

Table of Contents

Abstract	2
Introduction	2
System Diagram	3
Sensors	3
<u>Wall Sensors</u>	3
<u>Frequency to Voltage Converters</u>	4
<u>White Line Sensor</u>	7
<u>Flame Sensors</u>	8
<u>Tone Decoder</u>	9
Fire Suppression	10
Circuit Board Design	11
<u>Frequency to Voltage Converters, H-Bridge, and 5V Regulator</u>	11
<u>White Line Sensor</u>	12
<u>Tone Decoder</u>	12
HC12 Modification	13
Software Design and Testing	15
Conclusion	20
Appendix A	21
<u>Code</u>	

Abstract

This semester our junior design class was tasked with building a robot to compete in the Trinity College Fire-Fighting Home Robot Contest, as well as our local version of the same contest. The robot was required to navigate a mock up house (without touching the walls), find a candle, put it out, and return home. The robot was required to be autonomous, have a closed loop motor control system, and sense its' environment (non dead reckoning).

Introduction

The first couple junior design classes were spent in lecture about the theory and implementation of various aspects of the robot. We spent this time, a week or two, considering our design. The first phase of our design was the mechanical aspect of the robot. The second phase was the circuitry. The third phase was actually coding and testing of our robot.

We were given a temporary plate to which we were to get a general idea of how we wanted to build our final robot. We were given a plate that had previously been used, so it was really difficult to mount anything where we need them to be. Most of the first part of the semester we had tape and glue holding everything together. When we finally received a new plate we were able to mount everything in the proper place with out any trouble. If given the opportunity to build another robot, we would start with our final plate to make better use of our time. The circuitry phase was reading and testing of all the possible sensors and circuits. Finally, the third phase was the actual writing code and testing the robot on the tables, in the hallways, and in the maze.

System Diagram

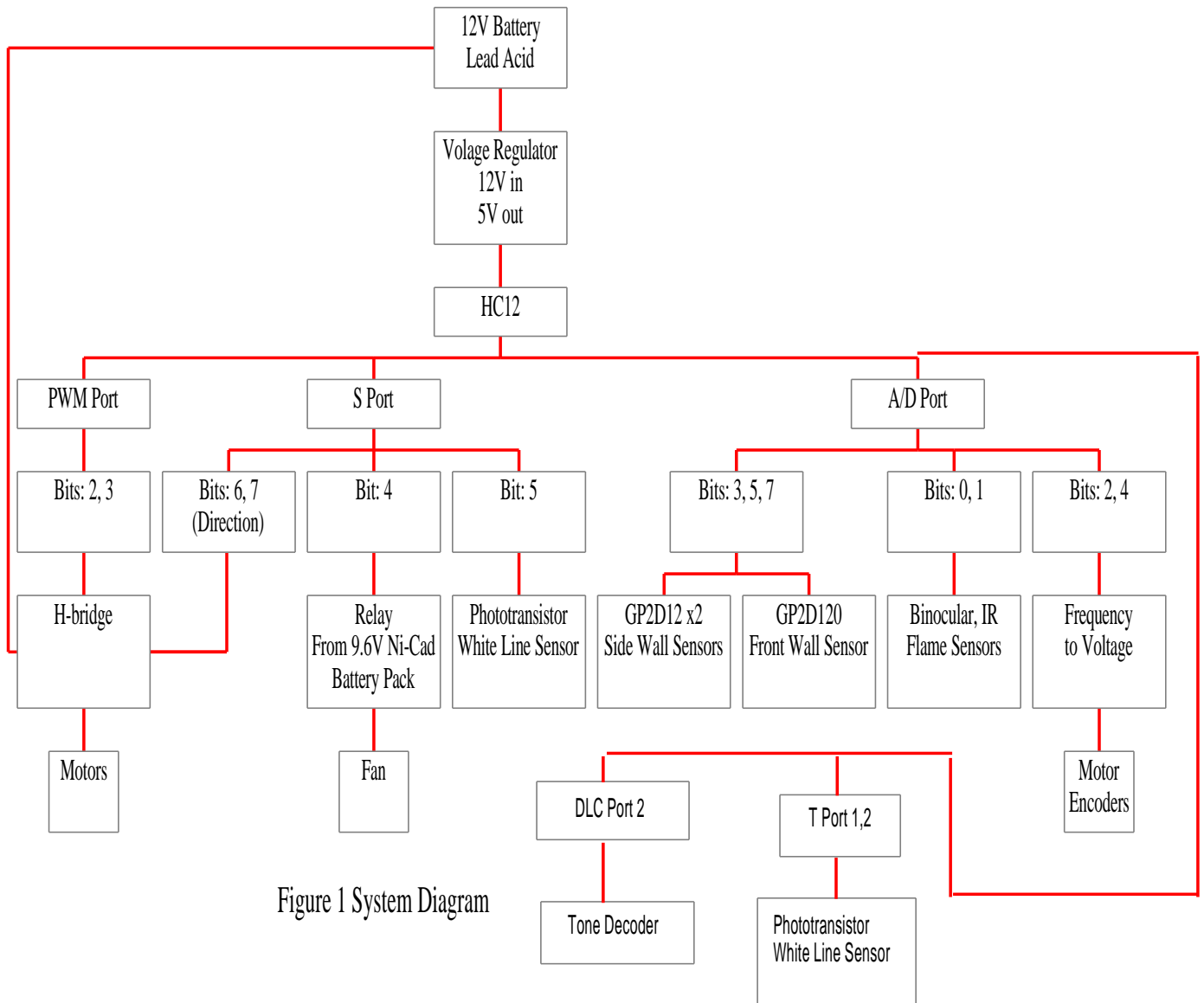


Figure 1 System Diagram

Sensors

Wall Sensors

In order to navigate through the maze, we needed some sort of sensor to tell us where the walls were. The main part of our code also depended on seeing a left wall

