Tracking Robot

ECE 372a Final Report

Team 204

Codename Team Brobot

Ben Lacy

Mark Roche

Jovan Vance

Shabeeb Shah

Table of Contents

List of Figures

1. Abstract

Our robot, “Brobot”, was designed with requirements set forth by Dr. Ratchaneekorn Thamvichai to utilize a set of infrared emitters and phototransistors connected to the PIC24FJ64GA002 on a 16-bit 28-pin starter board designed by Microchip to follow a designed track forwards and backwards in the ECE 372a lab. The software in Brobot used analog to digital conversions to read a specific voltage from the black and white lines along the track to successfully move along the track. Another IR emitter and phototransistor pair was utilized to read black, red, and white strips set along the track and display a value on the LCD display mounted on the robot. A speaker was mounted on the starter board to play “The Rains of Castamere” while following the track.

2. Introduction

The robot that was built, affectionately named Brobot, was designed for an ECE 372a project. The idea behind the project was to create a robot that was able to follow a track that is lined with black tape on near-white tile. The track was in the shape of the letters UA. Along with being able to follow the track from end to end, the robot was also required to turn around and follow the lines back after seeing three horizontal black strips. As well as following the track, the robot also had to be able to read data from binary strips placed along the track and display them on an LCD. One additional unique feature was required for the project, for this Brobot plays music while following the track.

The robot was designed using a PIC-24 Micro-controller. Many parts were provided and common to each solution of the problem. These parts were items like the plastic chassis, the motors and wheels, the omni-wheel, the LCD, and in this robots case – the speaker. A general understanding of the workings of a micro-controller may be necessary for understanding how the robot works.

The problem of line-following can be solved in many ways. Two pairs of light emitting diodes and photo transistors could be used in order to follow the line. These two pairs would bracket the line and would adjust to white, keeping the black line centered between them. Brobot was instead designed with three pairs of LEDs and phototransistors. This turned out to be a better solution because of the way the middle pair was prioritized. This led to the robot being able to traverse the track much faster than its two pair counter-parts, as well as helping it effectively reverse in order to find the track again when it gets lost.

There are many different ways to reach the solution when it comes to software design, but this robots design was made in order to work quickly and in compact segments of code that are easily editable. As opposed to having many of the conditional statements chained together the robot has compact independent code segments that are controlled by easily editable thresholds in order to adjust for every sensor including the binary reading. The design of the software for the music is very straight forward with not many different ways to achieve it. That being said there are many ways to solve the requirements of Part A in general.

3. Technical Discussion and Details

3.1 Design Details

The robot is capable of following a track with a black line, read binary cards and play music simultaneously. The robot follows a black track by using three couplings of an IR Emitter and a phototransistor while also reading two binary cards off to the side using the same coupling. While a binary card is being read, the robot will print the values to the LCD screen.

All Hardware:

The body of the robot is a Magician Chassis (ROB-10825) kit with three 1.8” x 1.8” PCBs, a speaker (SPEAKER MODEL), two DC motors paired with two wheels and one steel ball for stability – is there any other skeletal parts missing?. The guts of the robot is four 47ohm resistors, four 410kohm resistors, four IR emitters, four phototransistors, approximately 4 ft of 30 gauge wire and one 10ohm resistor for the speaker. The brain of the robot is an LCD, H-Bridge and a Microchip PIC24F micro-controller.

Robot Layout:

Placed on the front of the robot, we have three sensor couplings soldered to a 1.8” x 1.8” PCB for line reading. To ensure that the readings from the phototransistor were accurate, we placed the phototransistors exclusively adjacent to one IR emitter. (Fig 1) This gave us confidence that the phototransistors were getting light from one IR emitter. We soldered another sensor coupling to a second 1.8” x 1.8” PCB on the right side of the robot to read binary cards.

The LCD screen and H-bridge were place on the micro-controller held by header pins and wire wrapping. The screen and the H-bridge were not soldered to the board. Finally the speaker was placed on the third 1.8” x 1.8” PCB wire wrapped in series with a resistor (Fig 2). The DC motors and wheels are under the chassis in the same placement as the micro-controller. The steel ball is also under the chassis placed in the middle back region under the speaker.

