Final Report

# 

Team 203 PB&J

Pierce Simpson

Brandon Libjanic

Abigail Francis

Jonathan Hawkins

December 9, 2015

Table of Contents

[Overview 3](#_Toc437076118)

[Design Summary 3](#_Toc437076119)

[Hardware Design 4](#_Toc437076120)

[Hardware 4](#_Toc437076121)

[Additional Part description 8](#_Toc437076122)

[Pin Assignments and Pin Usage Descriptions 9](#_Toc437076123)

[Circuit Diagrams 11](#_Toc437076124)

[Software Design 16](#_Toc437076125)

[Overview 16](#_Toc437076126)

[Block Diagram 17](#_Toc437076127)

[Function Descriptions 18](#_Toc437076128)

[Ultrasonic Distance Sensor 20](#_Toc437076129)

[Task 1 20](#_Toc437076130)

[Task 2 20](#_Toc437076131)

[Task 3 20](#_Toc437076132)

# Overview

### Design Summary

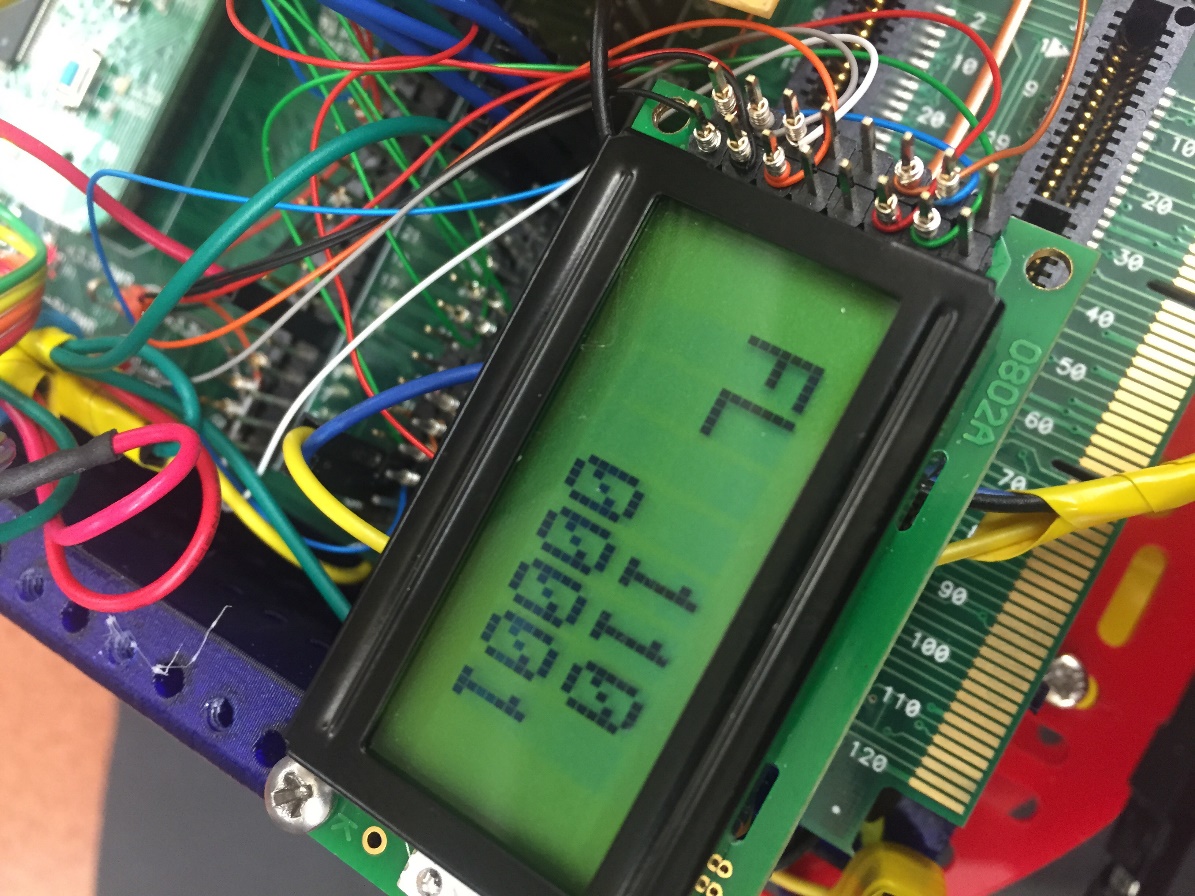
This robot follows a black line track, analyzes intersection nodes, and avoids obstacles that are in its path. The design consists of two independent motors that adjust speed and direction in order to follow the track. Eight infrared transmitters and receivers are mounted in a circular formation in order to detect the dark track. The circular formation allows for node analysis in order to determine the location of the robot on the track. When the robot detects an object on the front it will move off the track and then find the track again once the obstacle has been passed. Objects are detected using an ultrasonic sensor which determines the distance of an object in front of the robot. As that distance decreases the robot begins the sequence that will allow it to avoid the object.