through a sensor. The sensors we chose to use were the Sharp GP2D120's and the GP2D12's. We chose to use the 120's since their range of view was on the short side. It had a range of 4cm to 30cm, which provided more than a comfortable region since we used a left wall following algorithm. We were lucky to select two 120's that were had very similar data curve. This was helpful so that in the event if we needed to switch the left and right wall sensors for any reason, we could. (see Figure 2) Together, the pair of these sensors helped keep our robot centered as it traveled through the hallways.

For our front sensor, we used the Sharp GP2D12. This sensor had a range of 10cm to 80cm, which was enough for telling us when our robot got too close to a wall as it approached it. This larger range was necessary, so the robot would see the wall long before it had to adjust the motors to avoid making contact. The data for this sensor can be seen on Figure 3.

We chose these particular sensors because they worked on the infrared section of the spectrum. This was beneficial because outside light and colors would cause little or no interference.

Figure 2 GP2D120 Sensors

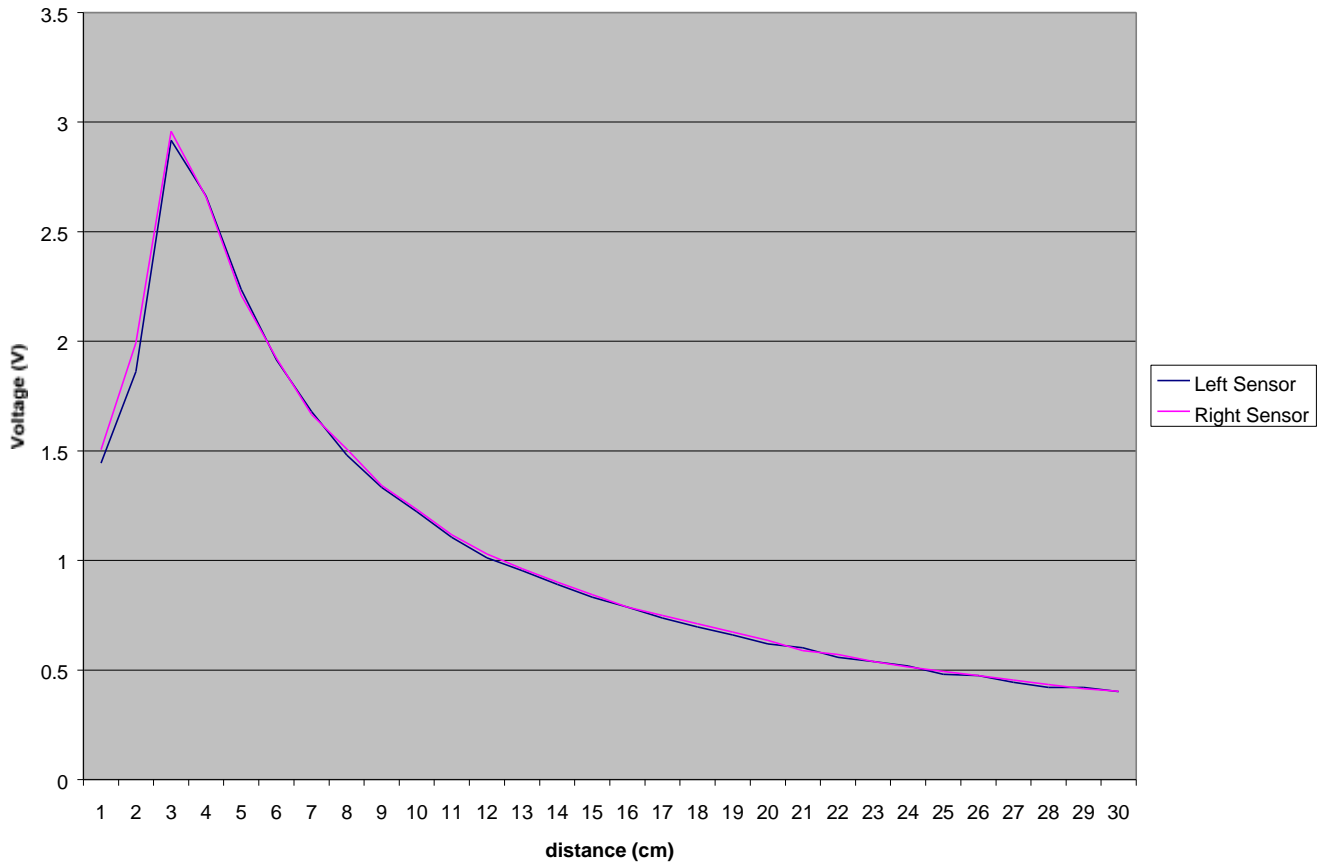
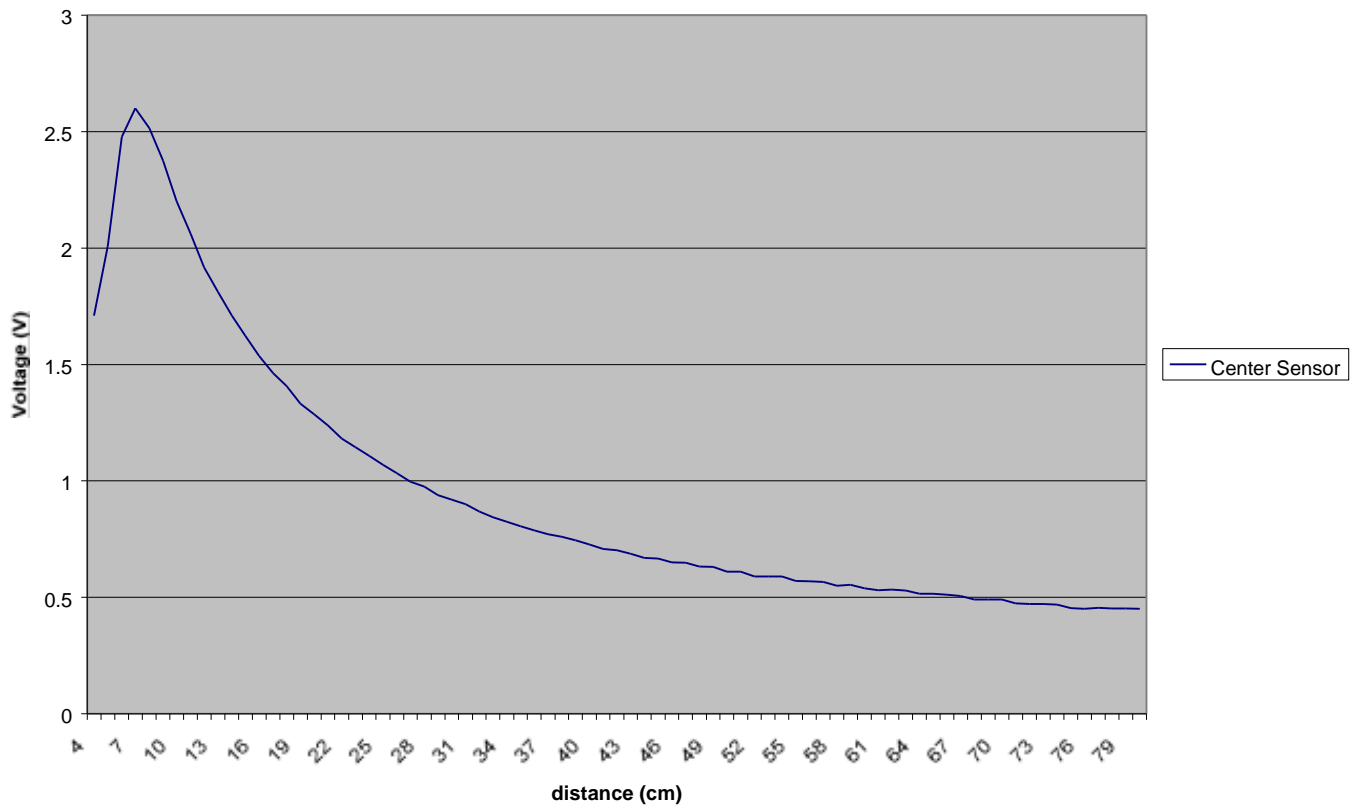


