Final Report Group 4: REGG1

Rocky Ginanni

Ernest Jim

Gerald Bivens

Gur Notani

Junior Design, Spring 20001
EE 382

Professors:
Dr. Bruder
Dr. Wedeward

Table of Contents

Page #

Abstract

The REGG1 Fire Fighting Robot works by using a combination of many subsystems. These subsystems work together to allow the robot to navigate and then locate a candle inside a particular maze that contains four rooms. Once the candle is located, the robot must then extinguish the candle and return to the original starting point. The maze that is built especially for this robot is constructed with walls painted a reflective white and a floor coated in a non-reflective flat black paint.

This report will cover five subsystems of the overall design of this Junior Design Project; navigation, fire sensing, white line detection, closed-loop motor control, fire suppression, and overall design of the robot. The design evolvement for these subsystems will be discussed as well.

Keywords: Navigation, Wall Following, White Line, Fire Control, Closed loop Motor control, Budget, Program, Figures

Introduction

The overall purpose of this junior design project was to design and build an autonomously functional robot. REGG1, as our robot was named (an acronym for the names of our group members), navigates a mockup of the floor of a house, finds and extinguishes a candle, and returns to the home-base circle. By dividing the various tasks up with some regard to our individual skills, our group was able to design and build all the different parts. For weeks the building of REGG1 was finished with only final touches to do.

Being C-challenged, our program seemed to have many problems that would have all of the working sub-routines functioning together. We finally incorporated code for REGG1 to wall-follow and navigate through the maze without the use of dead reckoning, or just measuring of how far the robot has traveled. By utilizing voltage vs. distance measurement from the sensors for wall following, REGG1 can maneuver throughout the maze. Another subsystem was white line detection. The purpose of the white line sensor determines if the robot has entered a room. It is also an indication of the proximity of the robot to a candle and the home or starting circle. Finally the fire sensing sub system enabled REGG1 (with just a single sensor) to implement fire detection and location through scanning while our fan put out the flame. After completing this task, REGG1 was instructed to find its way back to the home circle.

The closed loop speed requirement was filled with the use of our programmable Altera chip. It was able to read counts from the motor encoders for use by the HC12 microprocessor. We could then utilize this information in the main program so that the robot would maintain constant velocity and not slow down when going up a ramp or speed up when going down it.

Function to Form

The design of REGG1 started as top down approach, but as time went by and as we incorporated more and more of the final components, the design evolved by itself. Our timeline in the appendix highlights the major design changes that occurred throughout this semester (page 43).

Since the motors available were limited our, motor selection seemed to take precedence in our preliminary design. The motors' characteristics were conveniently placed in a table with links to their specifications and data sheets located on the instructors' class web pages. We were to come up with reasons for our selection (quickly) otherwise the most desirable motors would be used by other groups. For this we selected Maxon 6-Watt 22mm motors (green body) as they had a 14:1 gearhead ratio which would provide enough torque for our purposes. The motors also had 500-count quadrature encoder that would provide meticulous readings for our closed loop speed control. Because these motors were rated for 18volts the 12volts we supplied would not be a problem, but they would run a little slower than optimum.

Mounting brackets were made from a scrap leftover three-inch aluminum angle bracket obtained from the machine shop. Holes for the wheel shaft and mounting screws (2.6mm) were to be drilled precisely. The specifications design sheet on the class web page link has dimensions for these and a template (see figure 1.) made by printing the drawing of them, then the holes can be drilled exactly (134158gear.pdf). Care was taken to ensure that the encoder end of the motors was positioned to provide ground clearance. The encoder end of the motors was secured with plastic tie straps to keep the motors straight and level in the event that the weight of the robot would "pigeon-toe" the wheels.

