LINE TRACKING ROBOT

FINAL REPORT

ECE 372A
Submitted By: Team 214
Jason Blomquist
Ezadeen Naji
Thales Priolli

Submitted On:
May 10th, 2014

Table of Contents

1 Abstract

This project involves the creation and design of a line-tracking robot that uses infrared sensor pairs to detect and correct position. The designed robot meets the ECE 372A project requirements. It can follow the track made of black tape, scan and display information along the side of the track on an LCD display, and it can be controlled wirelessly with a custom designed external remote control.

2 Introduction

The challenge our team faced in this project was to build a robot capable of fulfilling three requirements:

1) Follow a predesigned track from beginning to end, turn around at the end, and follow it back to the beginning, without driving off the track.
2) Scan 2 randomly chosen binary strips, or “barcodes”, and display their relevant information on an LCD display.
3) Add an additional feature to our robot of our choice.

To follow the predesigned track, our robot used three sets of IR phototransistors and IR emitters. Using the data produced by this set of components, we were able to design code that allowed our robot to automatically correct its path along the track. Using a similar idea, we used an additional IR emitter and phototransistor pair to scan the randomly chosen barcodes. As the robot drove across the track, the barcodes were placed on the floor and scanned as it passed over top. For the last part of our design, we decided to implement wireless control via a custom built external remote control and radio frequency modules.

To fully understand our design, one must be familiar with the PIC24F microcontroller and starter board, as well as the following subjects and tasks:

- C programming
- Software interrupts and timers
- Interfacing hardware with microcontrollers
- Use of H-Bridge for motor control
- Analog-to-Digital Conversion, or ADC
- Pulse-Width Modulation, or PWM
- Custom circuit design
- Use of radio frequency transmitters and receivers
- Use of an oscilloscope, DC power supply, and digital multimeter

Our design has an advantage over other designs because in addition to the existing implementation of the line-tracking algorithm we have designed a way to control the robot independently and to the user’s desire. Our method of external control, besides meeting these project requirements, has many uses outside of this project and could be implemented in many other designs.

3 Technical Discussion and Detail

Our robot's design starts at its base, the Magician Chassis (ROB-10825), which we were provided in class. This kit provided our team with a basic shell, two DC motors, and an omnidirectional back wheel. We were also provided with IR emitters, IR phototransistors, an LCD display, an H-bridge for motor direction control, multiple PCBs, 30-gauge wire for making hardware connections, a 9V battery for stand-alone operation, among other minor hardware components. Additional parts were available through the ECE stockroom, and we obtained various circuitry components for designing our external remote control.

3.1 Hardware Design

Our team made a couple important hardware decisions that greatly improved the consistency and convenience of testing our robot. First, instead of powering our DC motors with the provided 9V battery, we opted to use CGR18650D Lithium Ion cells which we obtained from an old laptop battery. Using two sets of two in parallel, the batteries provided us with a very consistent output voltage of around 7.6V, and they had a combined capacity of ~4700mA which was way more than the 9V batteries provided to us. The datasheet for these batteries can be found in our references. In addition to using these batteries, we soldered a power transistor and potentiometer onto the power line going into the H-Bridge, and this allowed us to quickly change the maximum speed of the motors without adjusting the duty cycle in the software. Because our motors were being powered by an external battery, the 9V used for powering the starter board in stand-alone mode lasted much longer than it was before.

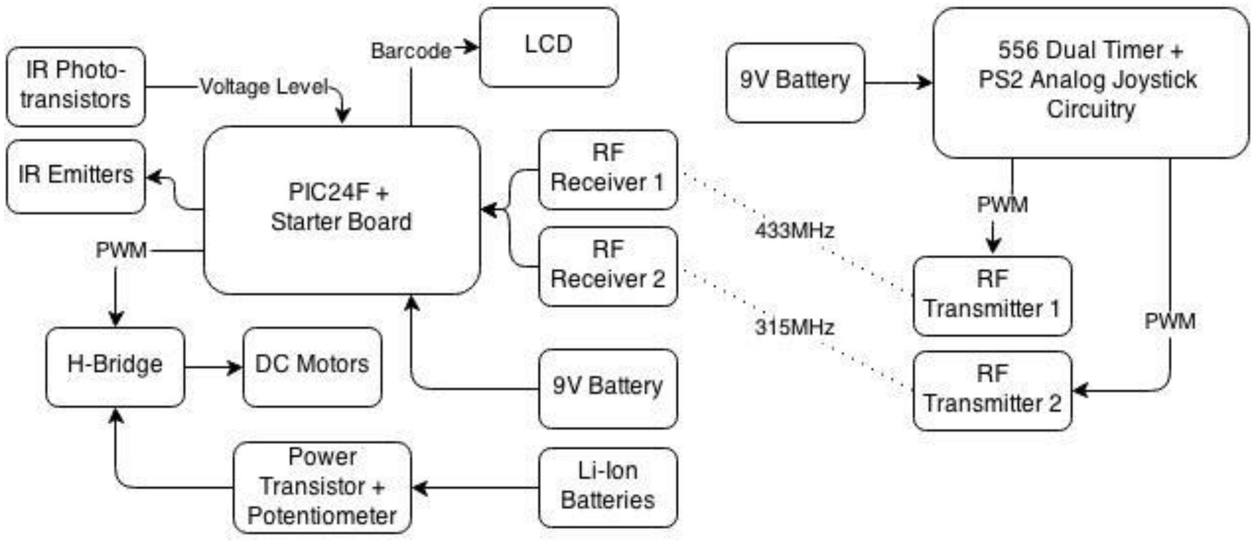


Figure 1: System Block Diagram

Figure 1 above shows the basic layout of our design. The starter board was mounted on the top of our robot, secured down with tape. The three pairs of directional IR emitters and phototransistors were securely mounted to the front of our robot, and the additional barcode-scanning pair was mounted on the left side of our robot. For photos of our robot, refer to Appendix A. An additional PCB was mounted to the top of our robot, which contained the LCD for displaying the barcode, the H-Bridge for motor control, and the radio frequency receivers.

As mentioned before, our team decided to implement wireless control as our additional feature, and to do this we had to purchase two sets of radio frequency (RF) modules that operated at two different frequencies, 315MHz and 433MHz. The two receiver modules were mounted to the top of the robot in such a way that allowed us to easily test the data being received and allowed for the antennas to have proper clearance. The transmitter modules were mounted to the external PCB that included our 556 timer and analog joystick circuitry, and this was all contained in a plastic cassette tape box, to prevent the circuitry from being exposed during handheld usage. Detailed circuitry and photos of our remote can be found in Appendices A and C.

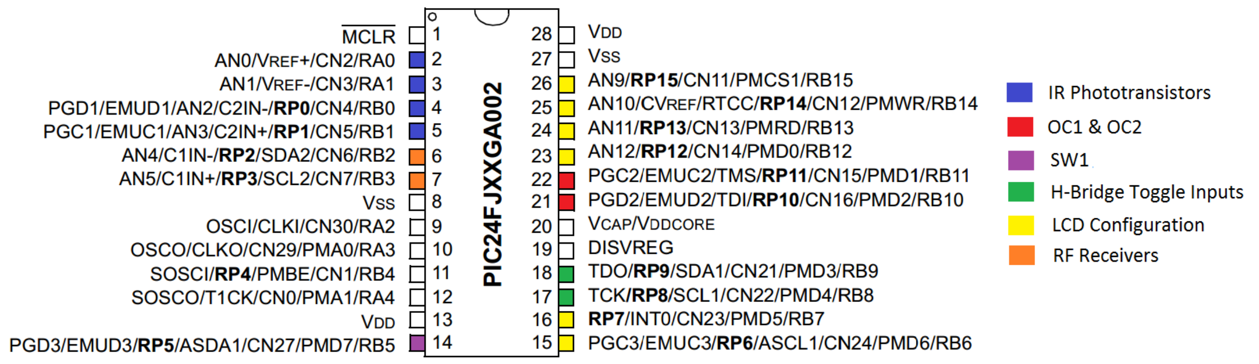


Figure 2: PIC24F Pin Configuration

Figure 2 shows the final pin configuration that we used for our design. The four sets of phototransistors were assigned to pins 2-5, and the two radio frequency receivers were assigned to pins 6 and 7. This pin configuration was necessary because all of these components required Analog-to-Digital Conversion, or ADC, to obtain workable values, and these listed pins had the option of being analog input pins, which is denoted by the presence of an ANx value listed to the side of the pin. The rest of the input pins were configured as digital inputs. RB5 was used for the integrated switch on the starter board for mode switching, the LCD pins were used to send the barcode information to the LCD display, the OC1 & OC2 pins were sent to the H-Bridge for changing the speed of the motors, and the H-Bridge toggle pins were used for changing the directions of the motors.