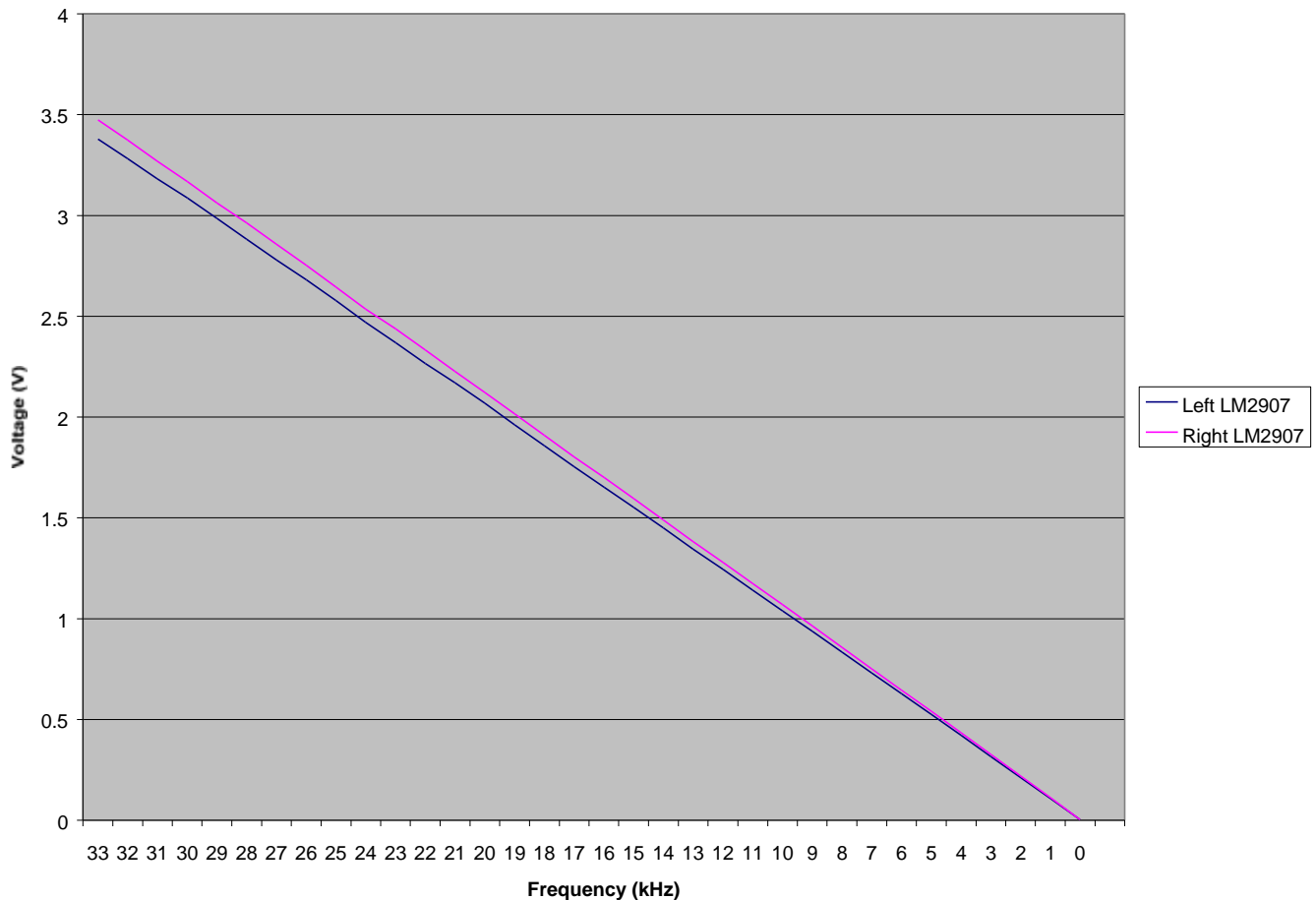
Figure 3 GP2D12 Sensor

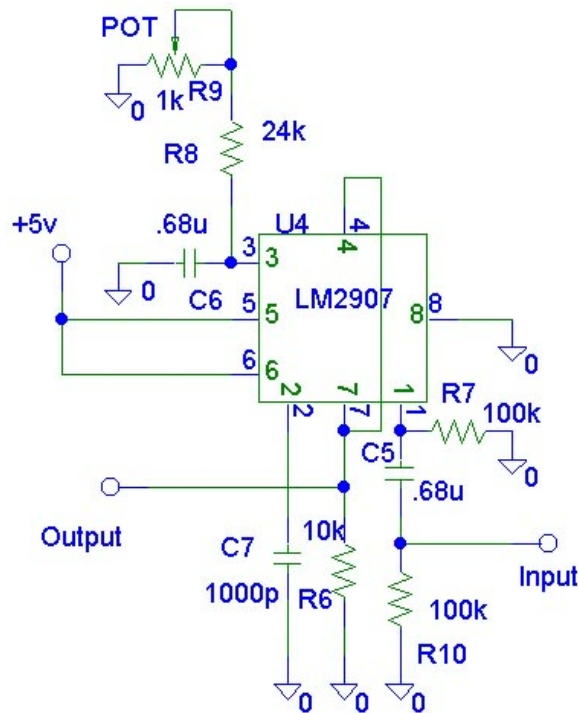


Frequency to Voltage Converters

Since we wanted to utilize closed loop speed control, we had to find a way to collect speed data from the motors and send it to the HC12 for speed correction. We figured that we could convert the frequency of the pulses sent out by the motor encoders into an analog voltage that we could feed into our HC12's A/D ports. Although this method seemed repetitive, (converting a digital signal to analog and then back to digital) we found that this would be the best way to simplify our code. Knowing that the HC12 would need motor speed data very continuously, we took advantage of the A/D ports very high refresh rate and avoided using the cumbersome and pokey input capture code.

Figure 4 Frequency to Voltage Converters





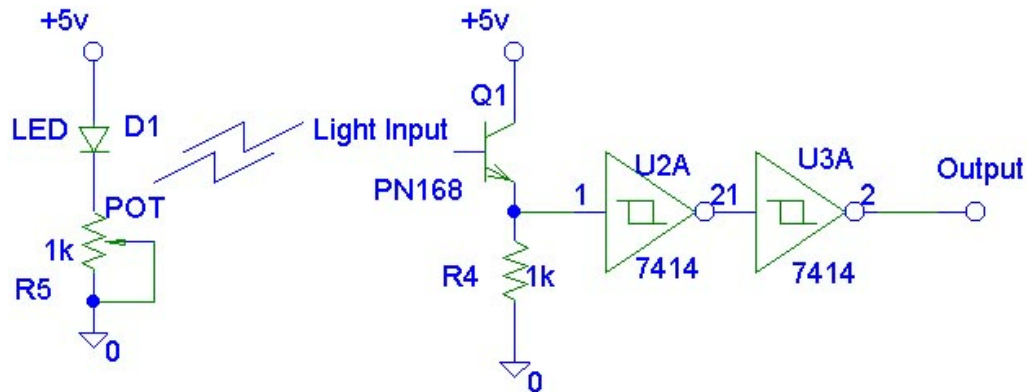
Frequency to Voltage Converter

Figure 5

White Line Sensor

We needed a white line sensor for a couple of reasons. We needed to be able to tell when we entered and exited a room, and also be able to tell when we were within twelve inches of the candle. To carry out this task we built the sensor out of a PN168 phototransistor and an array of green LED's. We mounted the PN168 in a hood to shield it from the direct light coming from the LED's. We also ran the signal coming from the PN168 through a Schmidt trigger to give us a nice zero output if it did not see a white line or a 5 volt output if it did. We then put everything together on a printed circuit board and mounted that inside another hood to shield out any outside light. We took this entire

assembly and mounted it on the underside of the front of the robot so it would catch a white line as it was moving forward.



