Navid Shahrestani

BridgeBot

EEL 5666 – Intelligent Machine Design Laboratory
Dr. Antonio Arroyo
Dr. Eric M. Schwartz
Ryan Stevens
Tim Martin

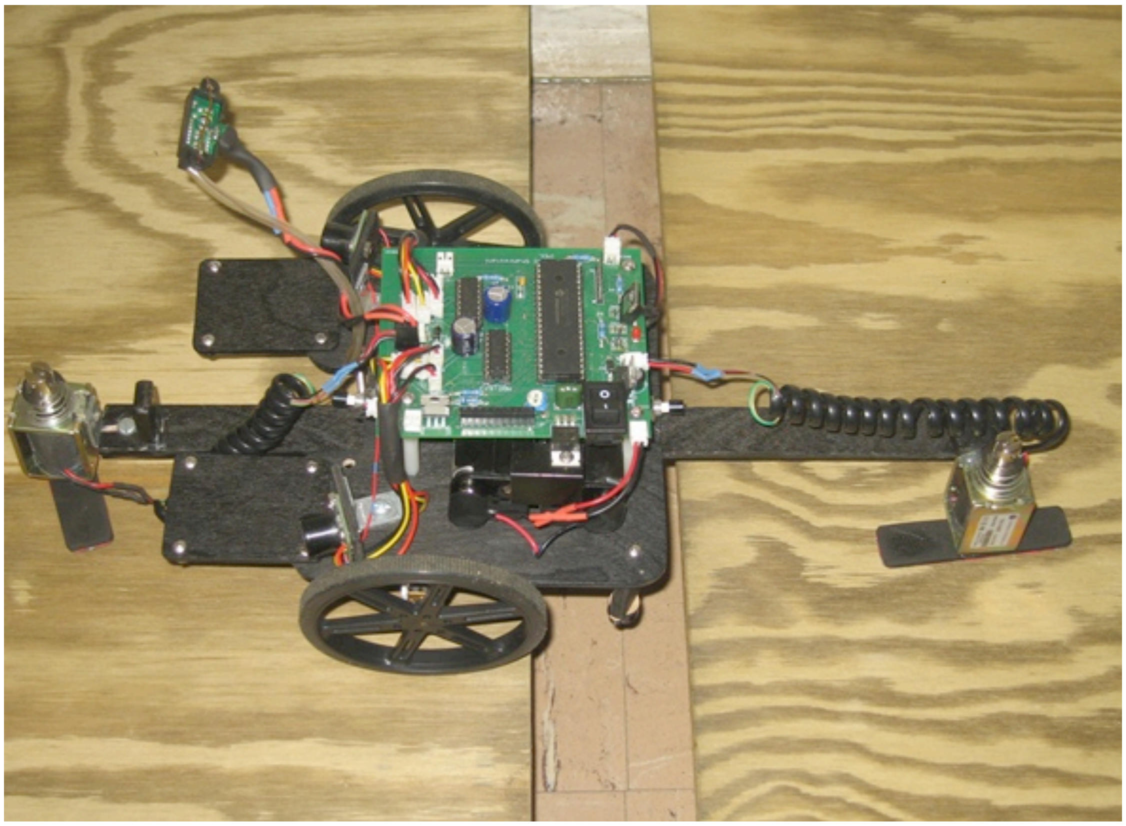


Table of Contents

Abstract

The BridgeBot's primary task is to navigate its world and avoid obstacles. When it encounters the end of the surface it's traveling on (most likely the edge of a table in this case) it will search for another surface nearby. If it encounters one, it will extend its bridge mechanism and lift the robot off the ground, allowing it to traverse the bridge across a rack and pinion system. Once over, it will continue its exploration of the surface.

BridgeBot will be controlled by a PIC processor on a custom designed PCB. It will use a variety of sensors (IR, sonar, bump) to traverse its world safely, and will have several actuators (DC motors, solenoids) to aid in its navigation and bridge gapping. It will use a rack and pinion system to allow the robot to traverse the bridge linearly across the gap, hopefully reaching the other side safely.