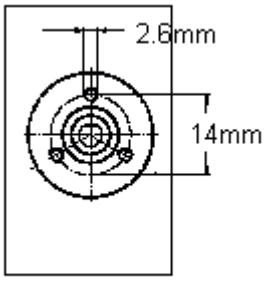

Figure 1. Motor mounting Bracket Template

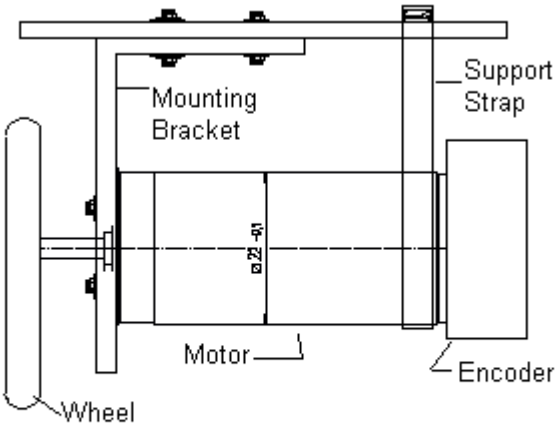

Figure 2. Motor mounting with support strap

The sensor mounting brackets were made similarly but from two-inch aluminum angle bracket around 3/4 in wide. These brackets were not as precisely made but the positioning of them was. The assembly plate was marked with 90 degree lines for the object-in-front sensor and 45 degree lines for the side wall sensors. The holes drilled for mounting the brackets were made so the angles could be adjusted if the sensor values were optimum (see figure 3).

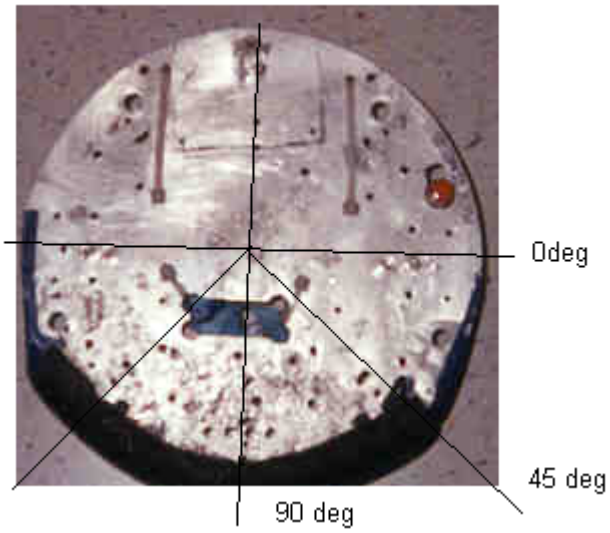

Figure 3. Optimum Placement of Wall Sensors

After power components were tested we came up with a block diagrams (see appendix page 39) and wiring diagrams (see appendix page 37) to start building the power distribution board. Since the high power (for motors) were kept separate from low power bringing everything into a single board was challenging. It helped to have an overall block diagram to place and solder the components on the poke-through board. Then the wiring diagrams could be utilized. The boxed headers for the enabling the motor encoders were made for a printed circuit board and in order to expedite their use it was decided to pull out the short pins an reinstall longer poke-through pins obtained from left over single pin headers. This worked well and alleviated the need (and expense) for poke through dip-sockets or block extensions. Another necessity that became apparent was the use of polarized connectors for the incoming battery power and outgoing power to the HC12 and motors (including fan). The mounting of this central distribution board was vertical rather than horizontal for ease of sensor connections and overall testing. The final board included everything in one central location (figure 4).