# Hardware Design

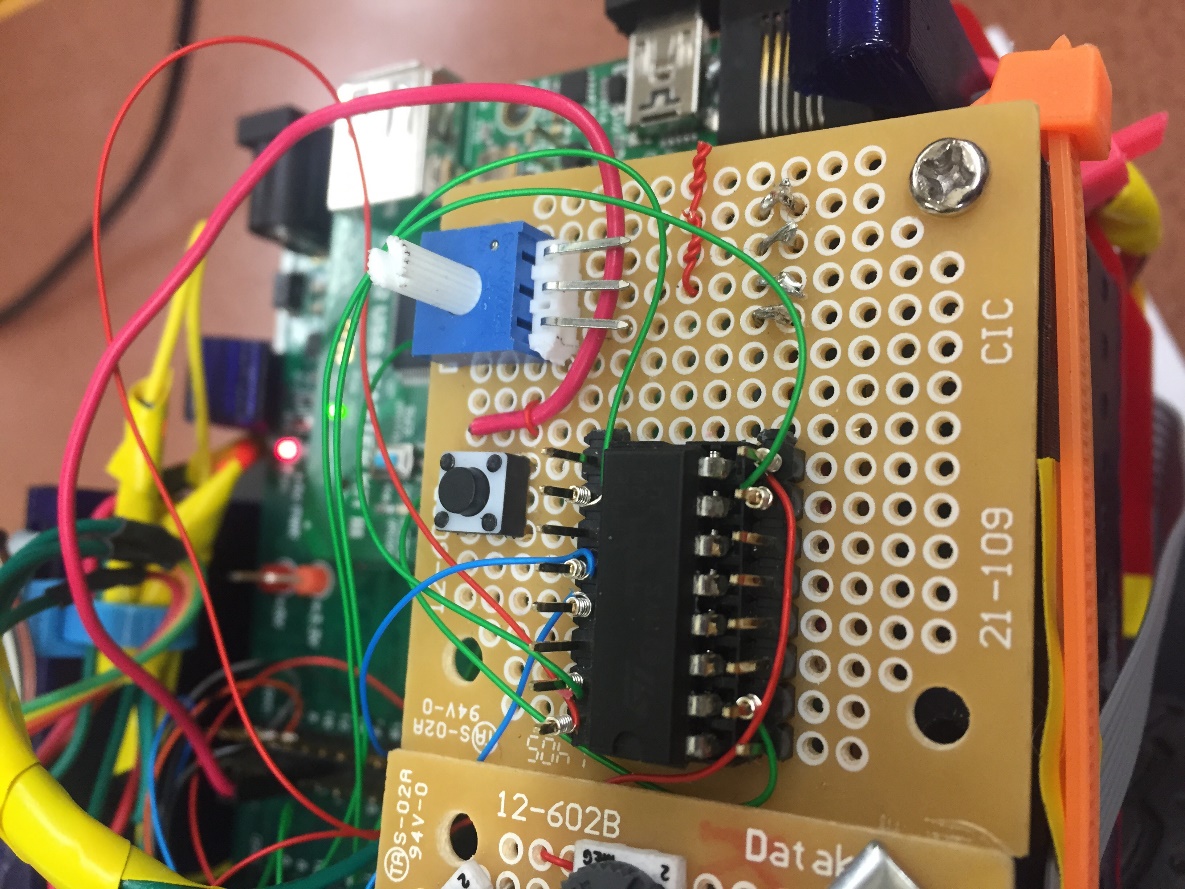
### Hardware



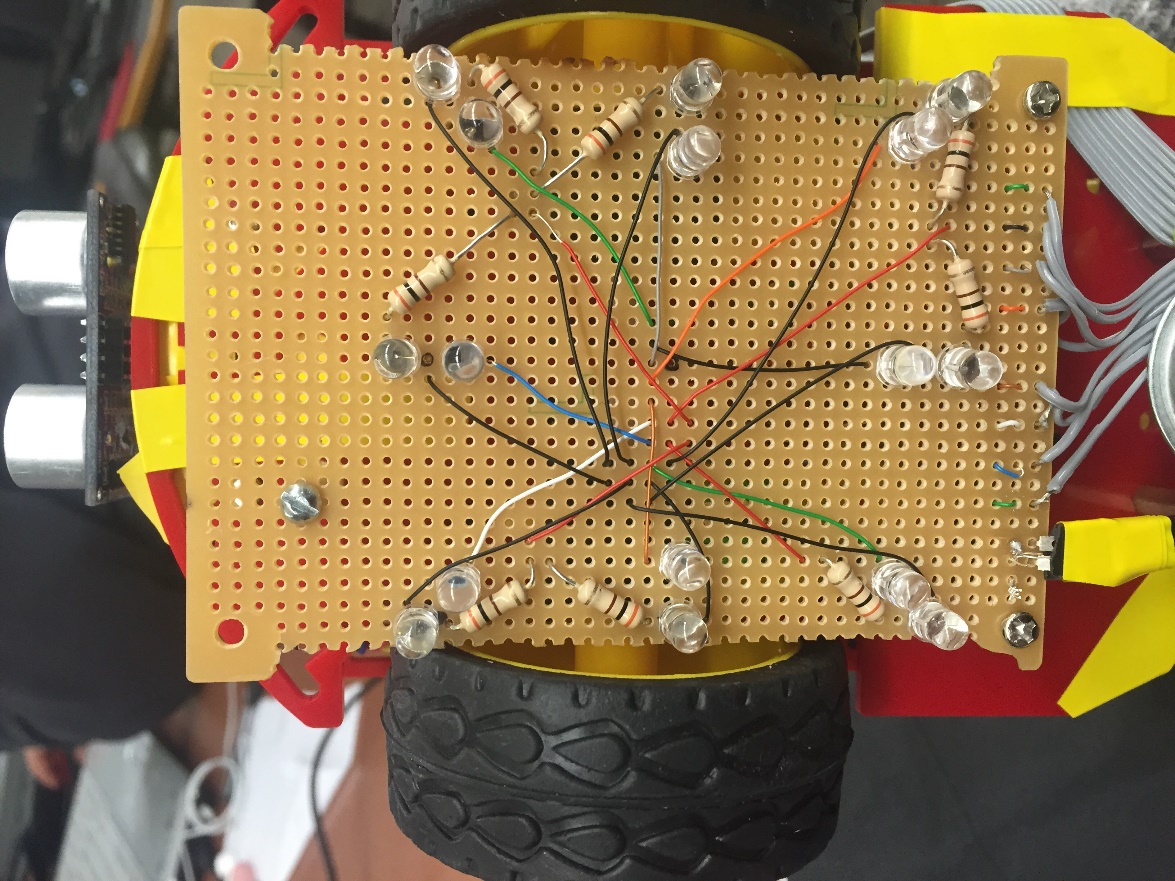
Ultrasonic Sensor



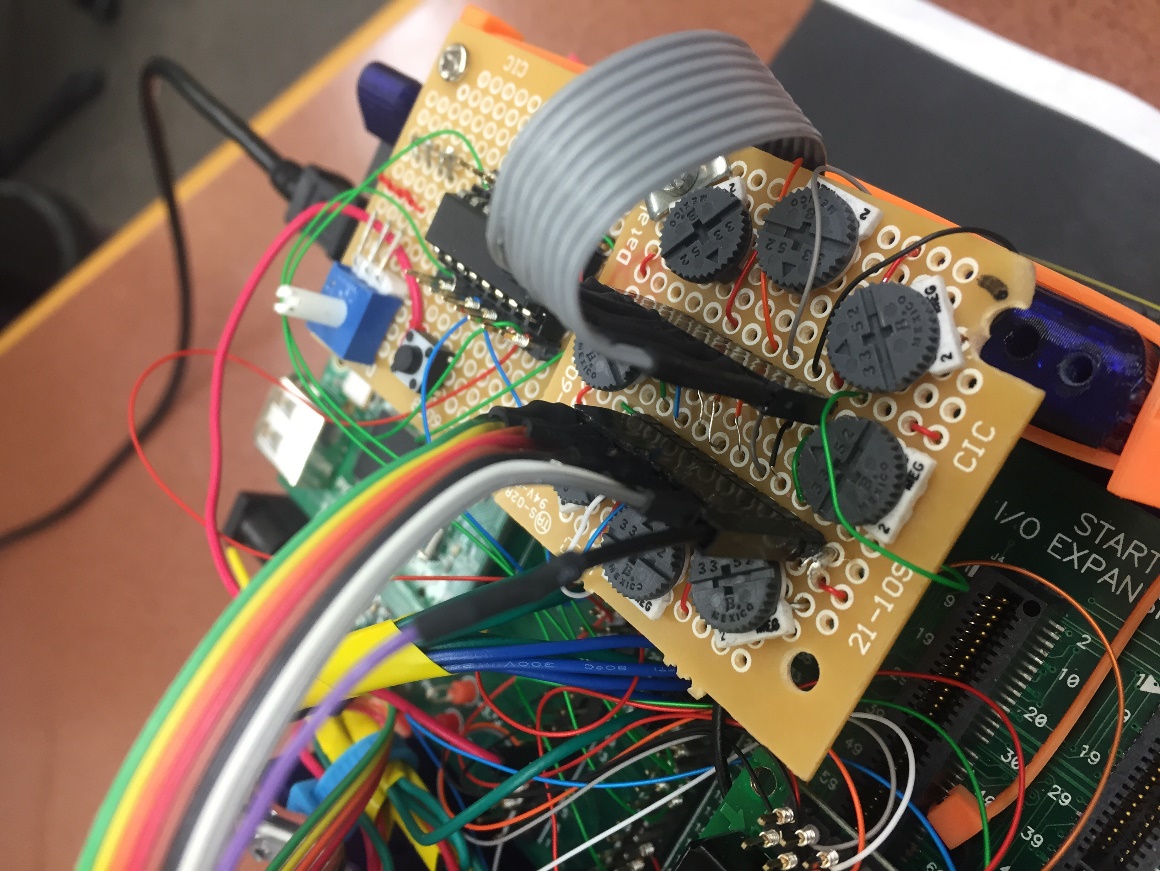
LCD SCREEN



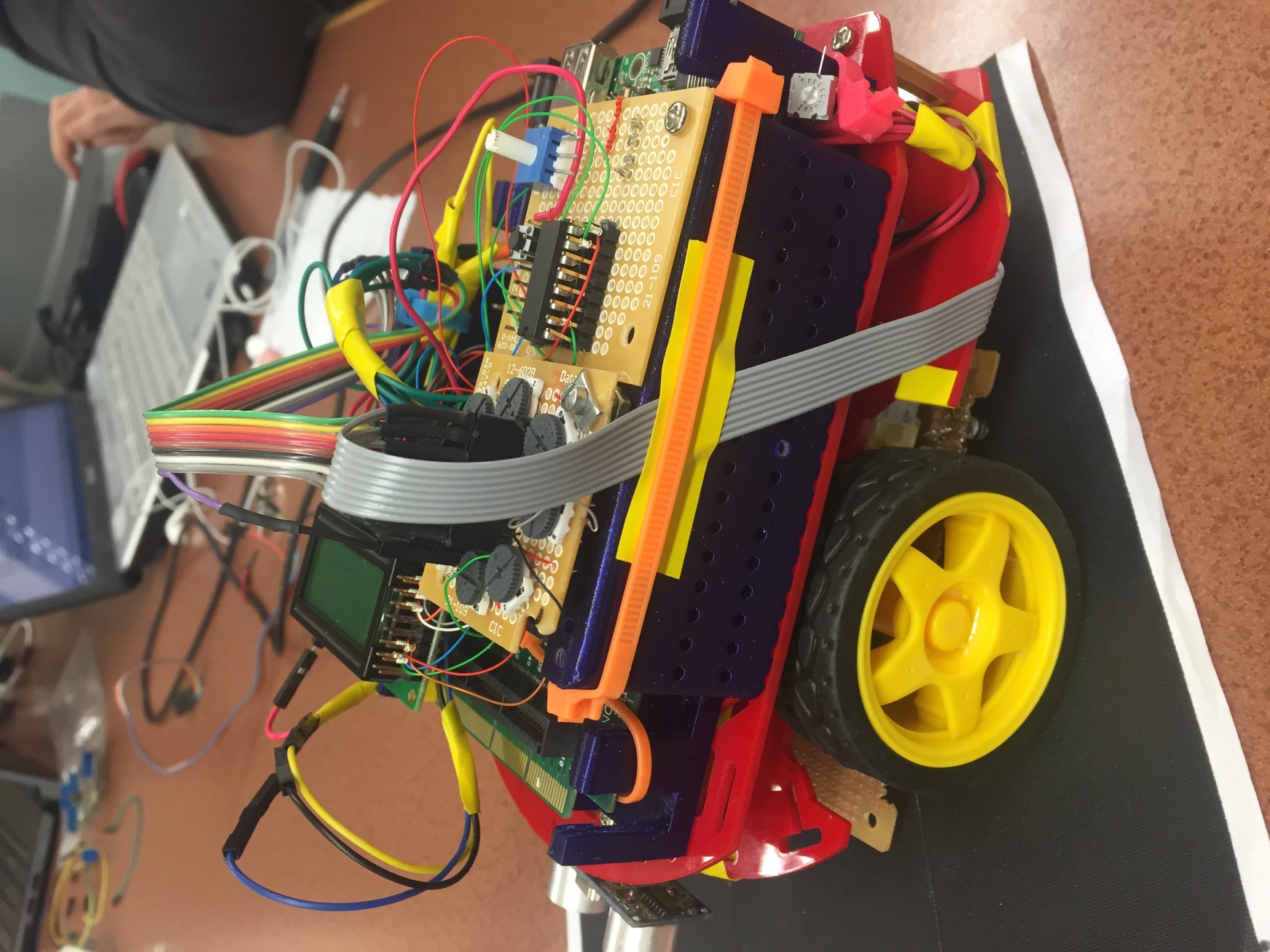
H-Bridge for Motor Control



IR SENSORS