Executive Summary

Bridgebot is an exploration robot capable of bridging gaps that it comes across. When first turned on, he will explore his environment, avoiding obstacles. If he finds and edge to the surface he is traveling on, he will line up with the edge and search for another surface nearby. If none is to be found, he will turn around and keep exploring. If there is a surface to cross to, then he will begin his crossing behavior.

A bridge will extend from under the base, using a rack and pinion gear system. It will extend fully, and then deploy two solenoid, one attached to each end of the bridge, lifting the main body of off the ground. Bridgebot will then use the same bridge mechanism, except now supported at each end, to cross the base from one platform to the other. Once safely across, he will lower himself, adjust the bridge position accordingly, and continue on his voyage.

Two IR sensors are used for "end of the world" detection, while two sonars are used for obstacle avoidance. A high torque DC motor runs the rack and pinion gear, with limit switches at each end of the mechanism, to allow for feedback of the mechanism. A third IR is used for detecting a possible surface to bridge to.The solenoids are driven by a pair of power transistors, and a hefty 24 volt battery pack was used to allow for enough force to disengage the wheels from the surface.

Introduction

The idea for BridgeBot's unique capabilities came from the limitations of other, more advanced robots. If you consider a Mars rover, a robot that has to navigate rough, unknown terrain, one can see a big advantage to having the capabilities of crossing gaps that the rover might encounter. Hence the BridgeBot is a scaled down prototype of this behavior. Having the ability to not have to completely avoid narrow gaps would greatly improve the efficiency of such vehicles, and allow for easier exploration of unknown terrain.

The BridgeBot will be responsible for safely navigating its environment. If it encounters a "cliff" it will approach the edge and search for another surface within its range. If it locates one it will cross the gap successfully. The rest of the report will guide you through the brains behind the actions, namely the PIC microcontroller and the custom PCB it is on. It will also go over how obstacle avoidance is achieved and what sensors where used. Lastly it will go over probably the most critical portion of the design, and the portion that warrants the most attention when designing; the bridge gapping mechanism.

Integrated System

The PIC 18F4423 will take environmental sensor readings from three IR sensors, two sonar sensors, two bump sensors and two limit switches. The sonar sensors will provide the main feedback for obstacle avoidance. Two of the IR sensors will be placed facing downwards to detect when an edge has been found, while the third will be used to locate another nearby surface. The bump sensors are used as a last resort for obstacle detection and the limit switches are placed on either end of the bridge mechanism to allow the PIC to know when the bridge has reached either end of its linear travel.

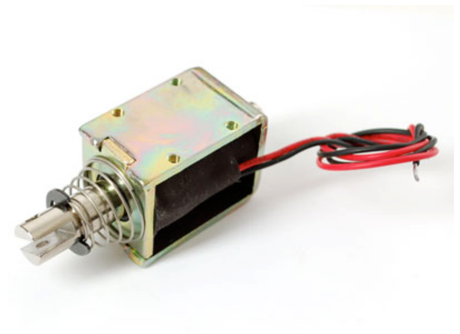
Actuation will be achieved by DC motors and solenoids. Two motors will be in charge of the drive wheels and the third will be used to extend and retract the bridge mechanism. The solenoids will be used to elevate the robot off the ground when it is necessary to cross a gap. The user will also receive feedback about the state of the robot from an LCD screen placed on the side of the base.

Mobile Platform

Although the final design has not been completed, the BridgeBot will most likely have a base cutout from the T-tech machine. The intricacies arise when dealing with the bridge mechanism. Much care must be taken to make the mechanism not only structurally sound, but lightweight to minimize tipping of the robot. A nylon rack and pinion gear will be integrated in the mechanism, most likely on some sort of I-beam structure to allow for rigidity in the bridge. It will then be attached to the base in a way that allows for linearly motion while keeping it fixed to the base. An early concept idea is to use the bridge mechanism not only to cross a gap, but as the mounting surface for the back caster of the robot.

Actuation

The drive wheels chosen were Pololu Micro Metal motors (Fig1) due to their small size and the fact that they are capable of the torque required for such a relatively lightweight robot. The motor chosen for the linear motion across the bridge is the same motor expect that it is geared down more than the drive wheels to provide more torque since moving across the rack and pinion is going to require more torque than simply driving the robot around.