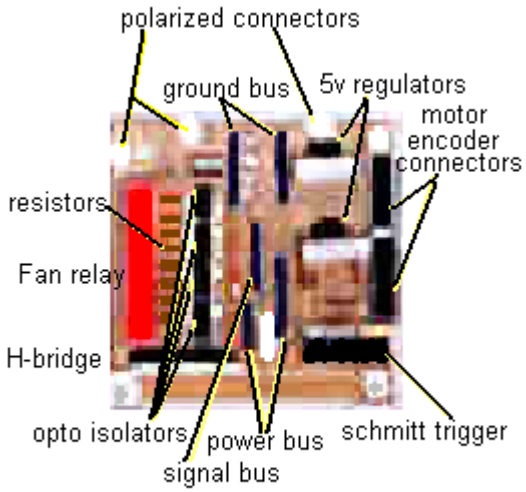

Figure 4. Power and Signal Distribution Board

Originally, our robot was designed to have a top assembly plate for our fire sensors and fan mount. The plate kept getting the way of testing and we kept it off for some time. Then when we got the wall following going we needed the fire sensor so it was just stuck up in the air by itself on a piece of 1/4 in all-thread. This proved to be a great streamlining idea and we incorporated it along with the fan motor. The idea was just to keep the fan blade high enough to keep from chopping wires. In the last week we decided to move the Pyro-electric sensor so that it would be somewhat adjustable to candle height.

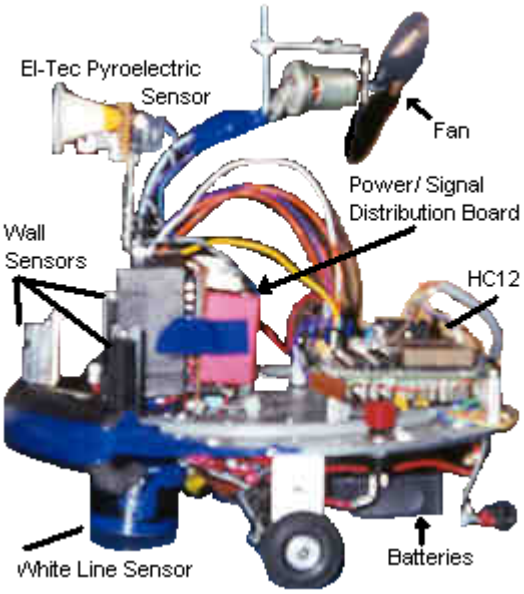

Figure 5. REGG1 Parts Placement

Single wire connections were replaced with homemade shielded cables. This was done with multi-colored ribbon cables. After replacement, the robot acted strangely and it was decided that due to cross talk (or insufficient shield grounding) of the sensor signals, those wires would have to be single connections. With a little more time, we could have made a cable assembly that would shield each individual signal wire, and then REGG1 would have been even more streamlined.

Description of the Power and Signal Distribution Board

Power distribution and regulation

We have used two batteries, 3 voltage regulators, 2 fuses, and a single switch in our design to regulate and distribute power to separate high/low power devices (refering to figure 9 page 39 in appendix). We have defined the power going to H-bridge, motors and fan as high power and to HC12, Altera chip, sensors, and motor encoders as low power. The two batteries are operated through a single switch to be able to turn on or off simultaneously. The high power devices are protected by 3-amp fast blow fuse and a 2-amp slow blow fuse protects the low power devices.

For the high power we have used 12 volts lead acid battery. This connects directly to the H-bridge and the fan and supplies VCC to optoisolators after step down conversion to 5 volts. The optoisolation protects the low power devices from any damage and interference by high power. The fan is controlled by opto-relay by getting the signals from HC12. This battery uses insulated wire for a high power ground. Opto-isolators and the fan get their high power ground connection from the middle pin of the voltage converter connected to the 12 V battery.

On the low power side we have a battery pack consisting of 8 AA size cells nickel metal hydride rechargeable batteries, selected for their high current density of 1500 mAH. Power from this battery pack after step down conversion to 5 volts goes to low power devices. The heat sensor El-tec chip gets further step down to 2.5 volts. The other lead of the battery is connected to the aluminum platform for a low power ground The low power circuits, optoisolators, and opto relay get their ground connection from the middle pin of low power side voltage converter. This opto-isolation has worked very well for us in reducing cross-interference between the operation of low power and high power devices. We have used LED lights to emulate the fan during the test runs to conserve energy.

An LM7805 voltage regulator is used to produce clean 5-volt supply. It does not need any external circuitry It has 90 dB ripple rejection ratio which is able to make the regulator to produce a good clean 5 volts It is capable of delivering 500 mA current which is sufficient for the circuitry it has to supply the current to.