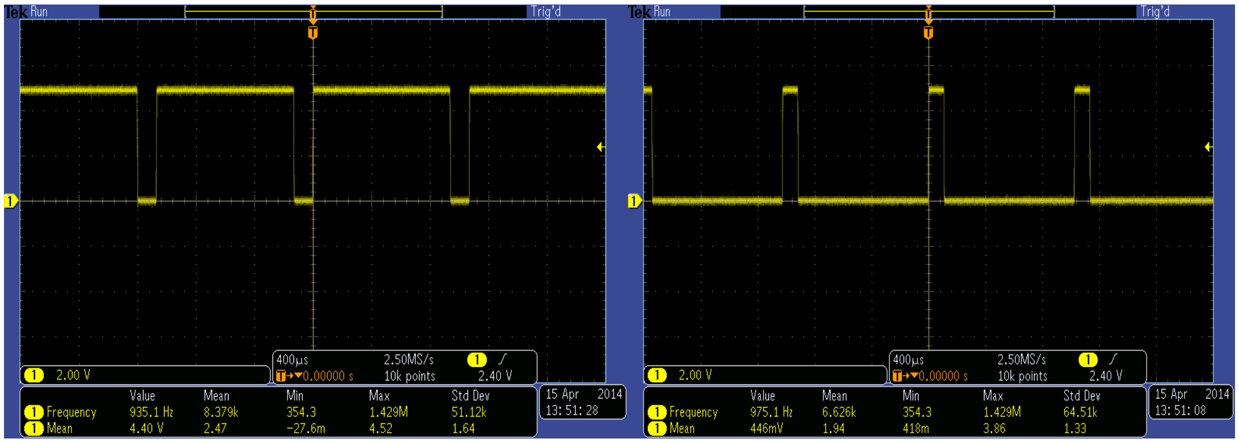
3.1.1 Wireless Remote Control Design

The analog wireless remote control primarily consists of two parts.

1. Dual Pulse Width Modulator PWM:

The dual PMW circuit uses a 556 timer (two 555 timer chips in one case) that generates two separate pulses with variable duty cycles that can be controlled by two separate potentiometers. The circuit is capable of providing controllable duty cycle that varies from 10% to 85%. See oscilloscope screen shots below. This circuit is a very common and simple circuit to construct and available online.

Please refer to Appendix C for schematic.



	Value	Mean	Min	Max	Std Dev
1 Frequency	935.1 Hz	8.379k	354.3	1.429M	51.12k
1 Mean	4.40 V	2.47	-27.6m	4.52	1.64

15 Apr 2014
13: 51: 28

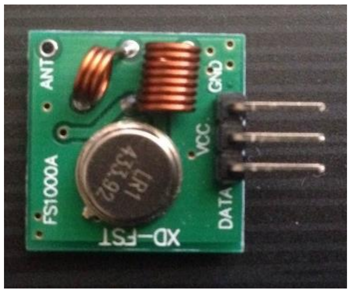
	Value	Mean	Min	Max	Std Dev
1 Frequency	975.1 Hz	6.626k	354.3	1.429M	64.51k
1 Mean	446mV	1.94	418m	3.86	1.33

15 Apr 2014
13: 51: 08

Figure 3: Oscilloscope Screenshots

2. Two RF Modules:

Two RF transmitters are used to transmit the pulses generated by the 556 timer circuit to the receiver unit attached to the robot. They broadcast the data to the robot using two different frequencies, 315MHz and 433MHz, to avoid interference. These RF modules, supplied by 5v power source, are capable of transmitting accurate data as far as 30 meters and through walls and would probably reach farther distances if higher power was used. These transmitters have three pins each: Power, Ground, and Data (See picture below). The data pin connects to the signal output pin of the 556 timer.



The antennas used are insulated copper wires. The antenna must be a quarter wavelength long for better communication. Therefore, the antenna for the 433MHz RF module is 173cm, and 238cm for the 315MHz RF module. Although the RF modules are able to communicate well without antennas, adding the antennas with the proper length maximize the distance and quality of the transferred data. Note that the antennas' length must be matched between each transmitter unit and its receiver.

The wireless remote control circuit is supplied by a 9v 250mAh battery provided by the ECE stock room. However, the circuit uses a LM7805 +5V DC voltage regulator to insure a steady voltage supply to the 556 timer and RF modules, which in turn assists the circuit to send consistent data.

The pulses sent to the RF receiver is passed to a Low-Pass RC filter consisting of a 10k ohm resistor and a 10uF electrolyte capacitor to convert the pulses to a DC voltage.

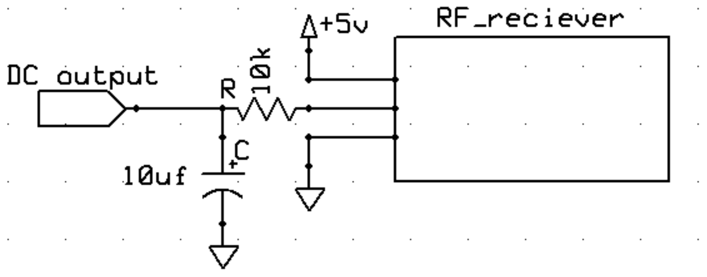


Figure 4: **Low-Pass filter connected to the data output of the receiver.**

The change of the duty cycle will result in a change in the average dc voltage measured at the output of the Low-Pass filter. Therefore, the average voltage is converted to a Digital value using the Analog to Digital Converter ADC offered by the PIC24FJ64GA002 microcontroller chip. Then these values are used in the software to program the PIC24FJ64GA002 to control the direction and speed of the motors on the robot.

3.1.2 Infrared Sensors (IR)

The infrared sensing units each consist mainly of an IR emitting diode and a phototransistor placed next to each other which allows the phototransistor to sense the amount of infrared light reflected from various colored surfaces. The IR emitter is supplied from a DC 3.3V source provided by the DM300027 Starter Board through a 75 ohm resistor to insure that only 1.2v and 30mA go through the IR emitter. As for the phototransistor, we used a 1k ohm resistor connected in series with the phototransistor so that the series circuit act as a voltage divider. The collector pin of the phototransistor is connected to the 1k resistor and the emitter is connected to the ground.

It is known that different colored surfaces reflect different amount of light. Hence, the phototransistor allows different amount of current through its collector and emitter when different amount of light hit its surface.

When testing the IR circuit, we were able to measure voltage values at the collector pin of the phototransistor when the IR sensor faces different color surfaces. The voltage varied between 0.6v up to 3.1v which then was converted to digital values using the ADC. Thus, based on the different ADC values read by the microcontroller from the IR sensors we designed our software to execute different commands in correspondence to that ADC values. For example, correcting the robot direction when following the track, displaying information on the LCD, etc.

3.1.3 H-Bridge

The design of the H-Bridge is taken from the data sheet provided to us in lab. However, we made a few changes so that the circuit serves our purposes. The L293B chip is mounted on a separate PCB board along with the LCD and RF receiver as mentioned earlier in sec 3.1 Hardware Design. The H-bridge is supplied by a 7.6v 4700mAh Li-Ion battery pack through a power control circuit for easy voltage adjustment.

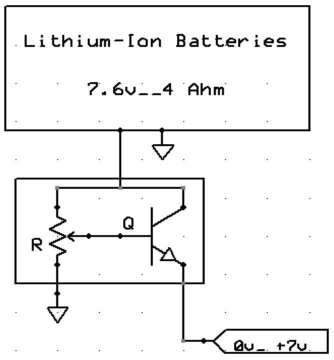


Figure 5: Voltage control circuit connected to the battery pack

3.2 Software Design

This software was developed in the programming language C, and it includes two source codes files (*.c) and two headers files (*.h), listed in Appendix D. The IDE utilized for the development was MPLAB X IDE with the Microchip XC16 Toolsuite. The main code used in the robot is the file final.c, it has the configuration for the PIC24F microcontroller and functions implemented that allow the robot to achieve the goals of the project.