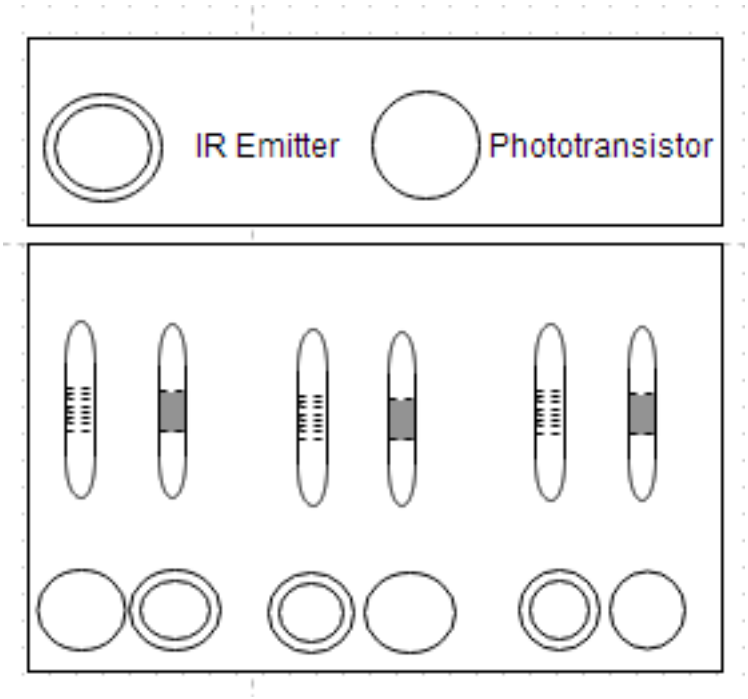

FIGURE 1: CIRCUIT ARRANGEMENT

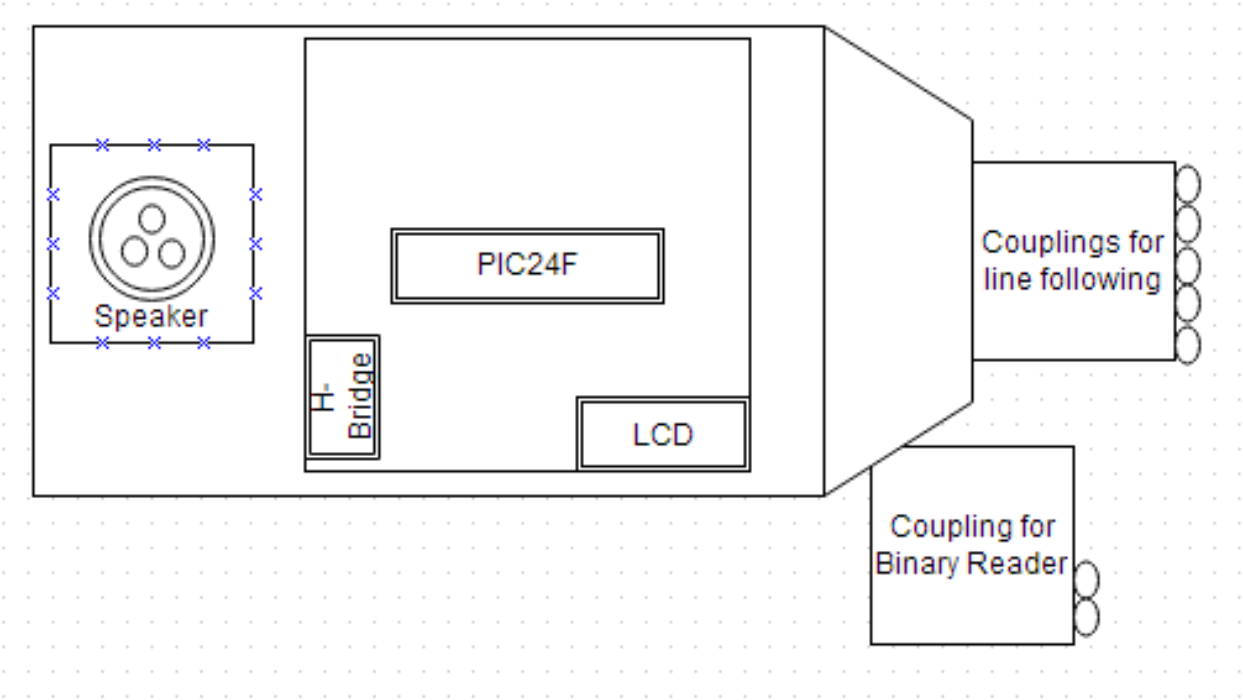

FIGURE 2: ROBOT HARDWARE ARRANGEMENT

Moving the Wheels:

We use RB10 and RB11 as PWM output pins for each motor. RB0 and RB1 pins are used to control forward and backward movement. In order to save pins, we wire wrapped two inputs to the H-bridge to one pin allowing us to use two pins total for the H-bridge. To set the speed of each wheel, we adjust the OC1RS and OC2RS registers. To have the wheels moving at maximum speed, we set the OC1RS and OC2RS to the PR2 value. The PR2 value is the max value the TMR2 can count up to. To slow down the spin of the wheel, we would set OC1RS and OC2RS to a fraction of PR2. To move the wheels in reverse, we changed the value of the LATB0 and LATB1 bits. If LATB0 is equal to LATB1 then the wheels will not move. We tested each combination of LATB0 and LATB1 bits to find which setting allows the wheels to move forward and backward. Since we share pins, there is no setting of bits that turns one wheel on and one wheel off, therefore to achieve this, we set OC1RS or OC2RS to 0.

Line Following:

Hardware:

The line following PCB board (Fig2) consists of three phototransistors and three IR emitters. The phototransistors are soldered to the PCB board facing the ground while also wire wrapped to a 410kohm resistor connected in series. Each IR emitter is also soldered to the PCB board in close proximity to one phototransistor. Each emitter is wire wrapped in series with its own 47 ohm resistor. At the terminal closest to the robot, each (IR R'S) resistor is wire wrapped together with each terminal farthest from the robot of the phototransistors. This allowed us to have to

only wire wrapp one of these terminals to ground on the micro-controller. We used the same technique to apply the 3.3 V supply to each circuit on the PCB. At the terminal closest to the robot, each (PHOTO R) resistor is wire wrapped with each terminal of the IR emitter that is farthest from the robot. (Fig 3)