Motor Control Power

There were several h bridges to choose from. We used the Allegro 2998 H Bridge We chose this H Bridge to power our robot because it needed no external circuitry It was also a dual h bridge which meant we did not need individual h bridges for each motor and furthermore it was cheap It is rated for 3 A peak current and 2 amps continuous current which was sufficient for the duty cycles of the PWM that we were to use to run the Maxon motor.

HC12 generates 2 pulse width modulation signals, one for the left motor and one for the right motor. The H Bridge takes pulse width modulations and direction voltage digitally from HC12. The H Bridge processes the duty cycle of the PWM from HC12 to determine how fast the motors are turning. The direction line determine in which direction the motors would turn If the direction line is showing ground input the H Bridge will send forward moving signal and if the direction line is showing 5 volt input the H

bridge will produce backward moving signal. Output pins 4 & 6 and 7 & 9 pulse after it is enabled; pulse modulates the two motors.

Wiring Board Circuitry and Signals

We have a single wiring board that houses 5 separate circuits, namely motor control circuit, fan control circuit, white line detection circuit, heat detection circuit, and motor encoder circuits. The schematic of the wiring board is in appendix #. The Altera chip is housed on HC12 EVB.

Pulse width modulation and motor directions are controlled by HC12 through a cable on port T and port P1.

We have used Maxon motors to drive our robot's locomotion. Coming from the motor encoder we have two signals in quadrature to determine the distance and the speed. We have two 16-bit counters and have one channel for divide down to keep manageable timer overflow rates.

As the motors turn that information is fed into the encoders and relayed to the HC12 A/D converter to determine the speed of the motor as well as to determine the position of the motor in relation to the reflecting surface. There is a line that connects the encoders to the Altera chip on the HC12 as a jumper so that we can have a feed back control. The output of white line sensor is passed through Schmitt trigger and transmitted to HC12 A/D converter. Connections for all the analog signals are interfaced with HC12 through a single bus connected to HC12 by a cable ribbon numbered 1-6.

Power Budget

Power consumption was estimated at the robot's initial design and we found that there was sufficient energy in the batteries to power the robot for extended periods of testing. Since the power distribution was broken into two parts there was less of a demand on any one battery.

	12 V System	9.5 V System
Item	Watts	Watts
Motors	12 W	
H bridge	6 W	
Fan	9 W	
HC12		0.43 W
Encoder		0.29 W
Fire Sensor		0.1 W
White Line Sensor		0.15 W
Distance Sensor		0.45 W
Regulator	2.5 W	5 W
Total	29.5 W	6.42 W

From experience the 12 volts system would last an hour of frequent testing and

9.6 volt battery would last 1 hour. Both of these batteries have a fast charge characteristic

System Control Requirements

Description	Requirement	Goal
Search Reliability	90%	100%
Navigation	95%	100%
Navigation technique	80%	100%
Motion Control	80%	100%
Max Speed	40%	100%
Max Torque	30” inc	30”inc
Avoid Obstacle	80%	100%
Maximum Range Before Fire Suppression	24”	24”
Return Home After Fire Suppression	90%	100%
Power Source	100%	100%

Sensor Subsystems

I. Distance Sensor Subsystem

Hardware

To enable the robot to navigate through the maze, this requires the ability to determine the distance to the nearest wall or object. This is accomplished by using a sensor that can determine the distance from the robot to that object. After reviewing all available sensors that can do this, the Sharp GP2D120 was the sensor of choice.

1. 4-30cm range
2. Fairly resistant to ambient lighting
3. Contains all necessary logic to determine distance
4. Requires that the signal output be run through an Analog to Digital Converter

Initially, five sensors were chosen to be able to see all around the robot. At a later time, it was determined that only three sensors would be needed to accomplish navigation. These three sensors would be placed towards the front of the robot. One directly front, and two more to either side of the front one at an angle of 45 degrees. Which would allow the robot to see ahead of itself, for determining the next turn and navigating faster.