3.2.1 General Configuration

At the beginning of the code, there are some specific configurations for the PIC24FJ64GA002, include statements for the header files listed in Appendix D as well as the built in header <stdio.h>, and define functions. Define functions are utilized to set global, constant values to a variable and to create a nickname for registers which can be used throughout the code. For example, the line "#define SENSOR1 ADC1BUF0" allows the user to use the word SENSOR1 instead of ADC1BUF0, and this is used throughout the code to help the user understand the function of the variables.

3.2.2 Main Function

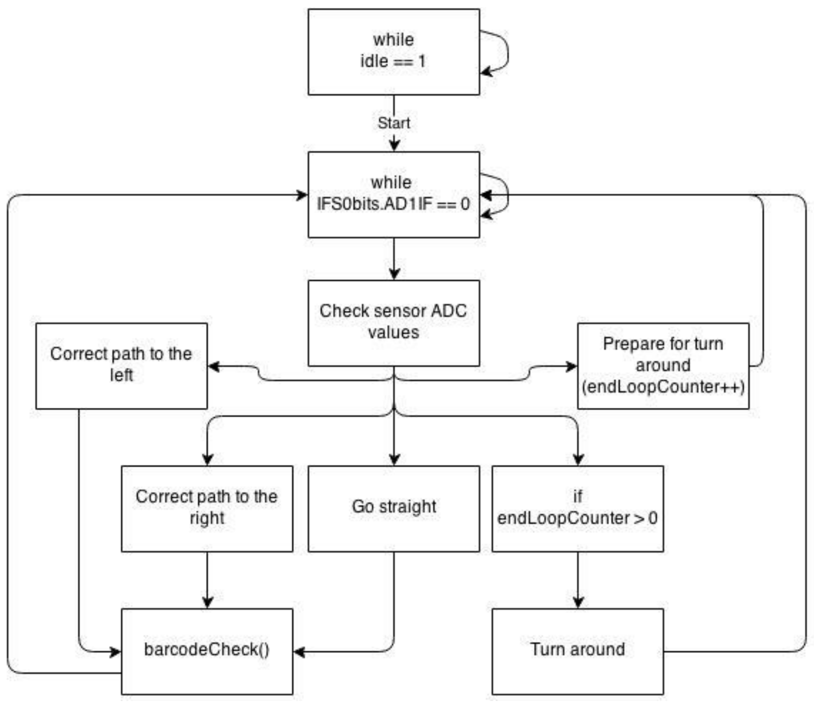


Figure 6: Main Function/Line Tracking Flowchart

Figure 6 shows the functionality behind our software that drives the robot along the track. The robot waits while the idle variable is high, and as soon as that value is updated by the CNInterrupt (shown in Figure 7) the robot attempts to start driving along the track. First, the IFS0bits.AD1IF flag is checked and the program will wait until the ADC conversion has finished. Then, it checks the current converted ADC values of the three front IR phototransistors. We had some inconsistencies between the ADC values from our three phototransistors, even after rebuilding the circuit multiple times, and these problems will be discussed in a later section. The thresholds that we used for the three phototransistors, starting at the leftmost, were 700, 800, and 950, respectively. If the ADC value of the leftmost phototransistor went above its threshold and the ADC values of the other two stayed below their thresholds, the duty cycle of the left motor was set to 0 and the duty cycle of the right motor was set to 100 so the robot could steer to the left. Similarly, if the rightmost phototransistor went above its threshold and the other two dropped below their thresholds, the opposite would happen; the duty cycle of the left motor was set to 100 and the duty cycle of the right motor was set to 0 so the robot could steer to the right.

At the end of the track, our robot had to turn around and follow the track back to the beginning. We implemented this turn around by using a 32-bit timer and a delay function. We tried various ways of turning around, but what ended up working best for us was reversing the motor directions and pivoting on one wheel for a specific amount of time at the end of the track. The area that our robot was turning around in gave us a large enough piece of track where even if the turnaround function was not perfectly consistent (because of its dependence on the voltage level going into the motors, controlled by the inline potentiometer and power transistor circuit), the robot would find the track and correct its path automatically on each run.

We ran into a few issues near the end of the track when our robot would drive over the corner where the center of the "A" started. There was a moment during that part where our robots sensors would all read black, and the robot would execute its turn around function. To prevent this from happening, we added an extra variable, which can be seen on the flowchart, called endLoopCounter, and it would be incremented after going over that part of the track. In addition, when going over that part of the track both times, once forward and then backward, the motor speeds were adjusted specifically for that turn.

Obviously, if none of these conditions were met, the robot would continue going straight. In addition, as seen in the flowchart, after each time the robot steers left, right, or continues straight, it checks to see if a barcode was sensed by the barcode-reading IR phototransistor.

3.2.3 Interrupt Function

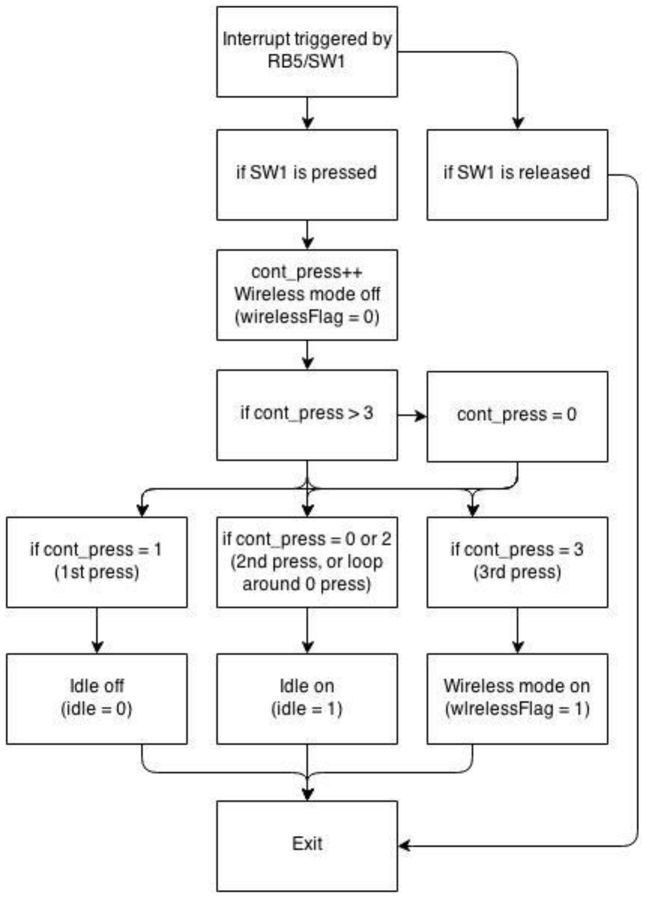

Interrupt triggered by RB5/SW1
if SW1 is pressed
if SW1 is released
cont_press++ Wireless mode off (wirelessFlag = 0)
if cont_press > 3
cont_press = 0
if cont_press = 1 (1st press)
if cont_press = 0 or 2 (2nd press, or loop around 0 press)
if cont_press = 3 (3rd press)
Idle off (idle = 0)
Idle on (idle = 1)
Wireless mode on (wirelessFlag = 1)
Exit

Figure 7: CNInterrupt Flowchart

Our team used the integrated switch SW1 on our starter boards along with CNInterrupt to change the functioning mode of our robot. Each time the switch was pressed, an interrupt was triggered and the cont_press variable was incremented. The order of states is as follows: idle -> track mode -> idle -> wireless mode -> idle -> etc.