POTENTIOMETERS FOR IR SENSORS & CONNECTORS



The Coolest Robot Ever

### Additional Part description

Table 1

Parts added to lab

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Part Name** | **Part Number** | **Cost** | **Source** | **Manufacturer** | **Purpose** |
| 1M Potentiometer | A0013 | $1.00 | Stock Room | China | To adjust sensitivity of the infrared arrays |
| Ultrasound Distance Sensor | HC-SR04 | $10.00 (2 pack) | Amazon | Sunfounder | Object distance detection |
| 300 Ω resistor | n/a | $0.00003 | Stock room | No one really knows | To regulate current flow through the IR LEDs |
| 15 kΩ resistor | n/a | $0.000031 | Stock room | Certainly not the Borg | To regulate current flow through the phototransistors |

### Pin Assignments and Pin Usage Descriptions

Figure 1

PIC32MX470F512L Pinout

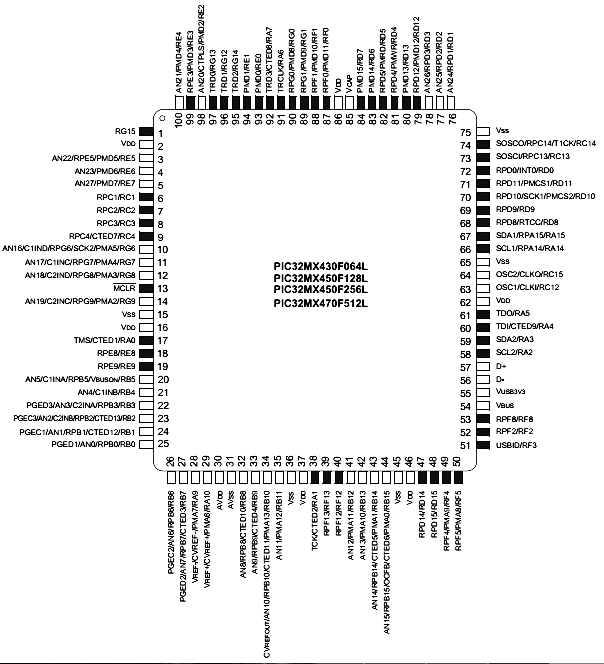
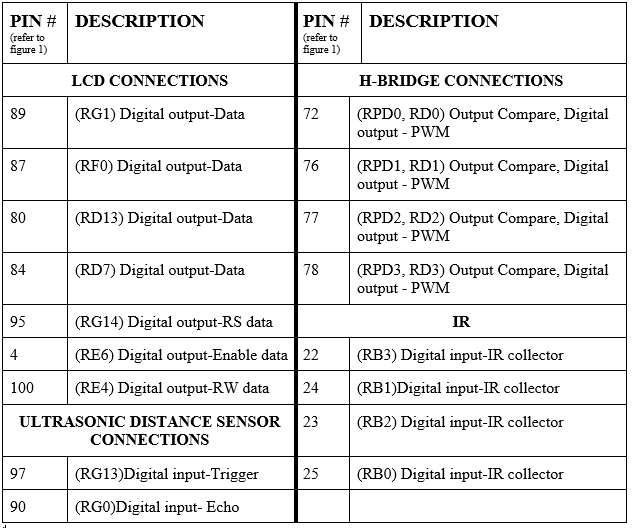