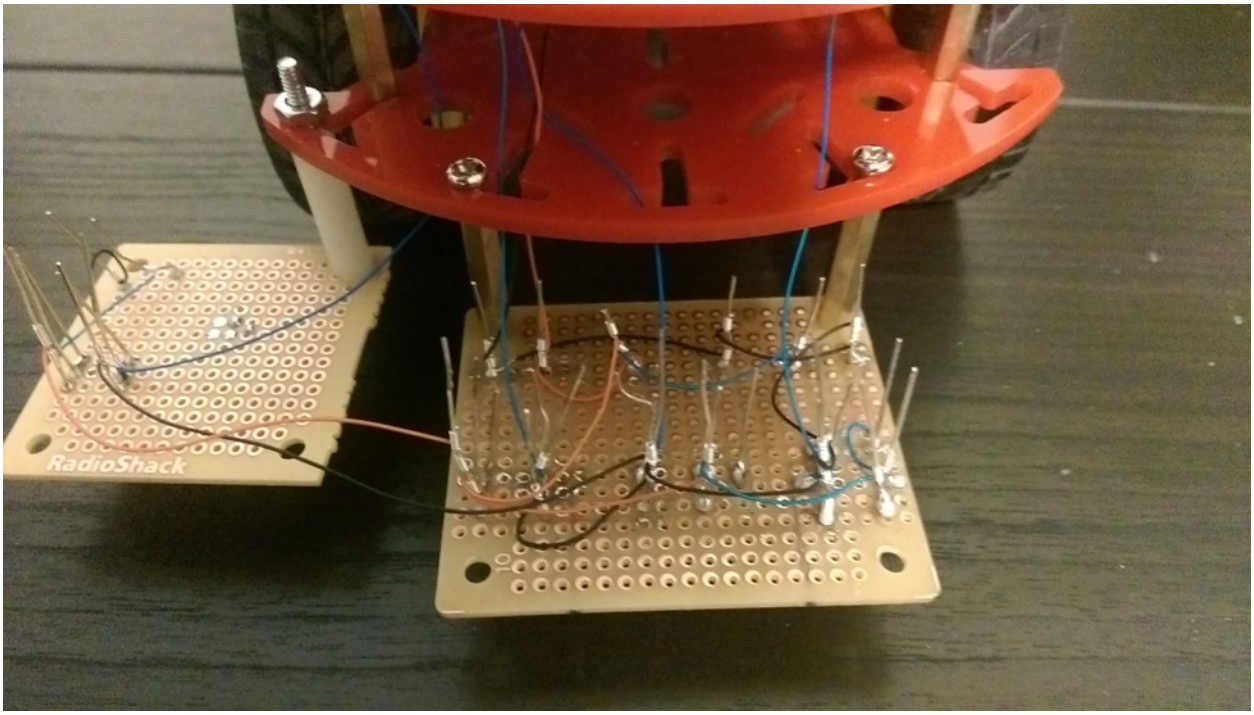

FIGURE 3: WIRE-WRAPPING LAYOUT

To get information necessary to following the line, we wire wrapped each phototransistor to a unique pin on the micro-controller. The pins we use for the line following phototransistors are RA0, RA1 and RB3.

Software:

RA0, RA1 and RB3 are enabled as inputs and used for analog to digital conversion. To achieve this, the TRIS bit for each pin were set to one and the AD1PCFG for each pin were set to 0. After the ADC initialization is done, the code enters an endless loop. In the loop, the code grabs the values stored in ADC1BUF0, ADC1BUF1 and ADC1BUF3, and multiply each value by 3.3/1024 in order to convert each value into a voltage. The program waits for the button on the board to be pressed before it will move. Once the button (RB5) is pressed, the program will go into a CN interrupt and assert a flag. Once the flag is one, then the program runs through a series of ‘if’ statements to determine the speed and/or directon the wheels should be turning. If the middle sensor is reading black, regardless of what other sensors are reading, the robot is told to move forward. Other wise, the robot will move in the direction of the sensor that is reading black. If all the sensors are reading black, then the robot will move backwards until a sensor reads black.

The program knows whether a sensor is reading black or white based on defined constants. Each sensor has its own black threshold. We determined each threshold by individually finding the lowest voltage the sensor reached while reading black. We felt we were forced to do things this way because we saw that each voltage reading were too unique.

Once the sensors all read black, we assert a flag and increment a counter. The flag is de-asserted when one of the sensors reads white. This signifies that the robot has passed one black strip. After the second black strip is read, the robot starts turning around by turning off the right wheel by setting OC2RS to 0for a defined amount of time. Once the time is up for turning around, the turn around operation is over and the robot must me hovering over a black line or else the robot will be lost.