3.2.4 Barcode Scanning Function

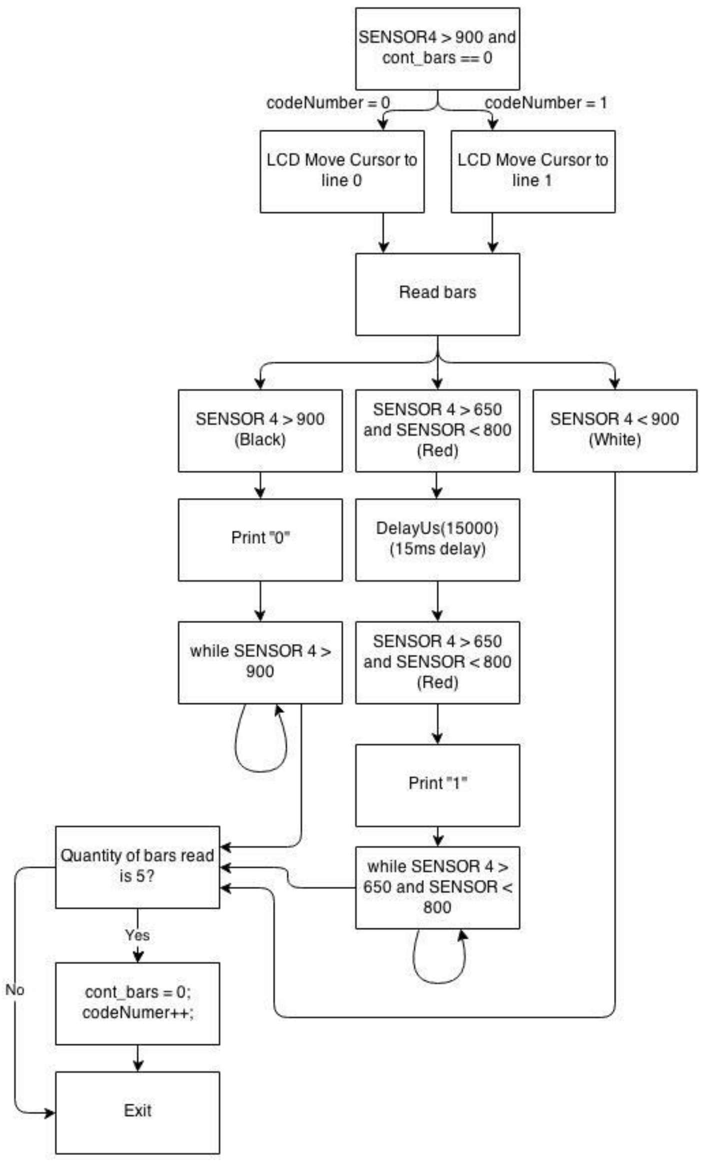

SENSOR4 > 900 and cont_bars == 0
codeNumber = 0
codeNumber = 1
LCD Move Cursor to line 0
LCD Move Cursor to line 1
Read bars
SENSOR 4 > 900 (Black)
SENSOR 4 > 650 and SENSOR < 800 (Red)
SENSOR 4 < 900 (White)
Print "0"
DelayUs(15000) (15ms delay)
while SENSOR 4 > 900
SENSOR 4 > 650 and SENSOR < 800 (Red)
Print "1"
Quantity of bars read is 5?
while SENSOR 4 > 650 and SENSOR < 800
Yes
No
cont_bars = 0; codeNumer++;
Exit

Figure 8: Barcode Scanning Flowchart

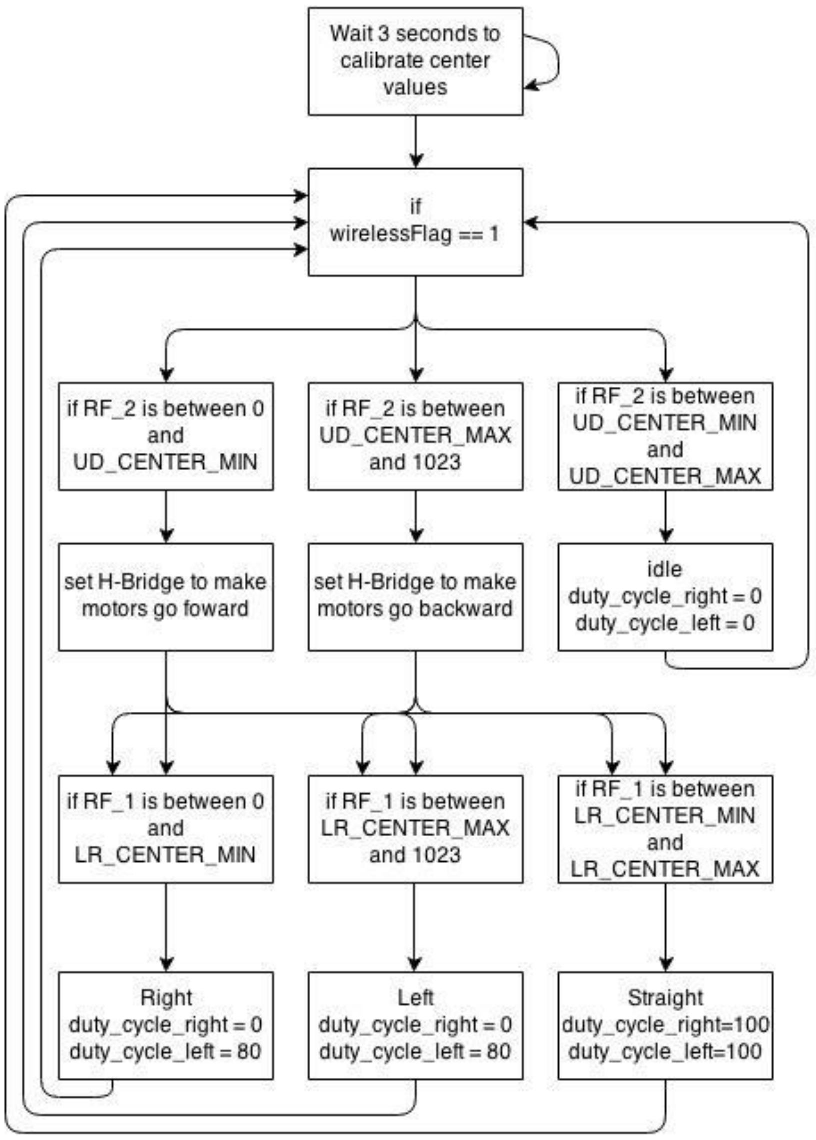
Figure 8 shows our barcode scanning function. As mentioned above in the main function section, each time the robot goes straight, left, or right it checks to see if the barcode sensor sees a barcode. This is accomplished by checking to see if the IR phototransistor, which is listed as SENSOR 4 in the flowchart, has its ADC value go above 900 while the robot is driving along the track. If this occurs, the function will start to execute. Once the barcode sensor sees a specific colored bar, the program waits for the ADC value to go beyond that specific threshold with a while loop. For instance, when the first black bar is seen by our sensor, before exiting the barcode function, we have the following code: while(SENSOR4 > 900). Once that condition is not true, the barcode function exits back to the main function so the robot can continue to correct its track position.

Our code also implements a small delay when the ADC value goes between 650 and 800, and this is to account for the transition between black and white bars. If the ADC value stays between 650 and 800 after a 15ms delay, we assume that the current color bar is red. Before exiting the function, we check to see if the current number of bars read is 5. If it is, we assume that the first barcode has finished scanning and the next time the barcode scanning function is called, LCDMoveCursor is called to move the cursor to the second line of the LCD to prevent the barcodes from overwriting each other.

3.2.5 Loop Delay Function

The implementation of this function is fairly straight forward. The loop delay function was created to help with the turnaround function in the main code. It uses timers 4 and 5 to create a 32-bit timer, which the user can use to delay for a relatively long period of time. Each time we called this function in our code, we changed the PR4 and PR5 values to meet our specific needs at that point in the program.

3.2.6 Wireless Mode Function



Wait 3 seconds to calibrate center values
if wirelessFlag == 1
if RF_2 is between 0 and UD_CENTER_MIN
if RF_2 is between UD_CENTER_MAX and 1023
if RF_2 is between UD_CENTER_MIN and UD_CENTER_MAX
set H-Bridge to make motors go foward
set H-Bridge to make motors go backward
idle duty_cycle_right = 0 duty_cycle_left = 0
if RF_1 is between 0 and LR_CENTER_MIN
if RF_1 is between LR_CENTER_MAX and 1023
if RF_1 is between LR_CENTER_MIN and LR_CENTER_MAX
Right duty_cycle_right = 0 duty_cycle_left = 80
Left duty_cycle_right = 0 duty_cycle_left = 80
Straight duty_cycle_right=100 duty_cycle_left=100

Figure 9: Wireless Mode Flowchart

Figure 9 shows the basic design of our wireless mode function. When the CNInterrupt, from above, changes the value of wirelessFlag to 1, the main function will jump to the outer wireless mode function. The first thing that happens when entering wireless mode is the calibration of the center values of the analog joystick after a 3 second delay. The user must first plug the 9V battery into the external remote control, and then switch the robot into wireless mode. The initial ADC values being received by the RF receivers will become the “center” position for the joystick. This code was necessary because of the inconsistencies in the received ADC values. In the flowchart above, RF_1 refers to the potentiometer that controls the left-right direction, and RF_2 refers to the potentiometer that controls the up-down direction. After initial calibration, LR_CENTER_MIN = (RF_1 – 75) and LR_CENTER_MAX = (RF_1 + 75). This gives us a center range of ADC values of 150, which makes the robot more stable because of the inconsistencies. Similarly, UD_CENTER_MIN = (RF_2 – 75) and UD_CENTER_MAX = (RF_2 + 75).