Table 2

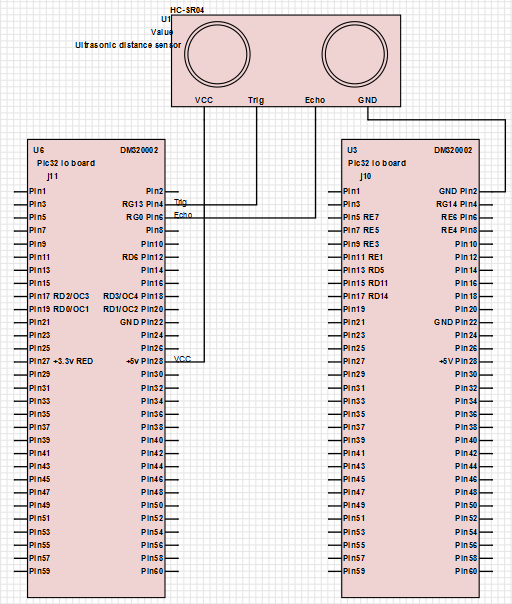


### Circuit Diagrams

Ultrasonic Distance Sensor Circuit Diagram

Figure 2

Ultrasound distance sensor circuit diagram



IR Circuit Diagrams

Figure 3a

IR circuit diagram (connections to expansion board)

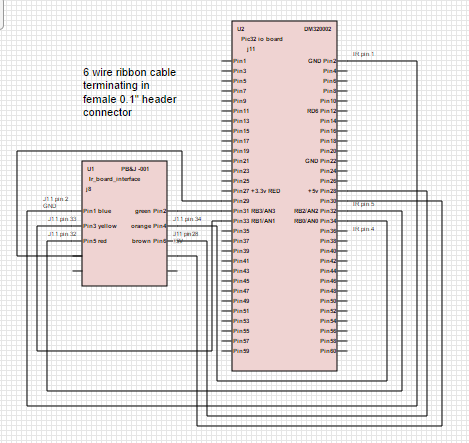
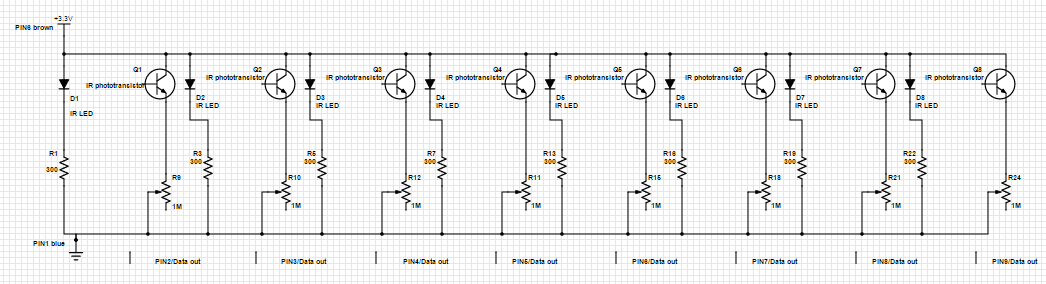
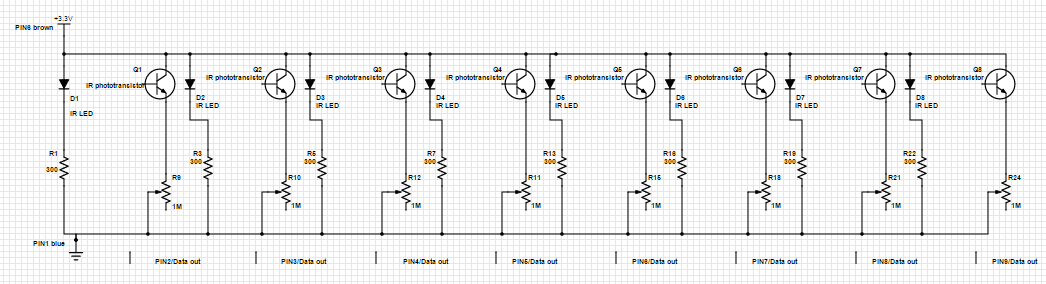