White Line Sensor

Figure 6

Flame Sensor

Another integral piece of our robot was our flame sensor. This was especially important so we could find the fire that our robot was to extinguish. For this application, we again used the PN168 phototransistors. For this sensor, we used two of them mounted side by side so a binocular vision would help us zero in on the flame. We mounted the two PN168's inside of a plastic hood that we made out of some scrap plastic as seen in the diagram below. This method proved to work pretty well. The reading each PN168 would take from the candle would be about five volts even at a distance over five feet away. The only problem we encountered was the fact that outside light would interfere with the sensor's operation. To counter this, some additional shielding was

required. A piece of black paper to shield out light from the ceiling was more than sufficient for this.

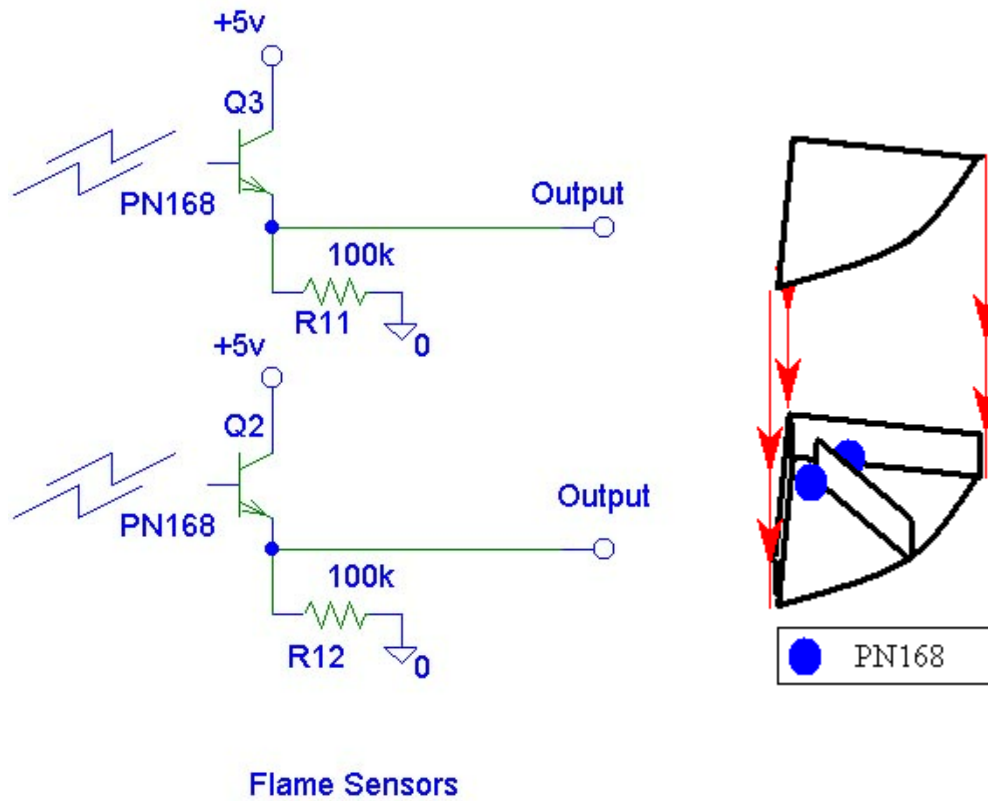
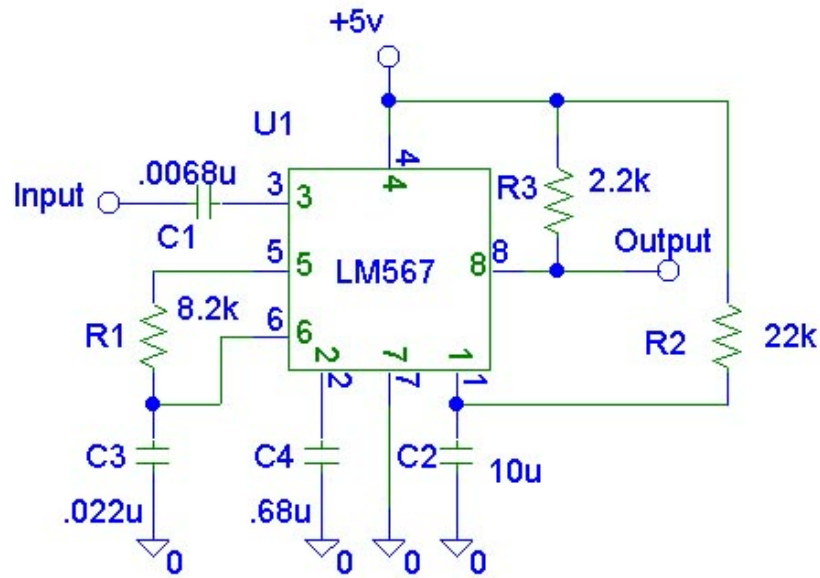


Figure 7

Tone Decoder

For bonus points we decided to employ a tone decoder which would start our robot when it received a 3.5 kHz tone. A hand held buzzer produced the tone we needed and the microphone connected to the tone decoder circuit received it. As long as the tone was of the right frequency, it would produce a digital signal (low) and send it to the HC12, which would start running our program. We found that there was no need to amplify the $200 \text{ mV}_{\text{p-p}}$ signal produced by the microphone because the tone decoder was

designed to trigger with a 100 mV_{p-p} signal. The fact that our tone decoder triggered only when our tone generator was about 1 inch from the microphone proved to be very useful in competition because we did not have to cover our microphone while other groups were running.