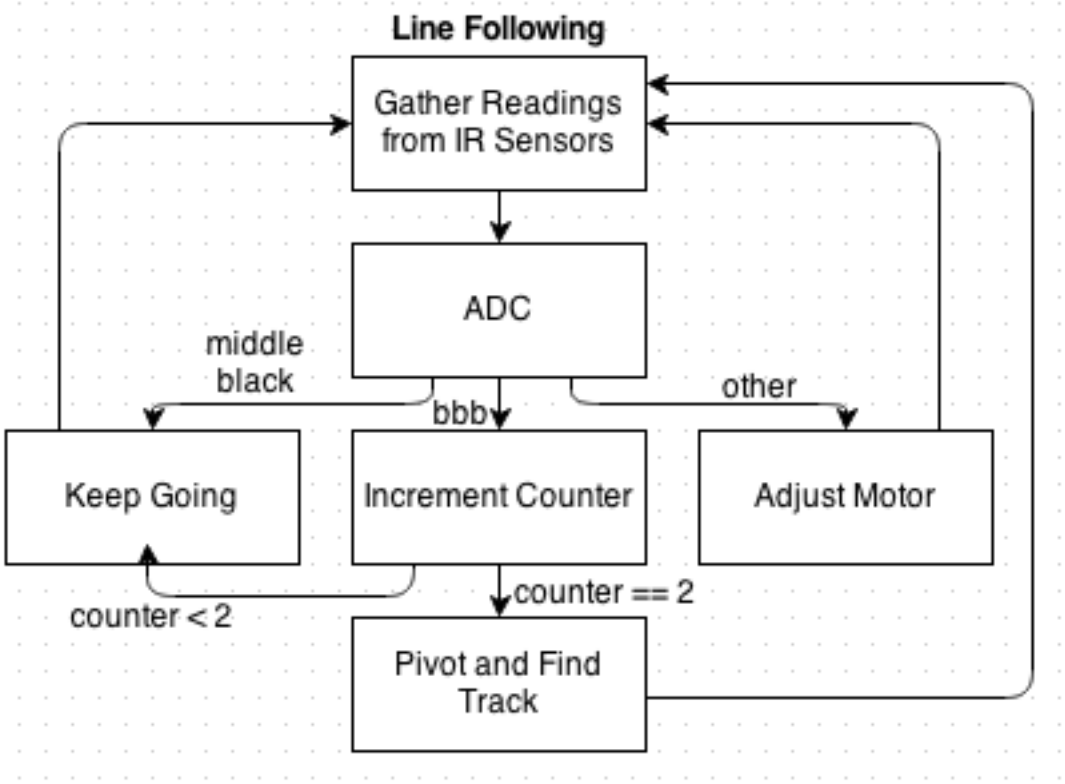

FIGURE 4: LINE FOLLOWING SOFTWARE

Binary Reading:

Hardware:

A 1.8” x 1.8” PCB is used to assemble the binary reader circuit. The circuit design is identical to one coupling on the line reader PCB. The coupling is placed to the right side of the PCB to allow more space between the right wheel and the binary card. (BINARY PCB PIC) To save wire, we connected the 47ohm resistor to the 47ohm resistors’ ground terminal and connected it to the front terminal of the phototransistor. The back terminal of the 410kohm resistor and the front end

of the IR emitter are both wire wrapped to the front terminal of the closest terminal of an IR emitter to be connected to 3.3 V. The phototransistor is connected to RB2 pin on the micro-controller.

Software:

RB2 is enabled for an analog input and has an identical initialization of RA0, RA1, and RB3. Once the program enters the loop, it graps the information stored in the AD1BUFF2 and converts that into a voltage by multiplying it by 3.3/1024. After the button is pressed, it then checks if the sensor is reading a black value. After the sensor passes through the first black bar, it will print the value of the next dark bar. A red bar would signify a 0 and a black bar would signify a 1. To determine if the sensor passed through a red or black, the program keeps track of the darkest voltage read before the sensor reads white. If the darkest value is above the defined black threshold, then the LCD will print 1. If the darkest value is above the red threshold but below black, then the LCD will print 0. The black threshold is the lowest black value we measured with the binary sensor. The red threshold was the lowest voltage the sensor output while hovering a red bar. The white threshold is the highest voltage that the sensor outputs while hovering over white. These thresholds are all saved as #DEFINEs. Once the program has read four values for binary, the binary count restarts and the second binary card will be printed on the second row on the LCD screen.

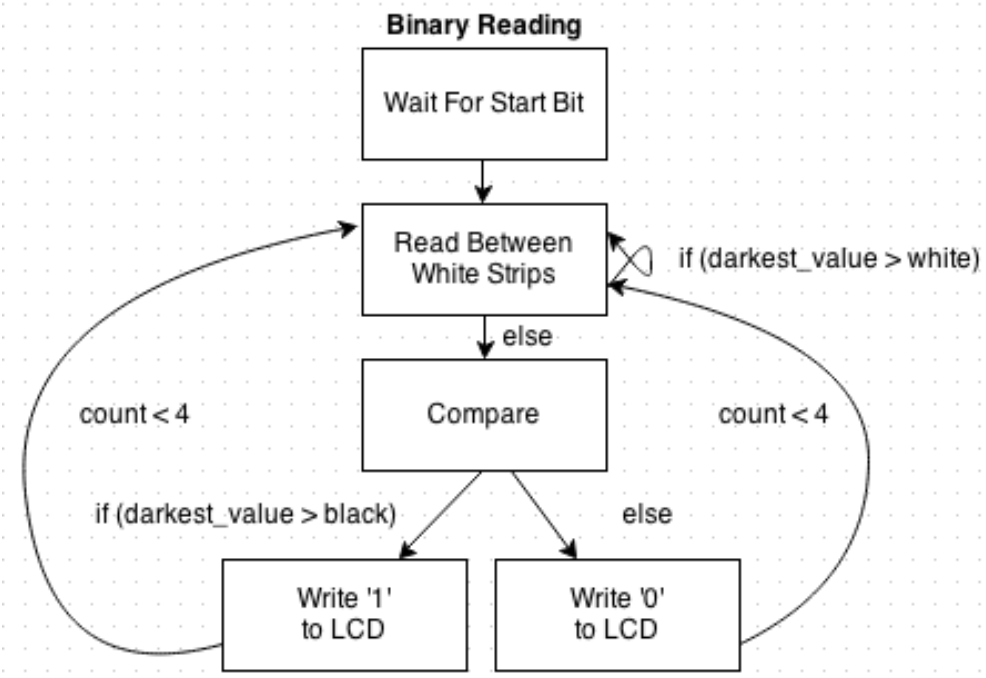

FIGURE 5: BINARY READING SOFTWARE

Music:

Hardware:

The speaker we used was a .5 watt 8ohm speaker that we connected with a 10ohm resistor. The resistor was connected to the positive terminal of the speaker while the negative terminal is connected to ground. The free terminal of the resistor is wire wrapped to a pin on the PIC. Both the resistor and speaker are place on a 1.8” x 1.8” PCB. Each connection is wire wrapped.

Software:

In order to produce a sound to the speaker we produced a pulse width modulation using the timer 3, PR3, and output compare 3 registers. The PR3 and output compare 3 register would be updated to change the note that the speaker played. In order to play the note for a proper duration the timer 4 interrupt was used. At each instance of a timer 4 interrupt, the PR3 register would read a new value from an array statically programmed with the specific frequencies of the notes of the song “The Rains of Castamere” and the timer 3 register would be reset. In order to prevent a longer note from ringing or playing repeatedly a simple control flow statement checked to see if the current note in the array is the same as the previous note in the array; when this was the case the PR3 register would not be reset. The code would continue to iterate through our song array and when the end of the array was reached the song would restart.

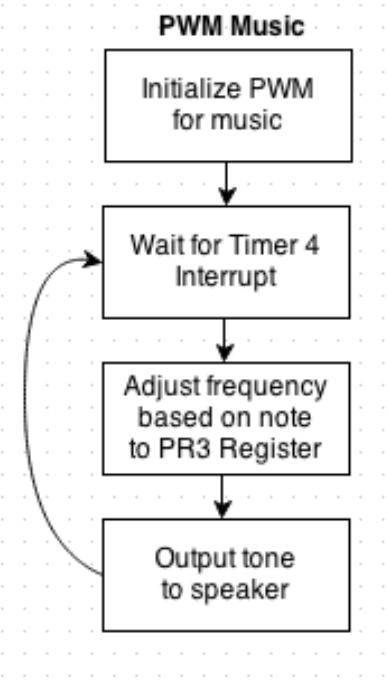

FIGURE 6: MUSIC PLAYING SOFTWARE

3.2 Design Verification and Testing

To design the circuit for reading the line, we first started on the bread board. We tested multiple resistors to connect with the phototransistor and IR emitter. The goal was to find resistor values that would give us enough parity to determine a ‘white’ reading and a ‘black’ reading. Once we found a circuit that worked for one coupling didn’t work when we placed multiple couplings close together, we had to start testing circuits on the PCB instead of the bread board. Testing resistor

values on the PCB took a little longer but was more effective because the readings we were getting were going to be closer to the readings we wanted with the final configuration.

To determine the black and white thresholds of each sensor, we printed each voltage reading from each sensor on the LCD screen. We then simulated each scenario the robot might go through while driving on the track. For instance, we would record the voltages on the LCD screen when the robots right sensor is over black and the other two are over white. The goal was to establish the lowest voltage of each individual sensor while that sensor was the only sensor hovering over black. To verify that the readings we found were correct, we would define them in our code and run the robot on the track. If the robot didn't run the desired way, we would try changing the black threshold of the individual sensors.

To test the turn around functionality of the robot, we applied different lengths of time to turn around. The ideal turn around would be long enough to get back on track but not too long where it is facing the wrong direction. It was easy to see if the time was too long or too short by looking at the behavior of the robot. If the robot turns and then go in reverse, then we know we need to increase the timer. If the robot turns and gets stuck in the horizontal strips, then we know that the timer is too long.

3.3 Discussion of Problems Encountered

Hardware Issues

One of the very first issues we came to face was the control of the motors based on the inputs. At the initial stages of the project, we had used four different pins as inputs to the two motors (as opposed to only two pins later in the project). When initially programming the board to have the robot go forward, backward, and idle, the combinations of inputs that we thought would get us the desired functionality did not do what we thought they would do. This led us to try every different combination of inputs to observe what each of the combinations did. Since we were using four different pins, this corresponded to 16 possible combinations that needed to be checked. We eventually did find the needed combinations, however we didn't feel like we had as much control over the motors as we wished. This problem most likely came about to a possible mis-wiring or misunderstanding of how the inputs affected the motor.

The next problem encountered was what we determined to be a blown out IR Emitter/ Phototransistor combo. After the IR Sensor/Emitter array circuit had been completed, we started checking threshold values for different colors. After about an hour of testing, the left sensor pair began to behave in odd fashions. For example, even though the pin was not connected to the sensor, the board seemed to be reading values since the values being printed on the LCD were changing enough to warrant the idea. We also noticed that when we did connect the pin and the sensor, the left sensor voltage jumped up to 5V, but also brought up the other sensor readings by about 1-1.5V. We began to debug the issue and determined that it couldn't be the pin since when we connected the suspect pin to another sensor, proper readings were printed the LCD. This led us to believe that the problem most likely was present in the circuitry. We also were a little confused when we severed the left sensor pair from the rest of the circuit but still observed similar behavior with the erratic readings and reading when not connected. After consultation with the professor later in the week, it was determined that we replace the left sensor pair and their respective resistors. Upon installation of a new sensor pair, readings reverted to normal behavior, thus solving the problem. The old sensor pair was most likely broken when input

voltage was changed to 5V in attempt to get a bigger difference between white and black readings. The components were picked based on an input voltage of 3.3V, thus most likely got overloaded when the change was made.