The motors are driven by a L293 H-bridge and powered by a 12V RC car battery.

The real power hogs in this design however are the solenoids (Fig2) used to lift the base off the surface. They have a 10mm throw and provide a 15 N starting force. They will be powered through a TIP120 Darlington pair transistor capable of sinking the amount of current that the solenoids will use when turned on.

Sensors

For obstacle avoidance the Maxbotix LV-EZ0 (Fig3) sonar range finders were used. They work by sending out an ultrasonic wave and measuring the time it takes for the wave to propagate back to the receiver after bouncing off a surface. While not the most accurate at absolute distance they were chosen because they had a large beam width. This allowed for a wider viewing angle which in turn means that the sensor can see obstacles in a much wider field of view than other sensors.

Figure 3

The infrared sensor chosen was the Sharp GP2D120XJ00F (Fig4) due to its accuracy at close range. It works on a very similar principle to the sonar, except that instead of an ultrasonic sound wave, it emits an infrared beam that is detected by a photodiode when it bounces off a surface. It is the perfect sensor for sensing if another surface can be found when the BridgeBot searches for another surface to bridge to, and is mounted in a way to eliminate the blind spot at distances less than 4cm, which is inherent in most sensors of this type.

Figure 4

Behaviors

The BridgeBot will have three basic behaviors: search/obstacle avoid, locate surface when gap is found, and crossing the gap. For specifics, refer to code in Documentation section.

Experimental Layout and Results

The bridge mechanism when through several revisions before it's final design. I was worried about two main things; transferring rotational motion to linear motion, and having enough force to lift the robot off the ground.

The initial design used servos and a cam system to lift the robot off the ground using mechanical advantage. But the final solenoid design was chosen because of it's ease of implementation and cost restraints. This did have the unwanted side effect of adding more weight to the design by needing more batteries.

The rack and pinion was chosen over something like a linear actuator mainly for financial reasons. But after carefully getting the proper design in Solidworks for the rack mount and through careful testing, I am confident that the final solution of having a groove where the rack mounts is a better option.

Conclusion

Overall, I would consider this project a success. I was able to deliver on the proposed goal of the robot. All of the behaviors I had originally intended for this project were fulfilled and I am quite please.

If I could re-do this project I would redesign the bridge mechanism only on the fact that I wasn't able to get the range I had wanted on it due to the limited size of the cutting are on the t-tech. I would also find stronger solenoids, because to engage them enough I had to add extra batteries that I had originally not designed for.