Figure 3b

IR circuit diagram (internal connections)

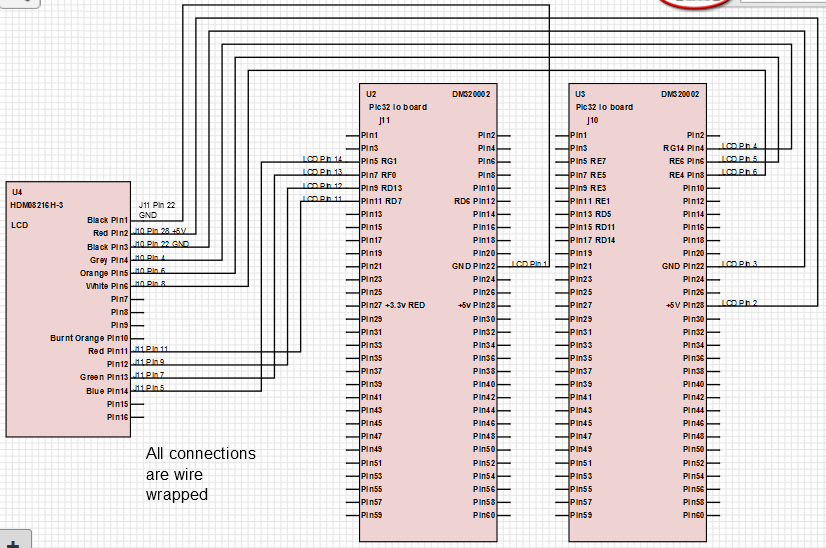




LCD Circuit Diagram

Figure 4

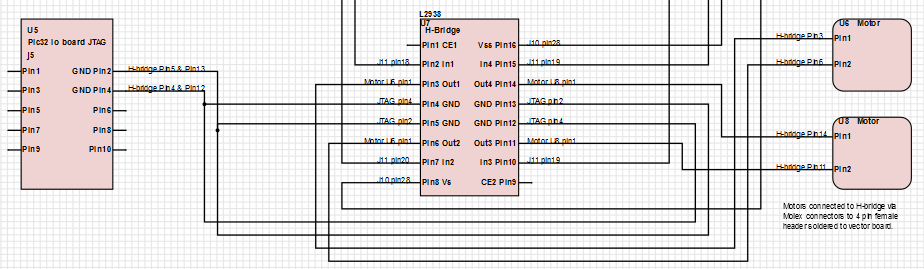
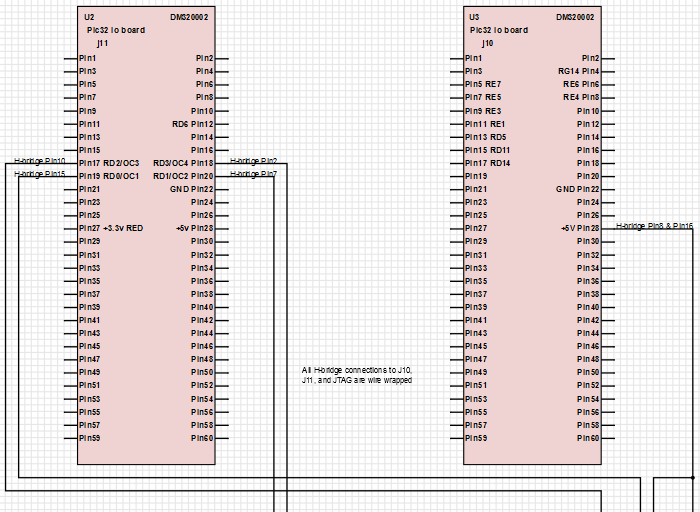
LCD circuit diagram



Motor/H-Bridge Circuit Diagram

Figure 5

H-bridge and motors circuit diagram (supposed to be one diagram, but SchemeIt sucks)