If the joystick was in the initial resting position, the robot did not move any direction. If it was pushed forward and the left-right position was left in the center, the robot would go directly forward. It the joystick was pushed forward and to the left or forward and to the right, as expected, the robot would go forward and left or forward and right, respectively. Going in the backward direction acted exactly the same, except the H-Bridge inputs were toggled to reverse the motor directions. Because of how we wrote our code, when the up-down position is centered but the left-right position is not, the robot will not turn because the up-down position takes precedence over the left-right position. If we had had more time, we would have added circuitry that would have allowed us to independently control the motor directions, so when the up-down position was centered but the left-right position was not, the robot would spin in place by running the motors in opposite directions.

4 Design Verification and Testing

To ensure that our robot was functioning properly, we carried out a series of tests. The beginning of our project involved getting the robot to follow the designated track, and using our IR sensor hardware, we had to design code that would make the robot follow the track and correct its path. In order to obtain workable values we printed the ADC values from the IR phototransistors on the LCD display attached to our robot. This allowed us to see what the values were at various states, and by using these values we were able to design code that met the project requirements. Similar tests were carried out for the barcode scanning IR sensor pair.

Our initial testing for the RF modules involved using an oscilloscope and a DC power supply to see what values we were obtaining from the wireless data transmission. Above, we have oscilloscope screenshots that show that we obtained a modulated pulse that varied between 10% and 85% duty cycle, and we decided that these values were definitely workable. Next, we connected this circuitry up to our PIC24F and tested to see if we could change the starter board’s integrated LEDs with the wirelessly transferred data, and this did indeed work as we had hoped. We were able to write code to change the LEDs on and off status based on the received and converted ADC value from the wireless transmitter and receiver pair. After verifying this functionality, we were able to fully implement our wireless control mode.

5 Problems and Solutions

Our team encountered quite a few problems throughout our designing process. One large problem we had was with the differences between the ADC values from the line-tracking IR phototransistors. We tried to rebuild the circuit in different ways a few times, but in the end we had to adjust our code in such a way that it would work with the current set of IR sensor pairs.

The provided 9V batteries proved to be a problem for most teams because of their low power output and need to constantly recharge, so our team decided we were going to use the Li-Ion cells that we conveniently had on hand. This made a massive difference in our testing because of the consistent output and long life.

The pair of motors that we were given originally did not provide enough torque to move along with track without getting stuck in place, and if we were to raise the duty cycle to try to prevent these motors from getting stuck, our robot would drive off the track. This prompted us to get another pair of motors from the ECE stock room. Our second set of motors was equally bad because they were not the same model, and one was much more powerful that the other. This caused our robot to constantly drive off the track, even with code modifications. When we finally got our third pair of motors, they provided enough torque to move along the track at a slow enough speed where our robot would not drive off the side, and we ended up doing our final demo with our third set of motors.

We also had a problem with our barcode scanning function when we originally wrote it because we did not take the transition between black and white bars in to consideration. When the robot moved over the binary strip, the ADC value from the relevant sensor pair would rapidly change. This caused our sensor to sometimes make an incorrect assumption about the current color of the binary strip, so we had to change our software to make an exception for this case. We did this by implementing a delay function when the ADV value went between 600 and 850, because it would go into that range when it was both on a red strip and when it was transitioning between black and white. The logic behind it was if the value dips into that range, wait 15ms and check again, and if it is still in that same range, the current color is red. This fixed the problem with the transition between black and white in our application.

Another problem we had was with the huge inconsistencies in the ADC values being read by the RF receivers. Originally, our code was written to use a preset center value, and it made the assumption that the ADC values from the receivers would sit in the middle between 500 and 550. We quickly discovered that this was not the case with our RF modules, so we had to make some hardware and software adjustments. The first thing we did was add proper antennas to our design, because we did not originally have them. Next, we changed the software to “calibrate” the center values each time the robot was switched into wireless mode. The explanation of how this was done can be found in the wireless mode section from above, 3.2.6.

6 Conclusions

Overall, our robot met the design requirements set forth in the project description, and our team was able to complete all the requirements in a timely manner. The robot was able to follow the track and consistently correct its path without driving off the track, it was able to turn around at the end of the track and follow it back to the beginning, and it was also able to read the randomly chosen barcodes during each track run.

In the end, we were happy with what we were able to accomplish in the amount of time we were given, but if we had had more time there are various improvements in the wireless mode function that we would have liked to make. Specifically, as I mentioned at the end of the wireless mode description, we wanted to implement a way to independently control the motors so our robot would be able to spin in place. Additionally, we would have liked to add a speed-scaling equation into the wireless mode function to more accurately control the direction of the robot, but this would have required us to purchase more reliable RF modules.