Documentation

```
#include "C:\Users\Navid\Dropbox\IMDL\Source Code\IMDL.h"
//#int_TIMER1

void  TIMER1_isr(void)
{

                //          output_toggle(PIN_E1);

}
void main()
{

   setup_adc_ports(AN0_TO_AN4|VSS_VDD);
   setup_adc(ADC_CLOCK_DIV_2|ADC_TAD_MUL_2);
   setup_spi(SPI_SS_DISABLED);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
   setup_timer_2(T2_DIV_BY_1,MAX_SPEED,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   setup_ccp1(CCP_PWM);
   setup_ccp2(CCP_PWM);
   set_pwm1_duty(0);
   set_pwm2_duty(0);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   setup_uart(FALSE);
   //enable_interrupts(INT_TIMER1);
   //enable_interrupts(GLOBAL);
   setup_oscillator(OSC_8MHZ);

        delay_ms(500);
        init_motors();
        delay_ms(500);
//      calibrate_sensors();
        lcd_init();
        if (input(BACK_STOP) ==0)
                calibrate_bridge_position();

  go_straight();


int i;
  while(1)
        {
        //      lcd_char("A");
          sync_sonars();
          check_obstacles();

                if ((right_obstacle == 1) || (left_obstacle == 1))
                                avoid_obstacle();

                if ((right_edge == 1) || (left_edge == 1))
                                {
                                        align();
                                        delay_ms(1000);
                                        check_gap();
                                        if (bridge == 1)
                                        {
                                                turn_around();
                                                go_straight();


                                        }
                                        else
                                        {
                                                extend_bridge();
                                                cross();
                                                go_straight();
                                        }
                                }
                delay_ms(100);
        }
}
/*********Traverse
Gap*************************************************************/
void cross (void)
{
        lift();

        delay_ms(3000);

        while(input(FRONT_STOP) != 0)
		{
			motor_set(MOTOR3,MAX_SPEED,BACK);
		}
	output_low(BACK_SOLENOID);
	output_low(FRONT_SOLENOID);
	delay_ms(500);

	motor_set(MOTOR3,MAX_SPEED,FORWARD);
	//even loaded
	delay_ms(5000);
	motor_set(MOTOR3,MAX_SPEED,STOP);

/*	while(input(BACK_STOP) != 0)
		{
			motor_set(MOTOR3,MAX_SPEED,FORWARD);
		}
			back loaded
	motor_set(MOTOR3,MAX_SPEED,BACK);
	delay_ms(1000);
	motor_set(MOTOR3,MAX_SPEED,STOP);*/

}
/*********Deploy
Solenoids***********************************************************/
void lift (void)
{

		output_high(FRONT_SOLENOID);
		delay_ms(100);
		output_low(FRONT_SOLENOID);
		delay_ms(100);
		output_high(FRONT_SOLENOID);

		delay_ms(500);

		output_high(BACK_SOLENOID);
		delay_ms(100);
		output_low(BACK_SOLENOID);
		delay_ms(100);
		output_high(BACK_SOLENOID);

}
/**********Move bridge to desired position using
stops*******************************/
void calibrate_bridge_position (void)
{
	motor_set(MOTOR1,MAX_SPEED,STOP);
	while((input(FRONT_STOP) != 0) || (input(BACK_STOP) != 0))
	{
		if (input(FRONT_STOP) == 0)
			motor_set(MOTOR3,MAX_SPEED,FORWARD);
		else if (input(BACK_STOP) == 0)
			motor_set(MOTOR3,MAX_SPEED,BACK);
		else
			motor_set(MOTOR3,MAX_SPEED,STOP);
	}
}
/************Extend Bridge
forward*************************************************/
void extend_bridge (void)
{
	while(input(BACK_STOP) != 0)
		{
			motor_set(MOTOR3,MAX_SPEED,FORWARD);
		}
	motor_set(MOTOR3,MAX_SPEED,STOP);
}
/************Calibrate sensor
thresholds**********************************************/
void calibrate_sensors (void)
{

}
/************Check if gap is crossable
**********************************************/
void check_gap (void)
{
	read_adc_chan(FRONT_IR);
	if (adc_reading < 500)
		{
			bridge = 1;
		}
	else
		{
			bridge = 0;
		}
}
/***********Sync pulse for
sonars**************************************************/
void sync_sonars (void)
{
	output_low(SONAR_SYNC);