The GP2D120's are a very nice package. They require only 5V at 50mA. Since these sensors have a range of only 4-30cm, the range between 0-4cm can give erroneous readings. To solve this, they can be moved inwards by 4cm from the edge. Though we discovered that we were able to accomplish navigation quite smoothly by having the sensors positioned at the edge of the robot, rather than inset by 4cm.

The output of these sensors can be fed directly into the HC12's A/D port that would convert the analog output to a digital one.

Since we initially chose to use five sensors, the sensors that finally used were numbered 2-4. Which helped keep track of the values that were used in the software programming.

From the manufacturer, these sensors would give out a logarithmic scale of distance vs. voltage. So for each sensor, we mapped this distance vs. voltage curve to determine the accuracy of each sensor. Then from this mapping, we could determine the best placement for them, and whether or not if a sensor gave out a too wide a deviation by comparison to the manufacturer’s curves. We did happen to discover that a couple of sensors were unstable in their output, so these were not used. All of this allowed us to increase the robots ability to navigate smoothly. Which it did very well. This is displayed in figures10-12 in appendix. As can be seen, the data obtained matches each other very closely. From these graphs, we also determined what the threshold values were to be used.

With the diameter of our robot at 25.2cm, we would be too small to stay in the middle of the hallways, so that meant that we had to stay a minimum distance away from the walls. Also, since our two side distance sensors would be placed towards the front, pointing towards the wall at 45 degrees, this correlates to a long distance in relation to the range of these sensors. We determined that we wanted to maintain a minimum of 5cm away from the wall, which translated into a total sensor range of ~10cm.

Software

Since the outputs of these sensors are analog distance signals, this required that we send this signal to the HC12's A/D Converter before we look at the final signal. Once this was done, we could now determine the correct value for minimum distance by using the equation:

$$ADR\#H = (Voltage * 255)/5$$

where *ADR#H* corresponds to a particular port number of the A/D converter, PORTT (Sensor2=*ADR6H*, Sensor3=*ADR5H*, Sensor4=*ADR4H*), the *Voltage* is the output voltage of the distance sensors, 255 is an 8-bit number value limit, and 5 corresponds to Vcc supply. Now that we knew that actual measurement for 5cm, we could set a constant global value for a particular sensor in the program. Then maintain a constant 5cm away from the left, front, and right walls.

Once we finished setting up the A/D converter, we were finally able to implement both left and right wall following into our program with little or no effort. These threshold values were also applied to the closed loop speed control algorithm described in the Closed-Loop Speed Control section.

Knowing that we would need to constantly see these values, we decided to implement an interrupt routine to see the A/D conversion every 2ms. Which allowed us to maintain a fairly constant wall following distance without any over compensation due to time delays.

Ultimately, we looked at when the robot was too close, to far, opening to the left/right detected, and wall in front of robot.

II. Fire Sensor Subsystem

Hardware

We did research to determine which type of sensor would best detect a flame from a long distance. Out of these sensors: IR Sensor, Ultraviolet Sensor, and CCD Camera, the UV Sensor would be the best for doing an initial scan of the rooms, then use a IR Sensor to locate the flame in the room. But due to budget constraints, the UV Sensor was not feasible. So we decided to be different a see if we could use just one sensor to do both tasks.

It was decided to use the El-Tec Pyroelectric Thermal Sensor to do this. This sensor has the ability to see thermal changes from as far as 5meters. It also sees only the infrared energy given off by a black body heat source such as flames or humans. Which corresponds to a narrow wavelength of 200-300nm. It also gives an analog output that necessitates running the signal through the HC12's A/D converter. This output changes depending on the direction of motion. That is, if scanning from the left to right (a high/low transition), right to left (low/high transition). So this particular sensor is directional.

By design, this sensor has all the necessary internal logic to give out an analog output when it senses a thermal change, that corresponds to a change in IR emission from the hot body source, on its face detector. It starts off at a 2.5V offset voltage from 0. Then when it senses anything, the output signal changes from this offset voltage ± 2V. The only problem in this transition is that it occurs rather quickly, ≤ 1ms. Which is not long enough for the HC12 to see this change. So, after many attempts at trying to see the output at the right time, we discovered that by putting the signal through an Integrator circuit, it would hold the output long enough so that the HC12 could see the change. This took the better part of the whole semester to figure out.