# Software Design

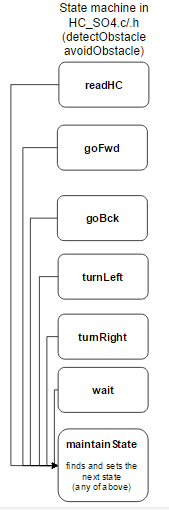
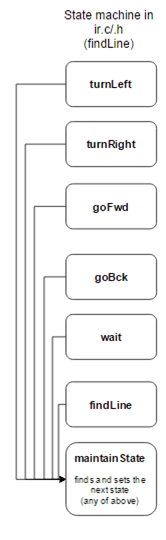
### Overview

Given the additional complexity of this project when compared to previous labs, the focus of the software design for the final is on scalability. The software is built in a way that enables both easy isolation of the robot subsystems and allows for easy modification of robot behavior at a high level. ‘main.c’ implements a state machine that carries out high level tasks while lower level tasks controlled by sub-state machines. This structure should make for simpler debugging.

This being said, the high level state machine will have states that that run tasks similar to the following: track line, check for robot, avoid obstacle etc. These tasks will then be defined as their own state machine in their corresponding .h and .c file.

The rightmost state machine below describes the high level state machine that we will be implementing in main.c. As described previously in this document, the states findLine, handleCollision, avoidObstacle, and detectObstacle in the main.c state machine will be located in separate files. The descriptions of these state machines are depicted by the diagrams below. Additionally, the diagram shows files that the state machines are in will be listed in. Each state will call a function that carries out an action. For example: the turnLeft state will call a function turnLeft() that makes the robot turnLeft. The readHc state will call a function readHc() that reads from the sensor. The same goes for all of the other states.

### Block Diagram



### Function Descriptions

Functions in pwm.c/.h

void initPWM();

Initialize the PWM for the motors and configure the pins for the robot.

void setMotorsIdle();

Sets motors to Idle.

void setMotorsBackward(int s);

Sets the motors to move in reverse with a speed of s.

void setMotorsForward(int s);

Sets the motors to move forward with a speed of s.

void setMotorsLeft(int s);

Sets the left motor to move forward with a speed of s, and right motor to stop.

void setMotorsRight(int s);

Sets the right motor to move forward with a speed of s, and left motor to stop.

void motorPiviotLeft(int s);

Sets the left motor to move forward with a speed of s, and the right motor to move backward with a speed of s.

void motorPiviotLeft(int s);

Sets the right motor to move forward with a speed of s, and the left motor to move backward with a speed of s.

void motorFindLine(int s);

Makes the robot move in a pattern that attempts to find a line.

Functions in ir.c/.h

void initIR();

Initialize the IR sensors and configure the pins for the robot.

void printIR();

Read from the IR sensors and write to the LCD.

int readIR();

Read from the IR and return an integer that can be decoded into the values read from each IR sensor.

int trackLine();

A state machine, as described above, that enables the robot to track a line.

irStateType parseIRData(int data);

Parse the data returned from the readIR() function, and return a state that the trackLine() state machine can use.

Functions in HC\_S04.c/.h

int FindDistance();

Configures the sensor to send out a pulse to determine the distance of an object in front of it.

int CalculateDistance();

Calculates the distance of an object in front of the sensor in inches, returns that value. Called within the FindDistance function.

void testUltraSonicSensor();

Once per second calls the FindDistance function and displays the distance found on the LCD screen.

# Ultrasonic Distance Sensor

### Task 1

The first task to be completed with the ultrasonic distance sensor was to detect the distance of an object in front of the sensor and display that distance onto the LCD screen. This also needed to be done dynamically, meaning the sensor should constantly scan the field in front of it and report the new distance. This task was completed by calling the testUltraSonicSensor function, which repeatedly sent a signal to the sensor to detect an object. An example of this task being completed can be found here:

<https://www.youtube.com/watch?v=qMZsuvicEvE>

### Task 2

The second task to be completed with the sensor was to detect an object obstructing its path and adjust its’ motors in such a way as to avoid the object and continue forward. This was done by introducing a sub state machine that was entered whenever the sensor detected an object within 5in. This state machine configured the motors to do a simple evasive maneuver and avoid the object. An example of this task being completed can be found here:

<https://www.youtube.com/watch?v=t_rYpcuMDTQ>

### Task 3

To be completed…