	output_high(SONAR_SYNC);
	delay_us(10);
```

```
        output_low(SONAR_SYNC);
}

/*********180
turn*****************************************************************/
void turn_around (void)
{

        motor_set(MOTOR2,MAX_SPEED,BACK);
        motor_set(MOTOR1,MAX_SPEED,BACK);
        delay_ms(800);

        motor_set(MOTOR2,MAX_SPEED,FORWARD);
        motor_set(MOTOR1,MAX_SPEED,BACK);
        delay_ms(2000);
}

/********Align with edge of
surface*************************************************/
void align (void)
{
        motor_set(MOTOR2,MAX_SPEED,STOP);
        motor_set(MOTOR1,MAX_SPEED,STOP);
        if (left_edge == 1)
        {
                        motor_set(MOTOR2,MAX_SPEED,FORWARD);
                        read_adc_chan(RIGHT_IR);
                        if (right_edge ==1)
                                {
                                        motor_set(MOTOR1,MAX_SPEED,FORWARD);
                                        motor_set(MOTOR2,MAX_SPEED,FORWARD);
                                        delay_ms(250);
                                        motor_set(MOTOR1,MAX_SPEED,STOP);
                                        motor_set(MOTOR2,MAX_SPEED,STOP);
                                }
                        while (right_edge == 0)
                        {
                                read_adc_chan(RIGHT_IR);
                                if (adc_reading < RIGHT_IR_THRESHOLD)
                                        right_edge = 1;
                        }
                        delay_ms(200);
                motor_set(MOTOR2,MAX_SPEED,STOP);
                        return;
        }

        if (right_edge == 1)
        {
                        motor_set(MOTOR1,MAX_SPEED,FORWARD);
                        read_adc_chan(LEFT_IR);
                        if (left_edge ==1)
                                {
                                        motor_set(MOTOR1,MAX_SPEED,FORWARD);
                                        motor_set(MOTOR2,MAX_SPEED,FORWARD);
                                        delay_ms(250);
                                        motor_set(MOTOR1,MAX_SPEED,STOP);
                                        motor_set(MOTOR2,MAX_SPEED,STOP);
                                }
                        while (left_edge == 0)
                        {
                                read_adc_chan(LEFT_IR);
                                if (adc_reading < LEFT_IR_THRESHOLD)
                                        left_edge = 1;

                        }
                        delay_ms(200);
                motor_set(MOTOR1,MAX_SPEED,STOP);
        }
}

/************** Check sensor
values**********************************************/
void check_obstacles (void)
{
        left_obstacle = 0;
        right_obstacle = 0;
        left_edge = 0;
        right_edge = 0;

        read_adc_chan(LEFT_IR);
        if (adc_reading < LEFT_IR_THRESHOLD)
                {
                        motor_set(MOTOR2,MAX_SPEED,STOP);
                        motor_set(MOTOR1,MAX_SPEED,STOP);
                        left_edge = 1;
                }
        read_adc_chan(RIGHT_IR);
        if (adc_reading < RIGHT_IR_THRESHOLD)
                {
                        motor_set(MOTOR2,MAX_SPEED,STOP);
                        motor_set(MOTOR1,MAX_SPEED,STOP);
                        right_edge = 1;
                }
        read_adc_chan(RIGHT_SONAR);
        if (adc_reading <RIGHT_SONAR_THRESHOLD)
                {
                        motor_set(MOTOR2,MAX_SPEED,STOP);
          motor_set(MOTOR1,MAX_SPEED,STOP);
                        right_obstacle = 1;
                }
        read_adc_chan(LEFT_SONAR);
        if (adc_reading <LEFT_SONAR_THRESHOLD)
                {
                        motor_set(MOTOR2,MAX_SPEED,STOP);
             motor_set(MOTOR1,MAX_SPEED,STOP);
                        left_obstacle = 1;
                }

}

/********Turn away from
obstacle*************************************************/

void avoid_obstacle (void)
{
        if (left_obstacle == 1)
                {
                        left_obstacle = 0;