There was a similar issue towards the end of the project where sensor values seemed to be stuck at certain values. At this point, we first thought to look to see if there were any shorts on the circuit. Due to the way our sensor array circuit was constructed, shorts were easily produced when legs of the same component touched. This situation was easily produced when transporting the robot in its bag. That being said, the problem was also easily spotted and fixed seeing as the solution was merely spacing the component legs apart. Once this was done, the robot exhibited expected behavior.

Software Issues

Although we ran into many hardware related issues, we did not run into as many software-related issues. One of the few problems encountered with the software was the jerky movement of the robot. This was not caused by hardware, since the hardware was doing its job by providing the voltage readings. The software was the part that had to interpret the readings and determine what to do based on said readings. At first, the software was made to prioritize going straight, and then turning off one of the motors based on if the left sensor or right sensor was reading black. This produced very jerky motion, which could potentially be an issue when it came to binary reading. The solution to the jerky motion was breaking up our turns into hard turns and soft turns. Soft turns did not turn one motor off completely, but instead kept it on at a much slower speed than the other motor, resulting in less dramatic adjustments. The thresholds for hard and soft turns were different, so there was no conflict on what kind of turn to take. Addition of the soft turns made the track following much smoother, although did not get rid of 100 percent of the jerky motion.

The other software related issue encountered was the the u-turn the robot had to execute in order to get back to the starting point. Our first idea was to have the robot pivot in place, i.e. have the wheels spin opposite each other, to execute the turn around. However, as mentioned before, our lack of complete control over the motors resulted in there being no possible input combination where the wheels spun opposite each other. This became irrelevant later when we had to condense pins to be able to add components to the robot. The condensation of the pins made it so that inputs to the H-Bridge were tied together, therefore taking away the possibility of pivoting in place. The solution to the turn around was to turn one of the motors off for a timed amount, then have the robot go straight until it caught the track again. This worked much better, however the time spent turning varied on battery life and usually came up short. After much testing, the rather trivial solution we came up with was to execute the turn once the second line is crossed instead of the third. This executed the turn earlier, thus giving the robot less real estate to cover to get back on track. Upon making this change, the robot got back on track after turning around almost every time.

Audio Issues

Besides line turning, the other issues we had were in producing music using PWM. This aspect of the project was actually working early on in the project albeit on a separate board in its own file. When it came time to integrate the code, we realized that the music code would have to

change, since it used a couple of loops that when integrated with the rest of the code, could potentially throw off line following. The solution to this was to use a timer interrupt to replace the functionality of the loop. Even with this change, the music would not properly play, and in addition, killed the rest of the program. After commenting out different blocks of code, we determined that reading values from the ADC buffers was somehow conflicting with playing the music and moving the robot. We eventually fixed this by moving all of the music code from the while(1) loop to the inside of the timer interrupt. This fix enabled the music playing and track following to coexist.

The last problem we faced with the audio was actually pretty minor. Although the audio the robot played as it was going around the track was audible, it had space to be louder. To do this, we used a npn transistor and a 1k ohm potentiometer to have volume control over the speaker. While we accomplished the goal of having a much louder speaker, the robot's movement was now heavily affected. With the amplified volume, the wheels on the robot barely turned when being held in the air and would not move at all when placed on the ground. This led us to scrap the newly put together circuit in favor of saving time and line following functionality. By scrapping the transistor and potentiometer, the robot was able to move again while playing an audible song on the speaker.

4. Conclusion

The tracking robot "Brobot" met almost all the requirements set out for it. The robot can consistently follow the track from end to end as well as U-turn and come back. It can effectively play music as it travels around the track. The robot can easily read strips of binary data that are placed alongside the track and print them onto the LCD.

A few goals set out for the robot were either not met, or could have been accomplished more successfully. The first being that the robot does not use the Raspberry Pi in order to give a first-robot-view of driving around the track. UA Wifi ended up being the blocker that stopped the Raspberry Pi from being implemented. While the line following was superb and helped the robot travel very fast along the path, its U-turn functionality was slightly too dependent on battery life and could have potentially been changed to function more reliably on any amount of power. Lastly, while the music was played audibly while the robot drove around the track, the speaker took a lot of power away from the robot leading to a much shorter traversal time. It is possible there is a solution for the speaker that pulls less power from the robot. Overall the robot was a success and met the requirements set out for the project.