Tone Decoder
Figure 8

Flame Suppression

For our flame suppression, we used a one ampere, six volt motor that we pulled out of a radio controlled car. To the motor, we attached a propeller from a radio controlled airplane. Together, the motor and fan produced enough of a wind to extinguish the candle. This system worked so well, we could extinguish the flame from over five feet away.

Figure 9 6V 1Amp Motor



Circuit Board Design

Frequency to Voltage Converters, H-Bridge, and 5V Regulator

Because of our single plate design, we wanted to use very compact circuit boards for our robot's many subsystems. Also, we planned on putting as many subsystems on each board as possible. We crammed the frequency to voltage converters, H-Bridges, and the 5V regulator on a single board while understanding that we would have to be careful since this board combined 5V and 12V components. All components that needed 5V could plug into a main 5V bus.

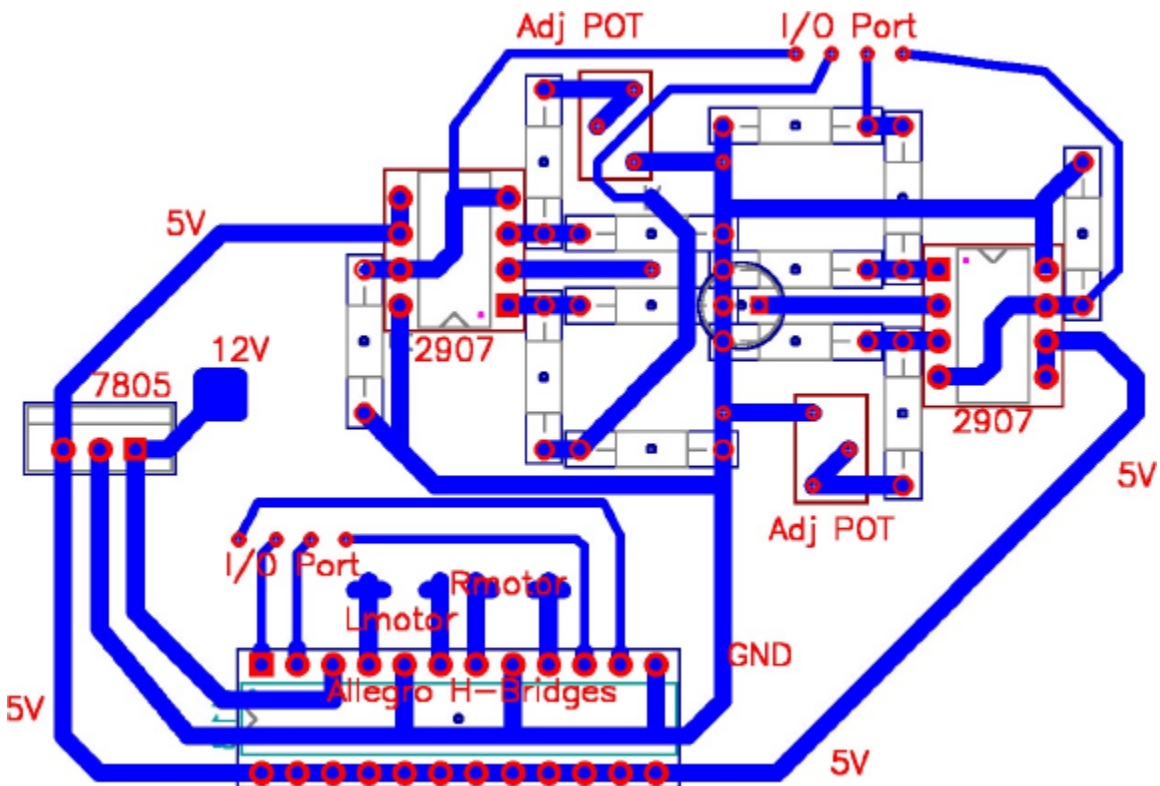


Figure 10 Frequency to Voltage Converters, H-Bridges, and 5V regulator

White Line Sensor

Our white line sensor also had to be compact enough to fit under the robot inside some sort of hood. We were able to fit the whole board inside a film container and mount it to the chassis with a single standoff.

Total board fits inside film container.

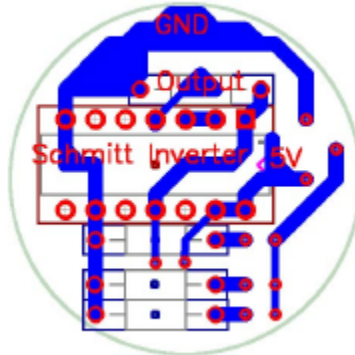


Figure 11 White Line Sensor

Tone Decoder

Our tone decoder circuit board was small enough to mount to the fan motor mount with one standoff. We pointed the microphone to the rear of the robot to allow the robot to drive forward without hitting us.

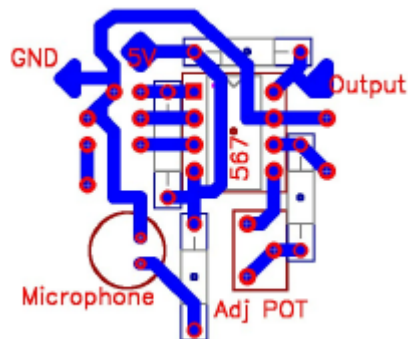


Figure 12 Tone Decoder

HC12 Modifications

An interesting part of the semester was recapturing analog channels 0 and 1.