                        //motor_set(MOTOR2,MAX_SPEED,BACK);
                        //motor_set(MOTOR1,MAX_SPEED,BACK);
                        //delay_ms(300);
                        motor_set(MOTOR2,MAX_SPEED,BACK);
                        motor_set(MOTOR1,MAX_SPEED,STOP);
                        delay_ms(750);
                        motor_set(MOTOR2,MAX_SPEED,FORWARD);
                        motor_set(MOTOR1,MAX_SPEED,FORWARD);
                }

        else if (right_obstacle == 1)
                {

                        right_obstacle = 0;
                        //motor_set(MOTOR2,MAX_SPEED,BACK);
                        //motor_set(MOTOR1,MAX_SPEED,BACK);
                        //delay_ms(300);
                        motor_set(MOTOR2,MAX_SPEED,STOP);
                        motor_set(MOTOR1,MAX_SPEED,BACk);
                        delay_ms(750);
                        motor_set(MOTOR2,MAX_SPEED,FORWARD);
                        motor_set(MOTOR1,MAX_SPEED,FORWARD);
                }
```

```
}

/**********************Drive
forward********************************************************/
void go_straight (void)
{
        motor_set(MOTOR2,MAX_SPEED,FORWARD);
        motor_set(MOTOR1,MAX_SPEED,FORWARD);
}

/*********************Make sure all motors are
off*******************************************/
void init_motors (void)
{
   output_low(MOTOR1_POS);
        output_low(MOTOR1_POS);
        output_low(MOTOR2_NEG);
   output_low(MOTOR3_POS);
        output_low(MOTOR3_NEG);
        output_low(MOTOR4_POS);
   output_low(MOTOR4_NEG);
        delay_ms(500);

        set_pwm1_duty(0);
        set_pwm2_duty(0);
        delay_ms(500);
}
/********************Change speed and/or direction of
motors*******************************/
void motor_set (char motor, int speed, char direction)
{


        switch(motor)
        {
                case MOTOR1 :
                                                switch(direction)
                                                {
                                                        case BACK              :
output_high(MOTOR1_POS);

output_low(MOTOR1_NEG);
                                                                                        break;


                                                        case FORWARD         :
output_low(MOTOR1_POS);
```

```
output_high(MOTOR1_NEG);
                                                                                    break;
                                            case STOP               :
output_low(MOTOR1_POS);

output_low(MOTOR1_NEG);
                                                                                    break;
                                    }
                                    set_pwm1_duty(speed);
                                    break;

                case MOTOR2 :
                                    switch(direction)
                                    {
                                            case BACK               :
output_high(MOTOR2_POS);

output_low(MOTOR2_NEG);
                                                                                    break;

                                            case FORWARD            :
output_low(MOTOR2_POS);

output_high(MOTOR2_NEG);
                                                                                    break;
                                            case STOP               :
output_low(MOTOR2_POS);

output_low(MOTOR2_NEG);
                                                                                    break;
                                    }
                                            set_pwm2_duty(speed);
                                            break;

                case MOTOR3 :
                                    switch(direction)
                                    {
                                            case BACK               :
output_high(MOTOR3_POS);

output_low(MOTOR3_NEG);
                                                                                    break;
                                            case FORWARD            :
output_low(MOTOR3_POS);

output_high(MOTOR3_NEG);
```