From the manufacture, this sensor has a 180° field of view. But we only need to look straight ahead when looking for the flame. So, supplied with the sensor were instructions to construct a cone with a Fresnel lens at the end. This Fresnel lens would limit the field of view to a narrow 3-inch beam of detection facing forward. So, to be able to see a varying height candle flame, we made the sensor-mounting bracket adjustable in the up or down plane. The nice thing about this sensor is that it is not affected by ambient light from the overhead florescent lighting, so no light filtering was required.

Software

The output of this sensor is analog, which required that we send the signal through the A/D converter of the HC12. But, as commented on before, the output had to be sent through an Integrating circuit before it could go to the A/D converter. Once this was done, we could establish a threshold value to look at all times. Which could have been ≥ 200 or ≤ 20, after the A/D conversion. We chose to go with the ≤ 20 value that corresponds to our initial direction of scanning in the rooms.

To look at the final converted values of this sensor, we used the interrupt routine to see it every 2ms on the A/D line *ADR3H*. Even though we established a threshold value, we constantly kept change our value to look at, because there were times that we would pick up random fluctuations in the sensor output. So, much experimentation was done to find a suitable value.

A single scan routine was used for scanning the room and then if a flame were detected, we would jump into a separate function to take care of it. The end result was that we were able to implement one sensor to take care of multiple functions at the same time. Which made for a more efficient design.

III. Whiteline Detection Subsystem

Hardware

Since at the home circle, entrance to every room, and around the candle base there is a whiteline or white solid disc, one of the most important functions that we needed to do was to see these lines or discs. As per the rules set forth at the beginning of the semester, the robot has to start on the home circle, be over the whitelines at the rooms, and candle base. So after looking at various devices at our disposal, we decided on implementing a combination IR emitter/detector pair housed in a plastic covering from the manufacture. The only draw back to this sensor was that there was no information available to determine maximum current values it could sustain before burning up. Which we discovered these max values after burning a couple of them.

Since this IR pair is not immune to ambient light, a covering was made to encase the IR pair to block out any unnecessary reflections. This IR pair was placed ~1/2 off of the floor. We didn't have any problems with detecting the reflection off of debris off the floor, which was very good.

The next addition needed was that the robot had to be in a certain distance over the whitelines, so the placement of the sensor was agreed to be towards the center of the robot. Thus allowing the added benefit of moving the robot, ~3", closer to the flame.

The design construction of this IR pair is a phototransistor and IR LED, which were housed in a piece of black non-reflective plastic. Both angled from each other at an angle of ~45 degrees.

This IR pair did require external logic to implement a functional operation on the robot. The figure (in appendix page 37) shows the circuit layout of the IR pair. The addition of the Schmitt Trigger was so that when the whiteline is detected, the output would go to 0V. Then when the black surface of the floor is seen, then the output would go to 5V. Now we could run this output straight into an input port located on the HC12 without any sort of conversion needed.

Software

The output of the Schmitt Trigger was fed into PT5 of PORTT. A RTIF, every 2ms, was used to detect when the whiteline was seen.

We ran into a problem when seeing the whiteline. Falsely seeing:

1. Home circle.
2. Whiteline when leaving a room.

Both cases had to be delt with in the coding.

In the first case, this was done by doing a delay start at the very beginning of the program, in which the robot drives forward a certain amount without looking at any

sensors, including the whiteline sensor. In the second case, when it sees the whiteline upon exiting the room, it would then advance forward a certain amount, then continue on wall following. In both cases, our solution these problems proved quite effective 100% of the time. The only draw back is that the delay start will be ineffective if the ramp is directly in front of the robot.