5. References

N/A

6. Appendices

6.1 Circuit Diagram

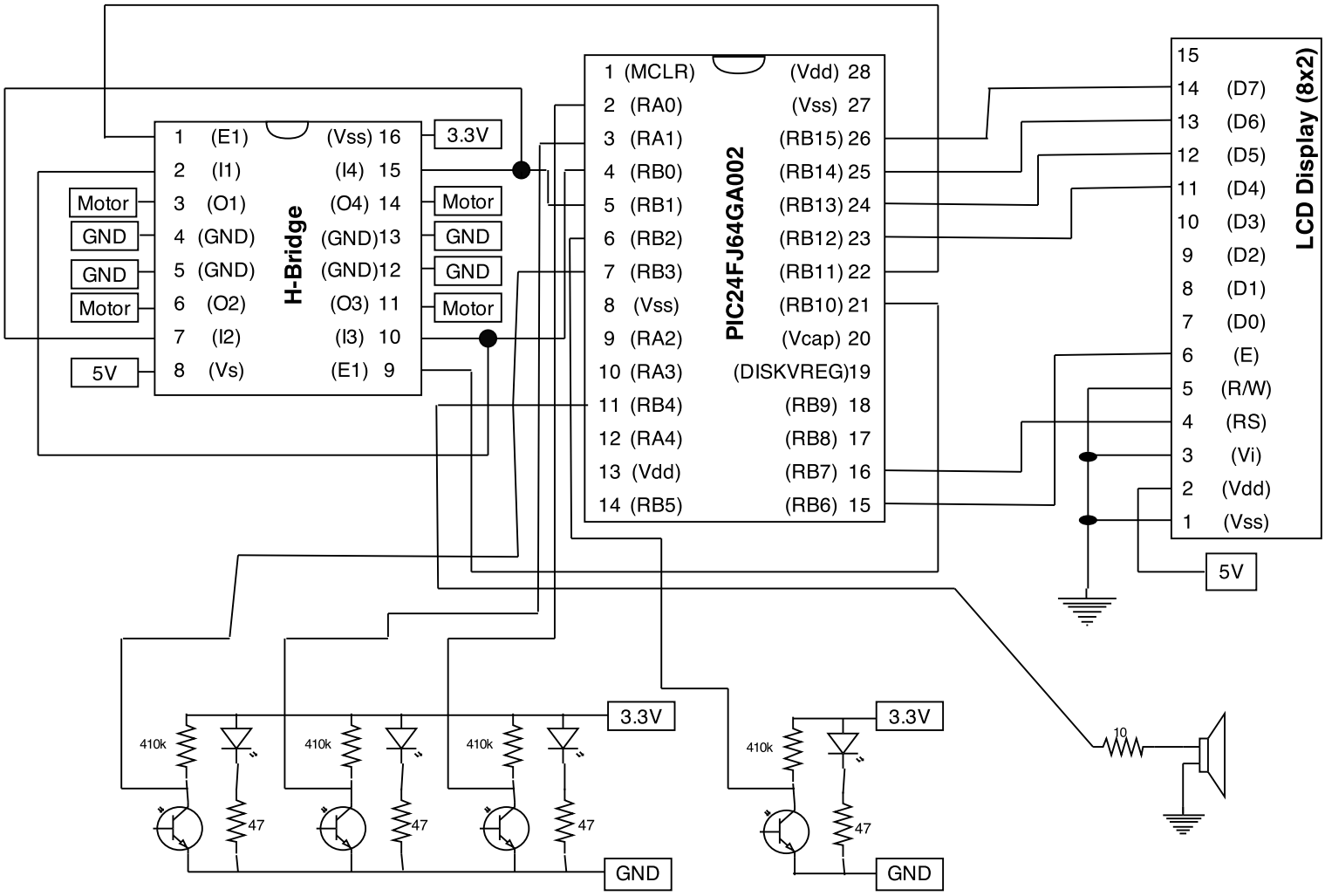

FIGURE 7: CIRCUIT DIAGRAM

6.2 Complete Part List

Part Name	Quantity	Price	Source
Magician Chassis Kit	1	Free	Stockroom
PIC24F Microchip Starter Board	1	$79.99	Microchip
PICKit 3 Debugger	1	$44.95	Microchip
8x2 LCD Screen	1	Free	Stockroom
Phototransistor	4	Free	Stockroom
IR Emitter	4	Free	Stockroom
410kΩ Resistor	4	Free	Stockroom
47Ω Resistor	4	Free	Stockroom
10Ω Resistor	1	Free	Stockroom
30 Gauge Wire	~4ft	Free	Stockroom
H-Bridge	1	Free	Stockroom
16-Pin IC Holder (Wire Wrap)	1	Free	Stockroom
Battery Clip	1	Free	Stockroom
.5W 8Ω Speaker	1	Free	Stockroom
Prototyping Boards	3	Free	Stockroom
9V Rechargable Battery	2	Free	Stockroom

FIGURE 8: PARTS LIST

6.3 Software Listing

The following files were written and programmed into the PIC24 micro-controller for the final demonstration for parts A, B, and C. Each files can be viewed in the submitted zipped folder.