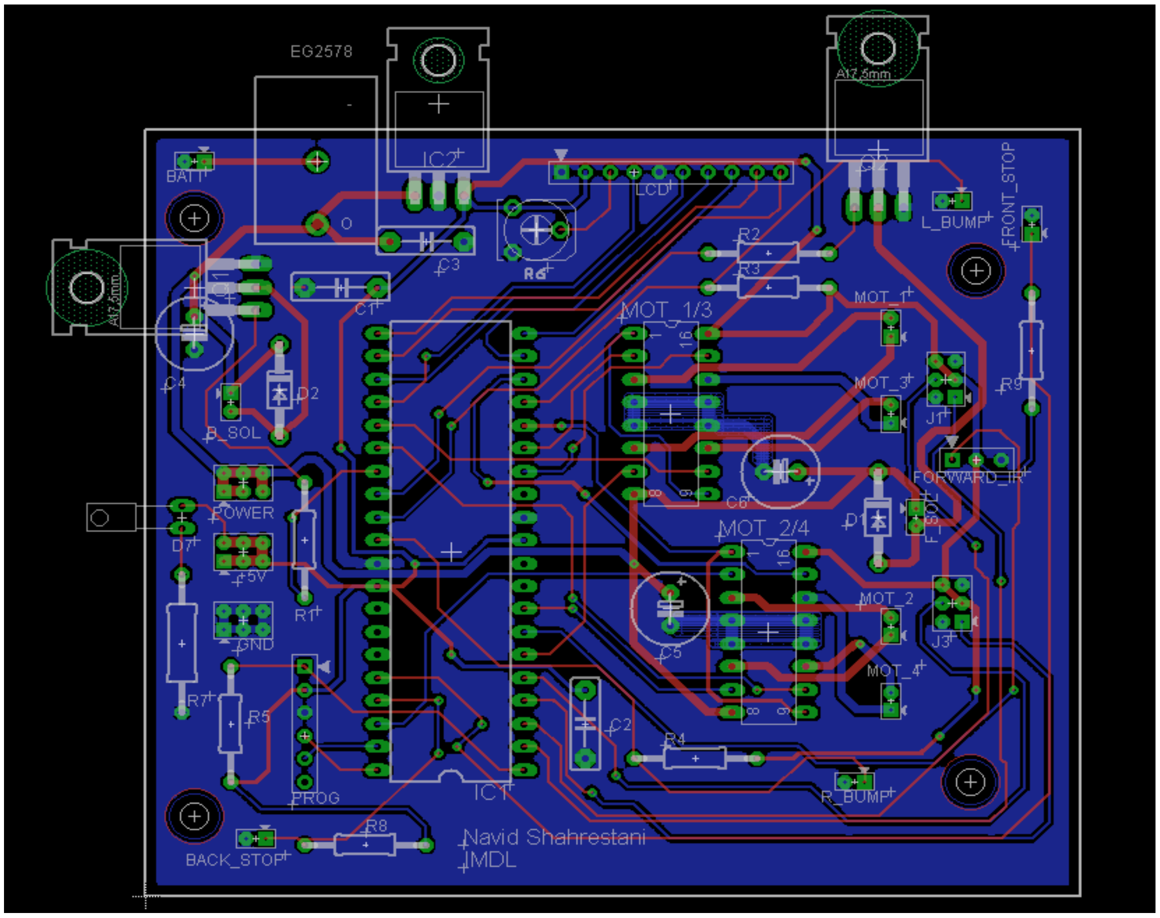
```
                                                                                    break;
                                                        case STOP                 :
output_low(MOTOR3_POS);

output_low(MOTOR3_NEG);
                                                                                    break;
                                                }
                                                        set_pwm1_duty(speed);

                                                        break;

                        case MOTOR4 :
                                                switch(direction)
                                                {
                                                        case FORWARD        :
output_high(MOTOR4_POS);

output_low(MOTOR4_NEG);
                                                                              break;


                                                        case BACK                 :
output_low(MOTOR4_POS);

output_high(MOTOR4_NEG);
                                                                                break;
                                                        case STOP                 :
output_low(MOTOR4_POS);

output_low(MOTOR4_NEG);
                                                                                    break;
                                                }

                                                        set_pwm2_duty(speed);
                                                        break;
      }
}

/*****************************************************************************
***********/

void read_adc_chan (char channel)
{
  set_adc_channel(channel);
  delay_us(50);
  adc_reading = (read_adc()>>4);
```

```
}

/*******************************************************************/
void lcd_command (char command)
{
                char temp;
                temp = command>>4;
                command = command && 0x0F;

                output_low(LCD_REG_SEL);
                output_low(LCD_EN);

                output_d(temp);
                delay_ms(1);
                output_high(LCD_EN);
                delay_ms(1);
                output_low(LCD_EN);

                delay_ms(2);

                output_d(command);
                delay_ms(1);
                output_high(LCD_EN);
                delay_ms(1);
                output_low(LCD_EN);

}

void lcd_char (char command)
{
                char temp;
                temp = command>>4;
                command = command && 0x0F;

                output_high(LCD_REG_SEL);
                output_low(LCD_EN);

                output_d(temp);
                delay_ms(1);
                output_high(LCD_EN);
                delay_ms(1);
                output_low(LCD_EN);

                delay_ms(2);

                output_d(command);
                delay_ms(1);
```

```
                    output_high(LCD_EN);
                    delay_ms(1);
                    output_low(LCD_EN);

}
void lcd_init ()
{
                    lcd_command(0x33);
                    lcd_command(0x32);
                    lcd_command(0x2C);
                    lcd_command(0x0F);
                    lcd_command(0x01);

                    delay_ms(100);

}
```

Appendices

- Board Layout


EG2578
IC2
LCD
BATT
R6
C3
C1
A17.5mm
L_BUMP
FRONT_STOP
R2
R3
MOT_1/3
MOT_1
MOT_3
R9
J1
FORWARD_IR
C4
D2
B_SOL
POWER
D7
+5V
GND
R1
R7
R5
PROG
IC1
C6
D1
F_SOL
MOT_2/4
MOT_2
J3
MOT_4
C5
C2
R4
R_BUMP
R8
BACK_STOP
Navid Shahrestani
IMDL