7 References

[1] http://www.romanblack.com/RF/cheapRFmodules.htm

[2] http://www.555-timer-circuits.com/

8 Appendices

Appendix A – Pictures of robot and hardware components

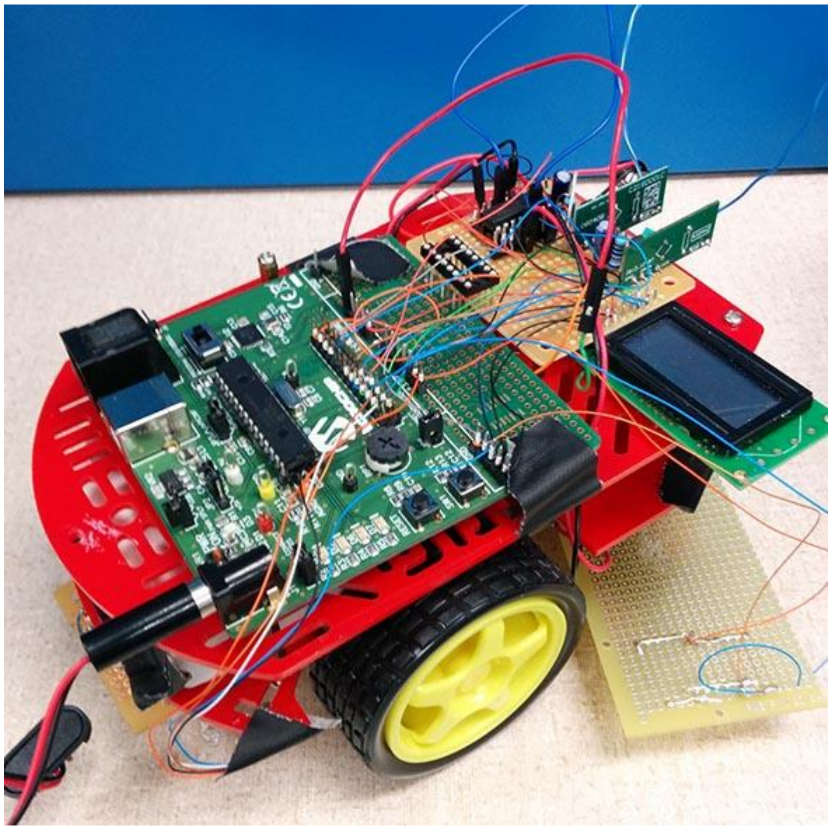


Figure 10: Top View of Completed Robot

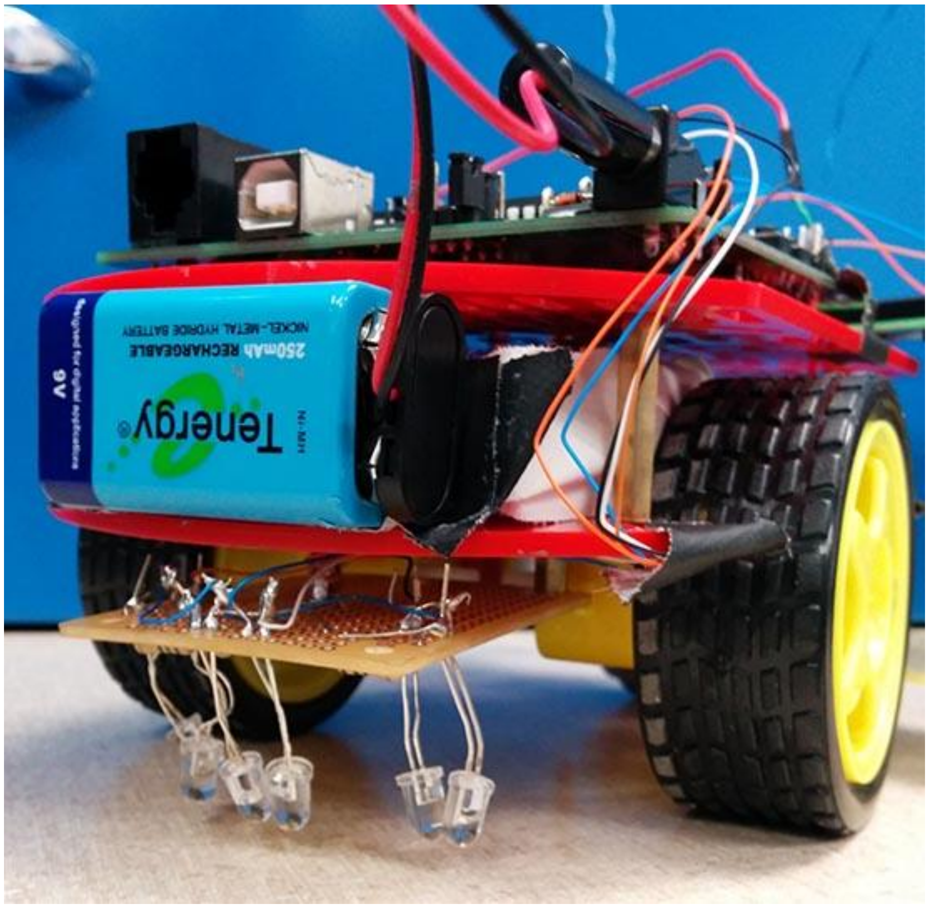


Figure 11: Front View of Completed Robot

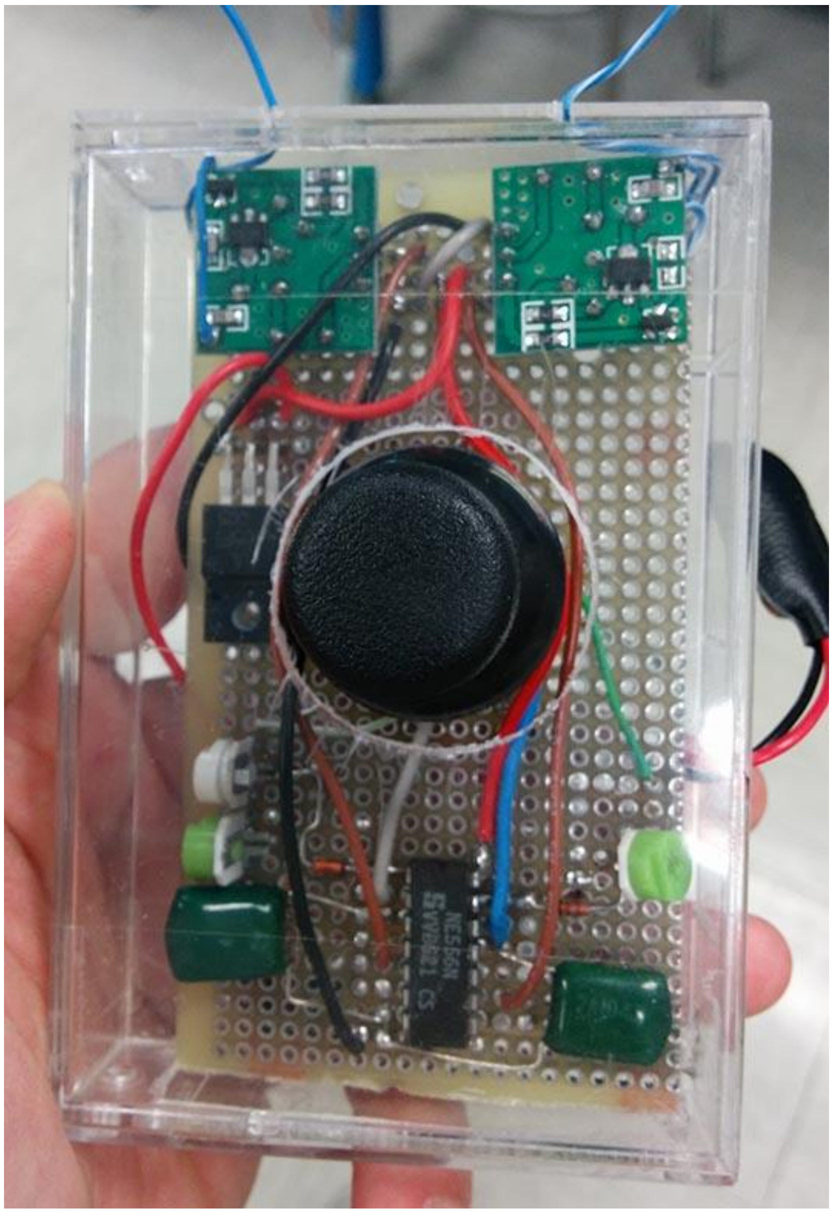


Figure 12: Completed Remote Control for Wireless Mode

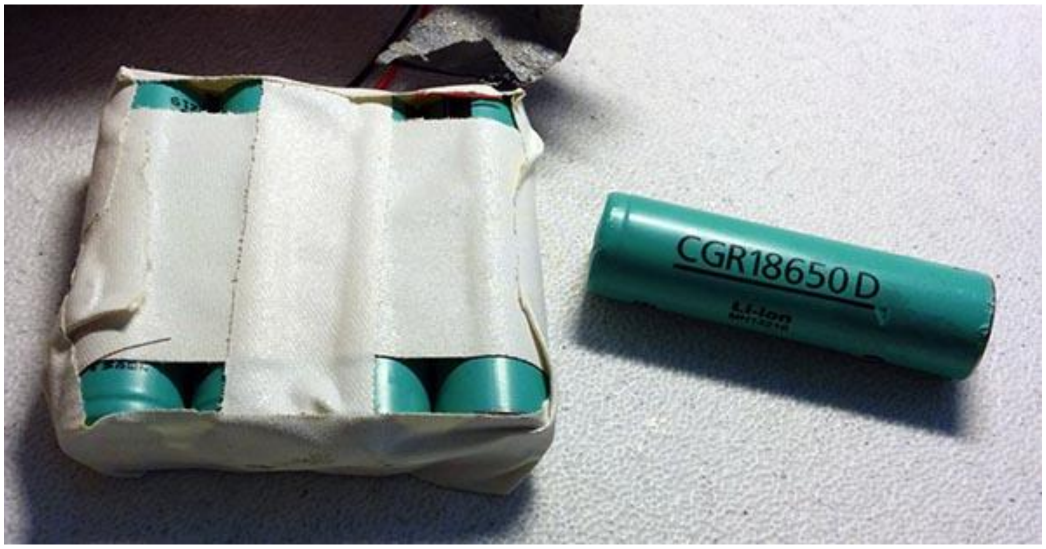


Figure 13: Lithium Ion Battery Pack used for DC Motor Power

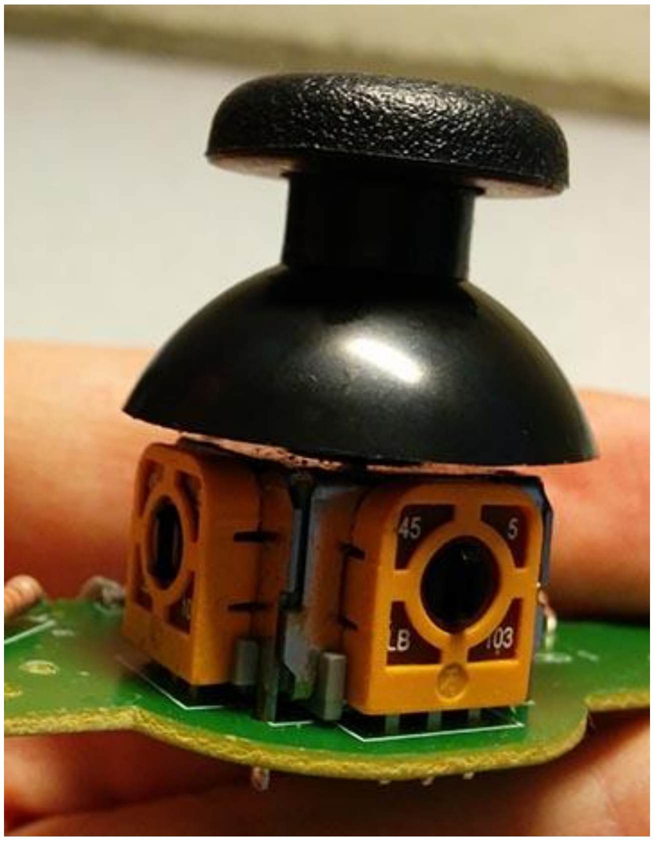


Figure 14: Two-Axis Potentiometer Joystick from PS2 Controller

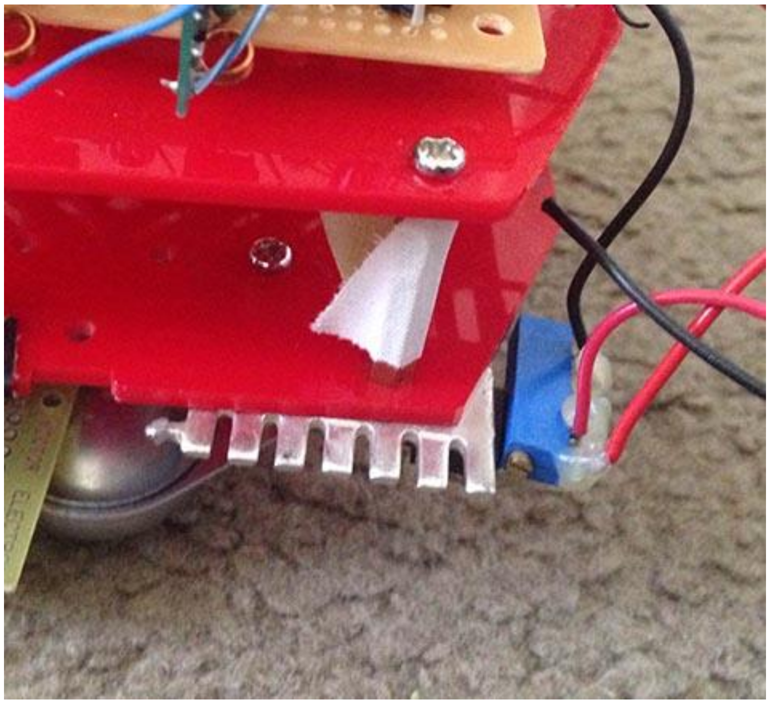
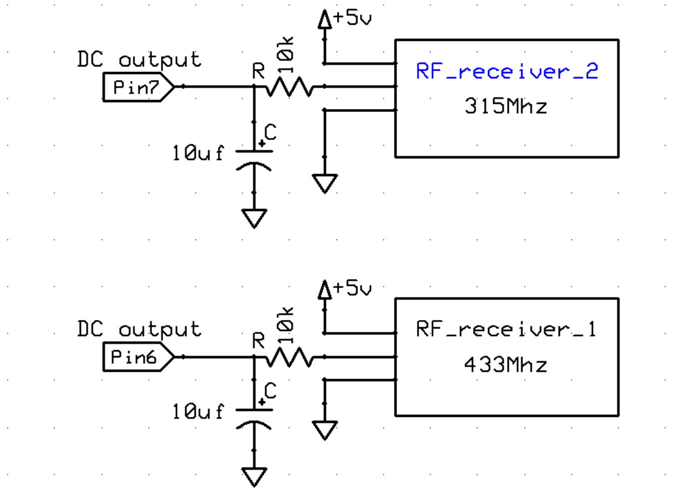


Figure 15: Power Transistor, Potentiometer and Heat Sink for Adjustable DC Motor Voltage

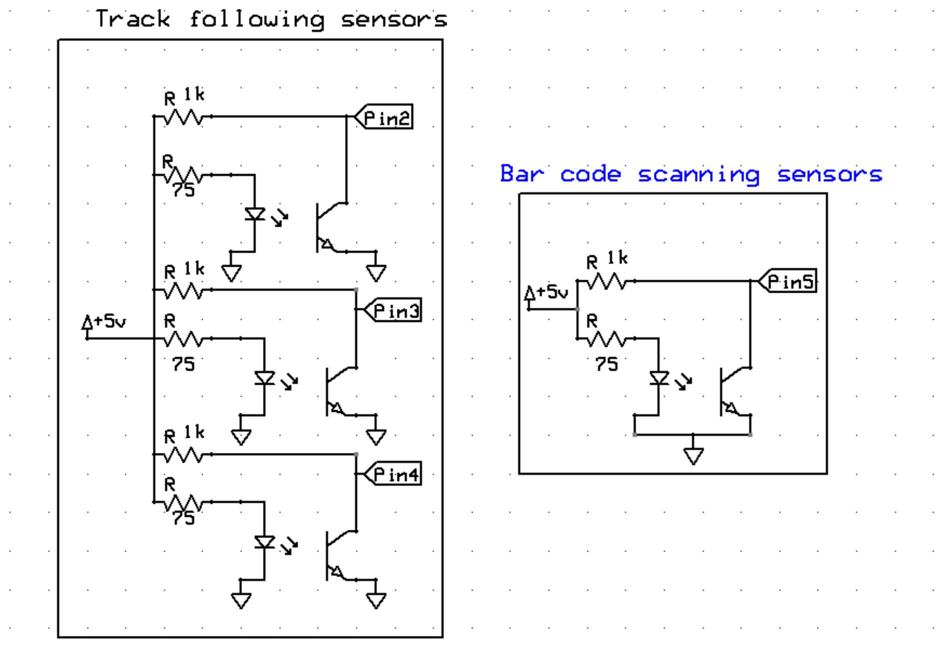
Appendix B – Complete Parts List

Part Name	Part Number	Quantity	price	Vendor
Starter Board Development PIC24FJ64GA002	DM300027	1	$79.00	Microchip