- FinalProject_v2.0.c
- lcd.c
- lcd.h

6.4 Pictures of Robot

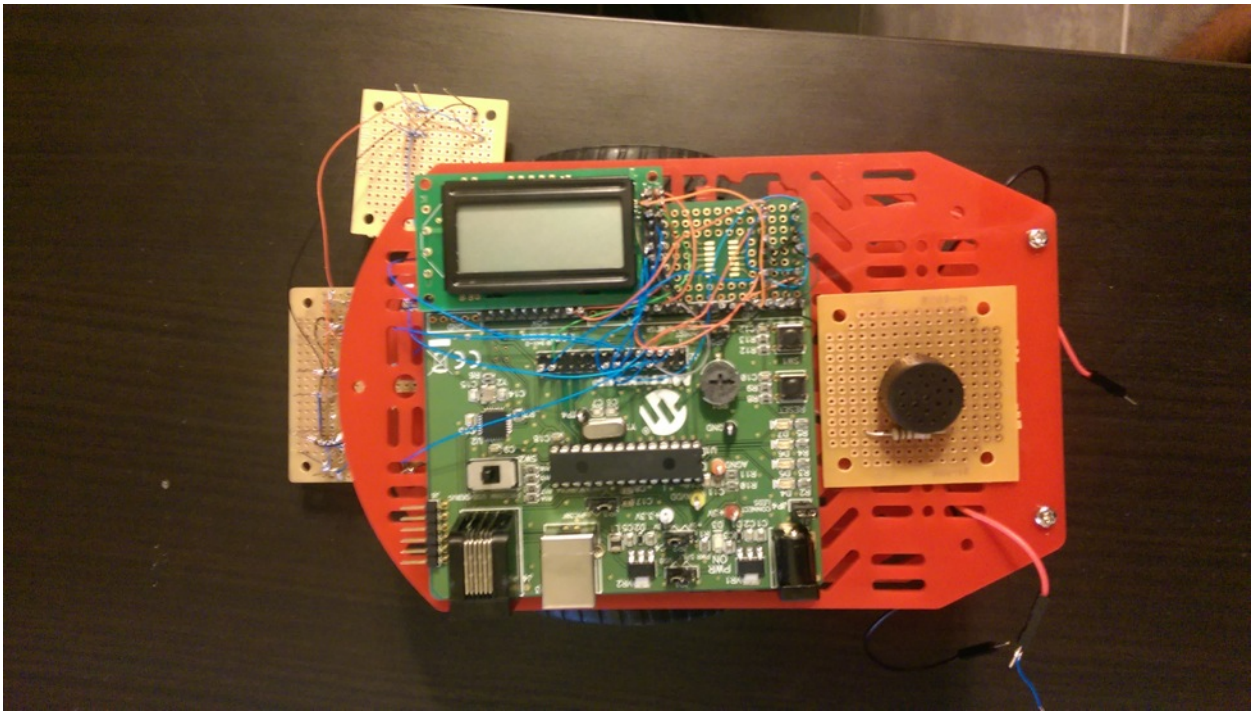

FIGURE 9: ROBOT TOP VIEW

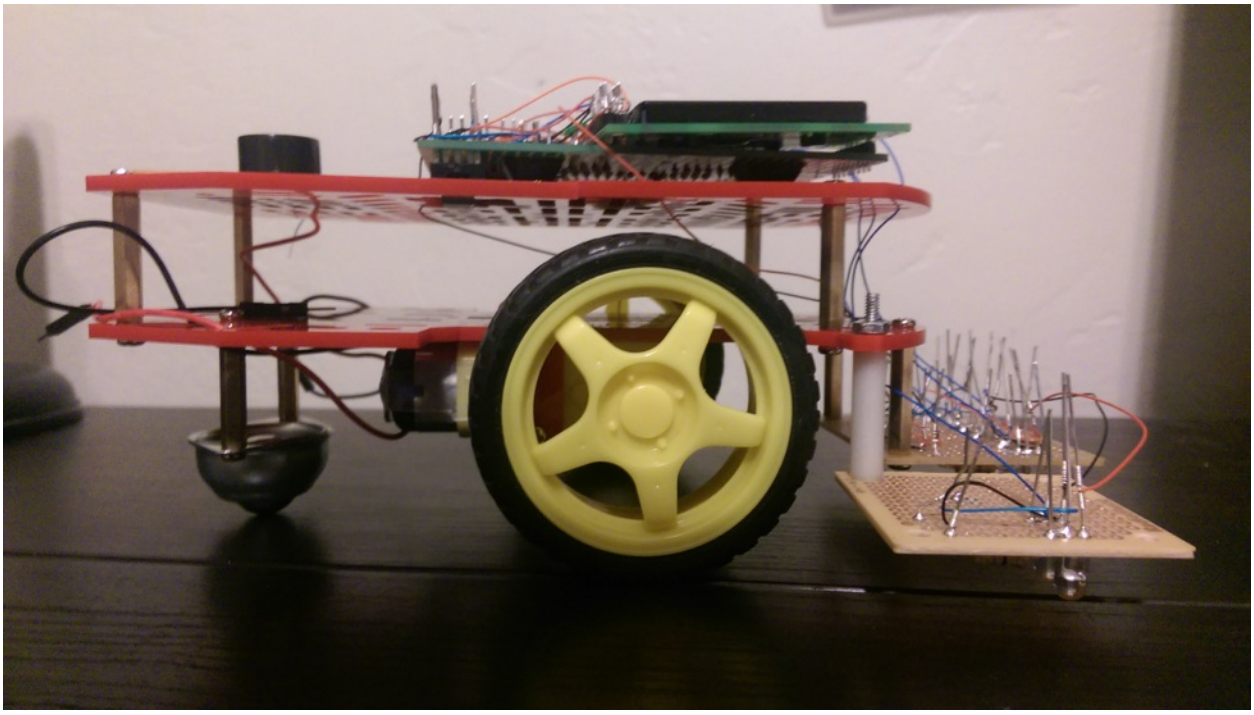

FIGURE 10: ROBOT SIDE VIEW