge
        TA2CCR2 = left;//Speed of motor CCR/1000
    }

```

Figure 20: Sets directions for H-Bridge and controls PWM.

```
341     while (!(UCA0IFG&UCTXIFG)); // USCI_A0 TX buffer ready?
342     if(UCA0RXBUF == 119)//w
343     {
344         direction = 1;
345         UCA0TXBUF = UCA0RXBUF;
346     }
347     else if(UCA0RXBUF == 115)//s
348     {
349         direction = 2;
350         UCA0TXBUF = UCA0RXBUF;
351     }
352     else if(UCA0RXBUF == 100)//a
353     {
354         direction = 3;
355         UCA0TXBUF = UCA0RXBUF;
356     }
357     else if(UCA0RXBUF == 97)//d
358     {
359         direction = 4;
360         UCA0TXBUF = UCA0RXBUF;
361     }
```

Figure 21: Code controlling UART transmission.

```
401         adc_value[0] = ADC12MEM1; //changes duty cycle
402         adc_value[1] = ADC12MEM2; //changes duty cycle
403         adc_value[2] = ADC12MEM3; //changes duty cycle
404         adc_value[3] = ADC12MEM4; //changes duty cycle
405         __bic_SR_register_on_exit(LPM0_bits);
406         break;                          // Vector 14:  ADC12IFG4
```

Figure 22: ADC values being saved into a fixed size array.