Magician Chassis Kit	ROB-10825	1	Free	Stock Room
RF module 433Mhz	NA	1	$2.50	eBay
RF module 315Mhz	NA	1	$2.50	eBay
LCD Display	HDM08216H-3	1	Free	Stock Room
H-Bridge	L293B	1	Free	Stock Room
16 pin socket	NA	1	Free	Stock Room
10cmX5cm PCB board	NA	1	Free	Stock Room
6cmX5cm PCB board	NA	2	Free	Stock Room
IR Emitters	LIR-204X	4	Free	Stock Room
IR Phototransistors	LT1893-82-0125	4	Free	Stock Room
30 Gauge Wire	NA	~	Free	Stock Room
Voltage regulator	LM7805	1	Free	Stock Room
Power Transistor	2N6487	1	Free	Stock Room
10k ohm Potentiometer	NA	1	Free	Stock Room
5k ohm Potentiometer	NA	4	Free	Stock Room
50nf capacitors	NA	2	Free	Stock Room
30nf capacitors	NA	2	Free	Stock Room
SW diode	NA	4	Free	Stock Room
1k ohm resistor	NA	6	Free	Stock Room
NE556N timer	NA	1	$1.00	Elliott Electronics
Heat sink	NA	1	Free	From Old Monitor
Plastic cassette tape box	NA	1	Free	Stock Room
Lithium Ion Cells	CGR18650D	4	Free	Old Laptop Battery
9v battery	NA	2	Free	Stock Room
Battery clip	NA	3	Free	Stock Room
PS2 Analog Joystick	NA	1	Free	Old PS2 Controller