Step 1. Collect the following:

- Two M68EVB912B32 evaluation boards (one for the master and one as the slave)
- BDM cable with 2X3 header connector (included in your HC12 cardboard box)
- A 11.5V source and a 5V source
- A few single header (f-f) connector wires

Step 2. Configure the boards

- On the Master:
Set the jumpers $W4 = 1$ and $W3 = 0$
- On the Slave (the board to be programmed):
Set the jumpers $W4 = 0$, $W3 = 0$, $W8 = 11.5V$ (i.e. V_{pp}), and $W7 = V_{pp}$
- For the BDM cable:
Make the connections such that the Master has BDM out connected to the BDM in on the Slave.
- Power up the two boards and the 11.5V source (could use a prototype board in Digital Lab).

Step 3. Download the new Debugdlc.s19 file

- Startup a terminal window (appropriately configured for the HC12)

- Type the following as illustrated below (with a return after each line)
R> STOP
S> FBULK
S> FLOAD
- Then download the Debugdlc.s19 file (send text file using Windows 3.11 terminal using the option for waiting after each line ...)

Step 4. Power down and disconnect 11.5V source and reset W7 to VDD

Step 5. Remove the resistors R7 and R8 and place 2-header pins in their place

Step 6. Wire wrap pin at R7 to PDLC0 and pin at R8 to PDLC1

This gave us two extra A/D channels. Because we had one HC12 with the memory expansion, we first tried this on an HC12 that did not have the extra memory. We did this just incase we did something wrong and we did not loose our most important board. Later in the semester we got a lot of experience loading the Debugdlc.s19 file because we had done it on 5 more HC12 boards for other groups in the class. Once we recaptured these channels on our unexpanded HC12, we wrote our wall following functions and our room scans with this board.

We did not find any problems with recapturing these channels, so we felt confidant enough to modify our expanded board. This is where we found some problems. On the original board, we never lost bytee, but our expanded board would lose bytee all the time. We talked to Dr. Rison about this problem and he gave us a different version of

Debugdlc.s19 so we would never use bytee on a startup. This did not work as well as we had planned. When we loaded this the second time we could erase the write protected area, but we were unable to erase the section that was not write protected. We were unable to load the new Debugdlc.s19 file into our HC12. We then reprogrammed our HC12 with the original Debugdlc.s19. We could not find a definite reason why we would lose our bytee. Before we went to Trinity, we would lose it if we turned the power to our robot on and off really fast, or occasionally when we would press the reset button. While we were at Trinity we had to reload bytee every time we would reload our program. After we returned from Trinity we have not reloaded bytee.

Software Design and Testing

Throughout the semester, we spent the majority of our time and energy on the software. For everything that was implemented in code, we would first write a small test program. This was very helpful when we were debugging later in the project. The software was probably the most difficult aspect of the project, but it was by far the most fun. After the competition at Trinity, we had a better understanding of the weaknesses of our robot and completely rewrote all of the code before the local competition.

To navigate the maze our robot would left wall follow the first 3 rooms. After leaving the 3rd room, we would right wall follow until the right wall would disappear. We would then drive straight until we found a front wall. At that point we would then begin to right wall follow into the 4th room. We choose to left wall follow through the maze because when we first got our left and right wall following functions working, we found we could left wall follow through the maze 3 seconds faster than we could right wall follow through the maze at the same speed. Because our white line sensor was activated

by a Schmitt trigger, we used the falling edge of IC1 to determine if we were entering a room. To actually find the white line around the candle we ran the same input wires to ports of our HC12 and disabled our interrupt.

The best part of our code was the wall following functions! Because of the closed loop design of our robot, we had no problems navigating the maze. The first function that we were able to successfully test was the left wall function. In this portion of the code our center sensor was dominant over the left wall sensor. If there was a wall directly in front of it, our robot would spin until a threshold was met. We had two separate

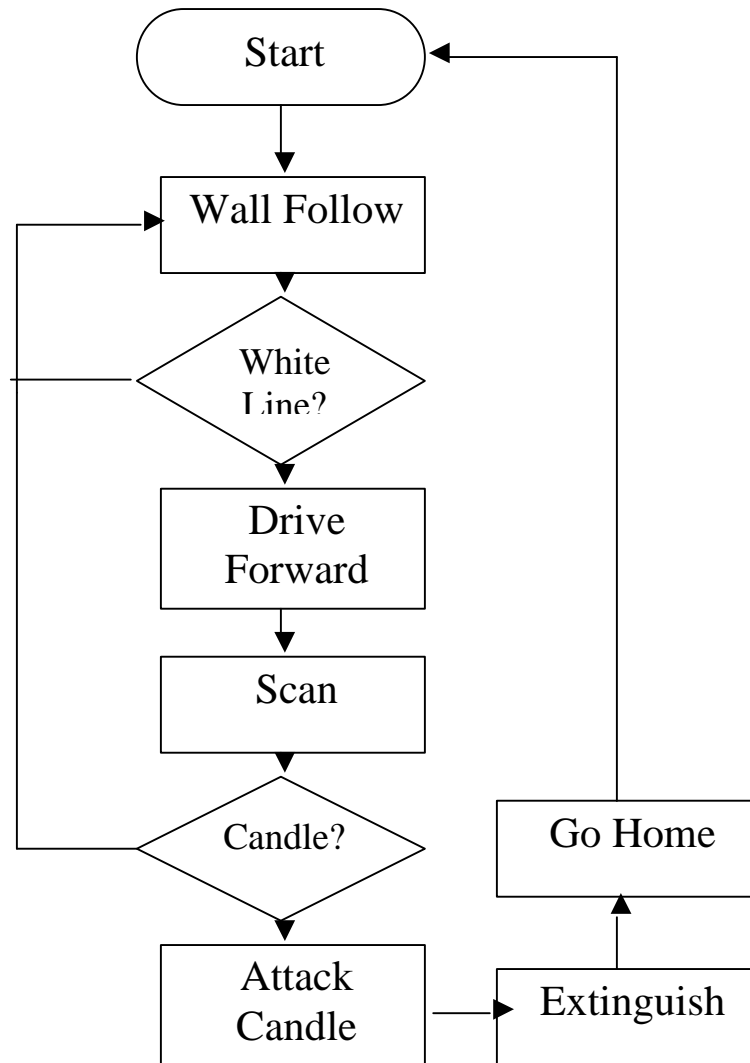


Figure 13. Flowchart for our program

constants for this. The first constant was used to find out if we were too close to the wall. The second constant was smaller, and was continuously compared to our front sensor values to make us turn faster. Basically what we did was make it think it was closer to the front wall than it really was. We would then use our left sensor to judge how far away from the wall we were and then use our proportional closed loop equations to adjust towards or away from the wall. The second function that we were able to successfully test was the right wall function. This function worked exactly like the left wall except we would look at the right wall sensor.

The next function that we were able to create was drive forward. This function, depending on the values that were passed to it, would drive forward a small distance checking to ensure it was the proper distance away from the chosen wall. This function was very useful because we would call it to get off the home circle and when we drove into rooms to check for a candle.

The scan function was used to check if there was a candle in the room. This function went through a lot of modification throughout the semester. When we originally wrote this portion of the code, we would pass it a value to scan different ways. Originally, in the first three rooms, we would scan and turn left about 45 degrees and then turn and scan to the right about 225 degrees. In the fourth room we would turn and scan to the right first, then we would turn and scan to the left. Originally we would immediately stop when both sensors were above a threshold, and call our Attack Flame function. Later we found out that this would cause us some problems. If we were scanning room 1 and the candle was in the first corner in room 2, our robot would find the candle in room 2 while still in room 1, and drive towards it until it found a white line.

It would then turn on the fan and extinguish the candle. The only problem was we were in the doorway of room 1 and we needed to be within 12 inches from the candle before it could be put out. To fix this problem in room 1 we would stop scanning after about 200 degrees while we were turning to the right. While we were at Trinity we had a problem with candle reflections on the walls in the smaller rooms. To solve this problem we changed the scan function to turn toward the left wall and stop. Then it would start turning to the right and scanning, storing the position of the robot when we would find the highest value of our sensors if they were above the fire threshold. If a candle in the room, we would then turn back to the position of the highest value of the sensors and call our Attack Flame function. This slowed our robot down a few seconds, no matter what room it was in.

Our Attack Flame function was really solid. We would drive toward the candle using our fire sensors to direct us and we would check to see if we were too close to a left or right wall. Our robot would run straight for the candle. In certain corners of all rooms if we tried to drive straight toward the candle our robot would hit the wall before it could find the white line surrounding the candle. To fix this problem, we would check how close we were to a wall. If we were too close we would drive along side of it until we found a white line. This added feature of in this function was well used when we were using our original Scan function. When we found a white line we would call our Extinguish function.

The Extinguish function would turn on the fan and check to see if the fire sensors were below a given threshold. If they were not below the threshold, our robot would turn on the fan a little longer. If the candle still was not out, we would turn to the left and

right while the fan was on. We never had a problem putting the candle out on the first try, so we took out the turning part of the program because of safety reasons. We also took out this part of the program because we would blow the candle around the room, then run into it when we were trying to leave the room. This would put wax all over the floor and really disrupt other groups when they were testing in the maze.

Our robot would keep track of which room it was in. When it put out a candle it had a different path home according to which room the candle was in. If the candle was in room 1 we would turn left about 90 degrees and drive until our front or right sensor found a wall. We would then right wall follow until we found the home circle. If we were in room 3 or room 4 we would turn right 90 degrees, drive forward until the front or left sensor found a wall, then left wall follow until we found the home circle. In room 2 we would turn right 90 degrees, drive forward until our center or left sensor found a wall, then we would left wall follow out of the room. When we were in the hall way we would act like a pinball. We would check both the left and right sensors, only turning the robot away from the sidewalls. Basically bouncing down the hallways until we found a wall directly in front of us. When we found this wall we would then start to left wall follow to the home circle.

Some of the bonus features of our robot were never needed. If our robot checked all the rooms for a candle and did not extinguish a candle, it would go to the home circle, stop, turn around and begin to check all the rooms again, lowering the fire thresh hold. This feature of the code was never used outside of testing, or demonstrating our robot to the faculty. It was a really neat idea, but we never had to use this during competition. Another feature of our robot was the ability, after it extinguished a candle and returned to

the home we could then place the robot back into the center of the circle and immediately tone our robot again without having to load the program after each run.

Conclusion

Our robot was a definite success. We started out with the goal of building a robot to win a competition, and that's what we did. Our robot took fourth place at the National Competition and first place at the local competition. We are happy with our results. If we had an extra week to work, we could have cleaned the robot up and make it more presentable. If we had it to do all over again we would design from the get go to be more compact and tidy.