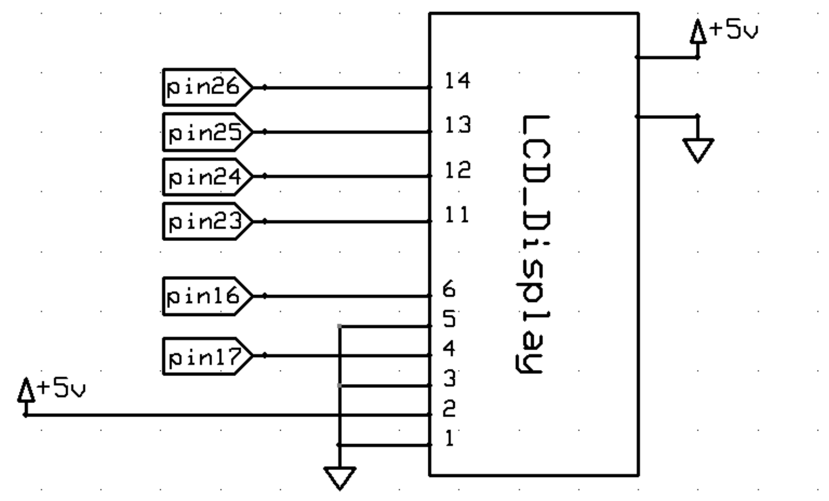
Appendix C – Circuit Diagrams



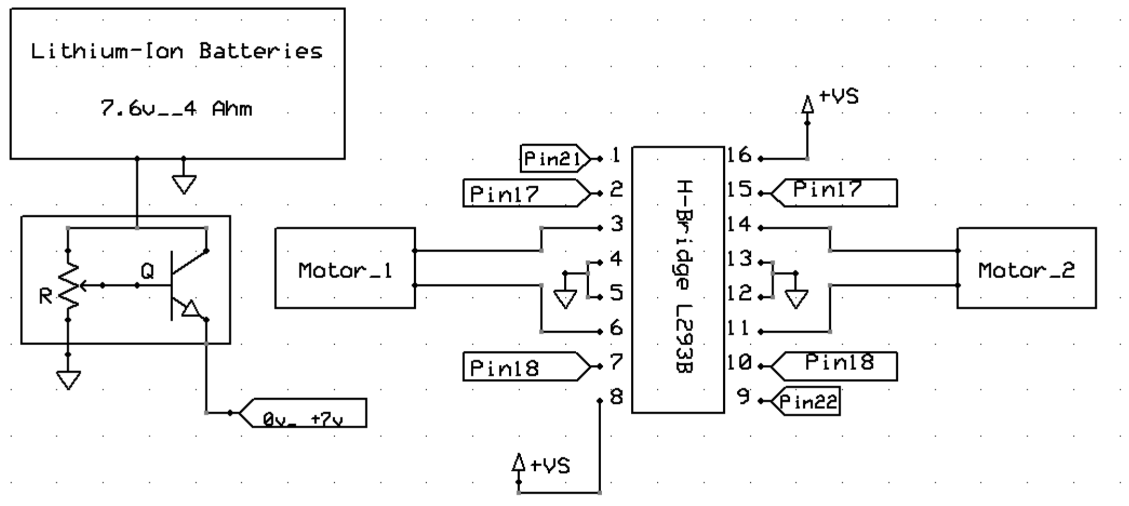
RF Module Circuitry



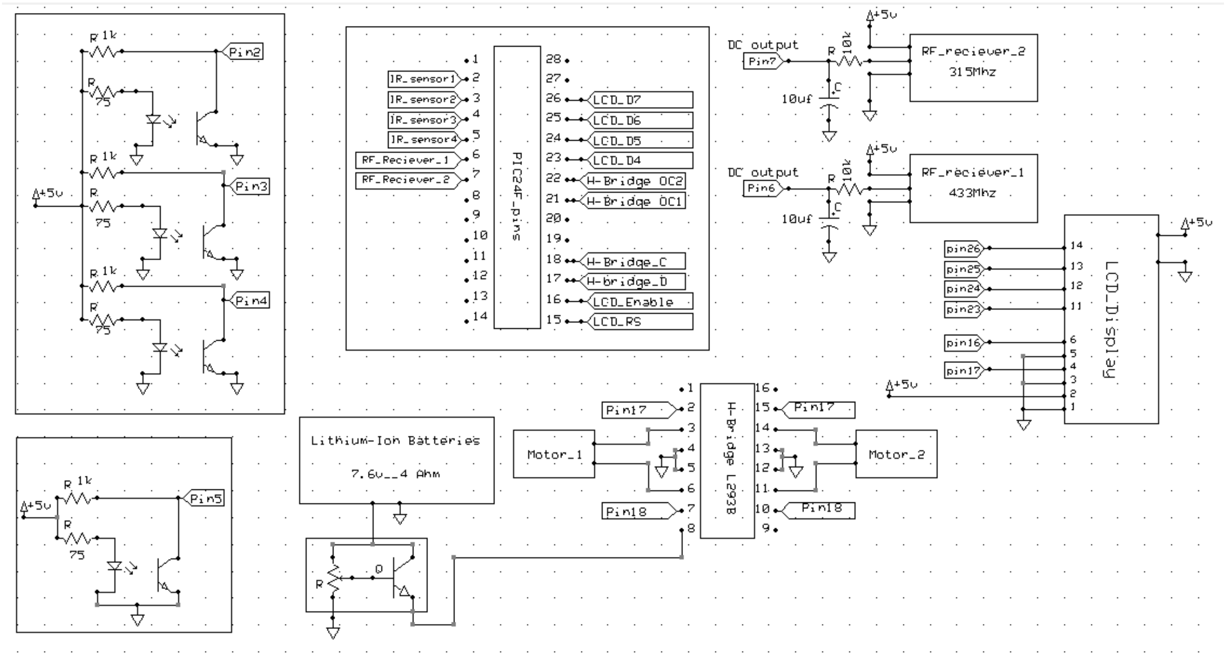
IR Emitter and Phototransisor Circuitry

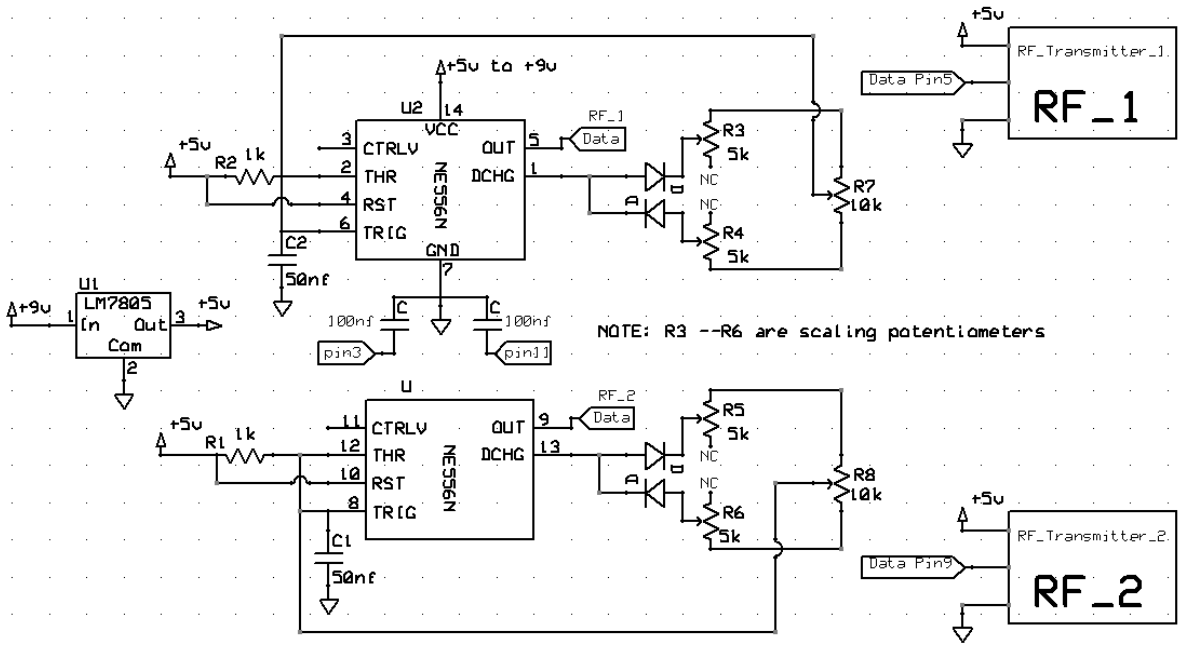


LCD Configuration



H-Bridge + DC Motor + Li-Ion Battery Configuration


Hardware Overview Diagram


External Remote Control Circuitry

Appendix D – Software Listing

MPLAB X and IDE v2.00 and MPLAB IDE v8.92 and Tera Term were used to complete this project.

The follow files will be available for review:

- final.c
- lcd.c
- lcd.h
- p24FJ64GA002.h

This source code will be zipped and provided separately in the d2l dropbox.

Appendix E – Datasheets

NE566N Dual Precision Timer

[1] http://www.ti.com/lit/ds/slfs023g/slfs023g.pdf

Panasonic CGR18650D Li-Ion Cells

[2] http://www.rosebatteries.com/pdfs/Panasonic%20CGR18650D.pdf