IV. Fire Suppression Subsystem

Hardware

Once the room has been scanned, and a candle has been found in that room, the robot must now proceed to extinguish the flame. To do this, we researched various devices that could extinguish a flame. A CO2 canister, fan, water spray, and concussion. Of these, the fan route would be the easiest to implement. The others were either too bulky, complicated for our use, or too dangerous (illegal in the rules).

Our suppression system would consist of:

1. DC Reversible motor
2. 1-Fan blade
3. 1-High power relay

The DC motor we scavenged out of the parts bin. With no literature on this motor, we had to determine the maximum current limit it could operate at. Since we didn't know, we concluded that we would only run the fan motor for about 3 seconds at full 12V power. Any longer than this could seriously damage the motor windings. And at the full 12V power, the fan could be able to extinguish the flame from a fairly far distance as determined from experimentation, but we had to be within 8" of the candle, so we were okay.

Software

Since the HC12 could not supply the 12V directly, we ran a 12V-supply line into a relay and an output line from PORTT into the relay as well. So when we asserted the output line from the HC12, it would cause the relay to make a direct short from the 12V battery to the fan. Thus turning on the fan from a high power source.

Now once the candle has been detected in the room, the robot will go into a separate function to locate the candle in that room by stopping and moving towards the candle. Until it sees the whiteline, which tells it that the robot is within a certain distance of the flame. At this point, the robot will go into another function that turns on the fan for 3 seconds. Since the air output of the fan is pretty high, we didn't feel that any further scanning would be needed at this point. So we then just turn around and exit the room.

The fan blade was a simple hobbyist’s propeller. Though, the more blades, and greater pitch available, the more air the fan could have moved. But we were happy with what we had. This setup worked flawlessly 100% of the time.

Altera Programming

Closed Loop Motor Control

The closed-loop motor control is handled by taking motor speed-readings inside the real-time interrupt in the HC12. Inputs to the motor control are two global integers that specify the rate of desired travel and the rate of desired turning. The detection of the actual speeds of the right and left wheels by reading off of counters that run off of the right and left wheel encoders respectively, and taking the difference between the present value and the previous values of each. The speed of each wheel is controlled by the PWM fed from the HC12. The PWM to the left motor is controlled by the sum of the desired speed minus the actual left wheel speed (as measured by the counters) plus the turning error. In order to correct for turning error the speed control algorithm need feedback from the speed of the left motor) and speed of the right motor) with a given input bias. In turning, between the right speed and left speed there is difference value plus a bias value. The right wheel speed is similar except the turning error is subtracted. As an example here are these lines of code (for going straight) from the complete program (see appendix):

```
lpos_speed = LSPED;          //left motor speed
rpos_speed = RSPED;          //right motor speed
lduty = (DS*KL)/100 + ((DS - lpos_speed)*KWL)/10;
rduty = (DS*KR)/100 + ((DS - rpos_speed)*KWR)/10;
PWDTY0 = ((lduty*PERIOD)/100) - 1;
PWDTY1 = ((rduty*PERIOD)/100) - 1;
```

Where DS is desired speed and the K values are constants determined by experimentation.

Altera programming was used to calculate the speed and position of our robot. This was done by taking signals form the encoders of each motor (see figure 6). Since the Altera chip we were using could not hold very much memory, we divided each Channel-A signal off each encoder by sixty-four. This enabled use to us four 8-bit counters on the Altera chip. And still have enough memory to use the program for expanded memory. Two counters were used for speed and two for position sensing. The complete Altera .gdf (graphical design file) is located in the appendix.

For position sensing the encoder signals were divided by sixty-four, then the divided signals were counted. The counter had the ability to count up and down, so one would know if the robot were moving forward or backward. The numbers are counted, then held in an address, and when the HC12 checks that address, it is given the position. For speed the encoder signals were also divided by sixty-four, then the divided signals were counted. The output of the counter was then connected to an 8-bit latch. A PWM signal from the HC12 was used to latch the count value to an address location every five milli-seconds. After the latch was complete, the counter would be reset to zero. When the HC12 wanted to know its speed, it would check the address that the speed was stored at.

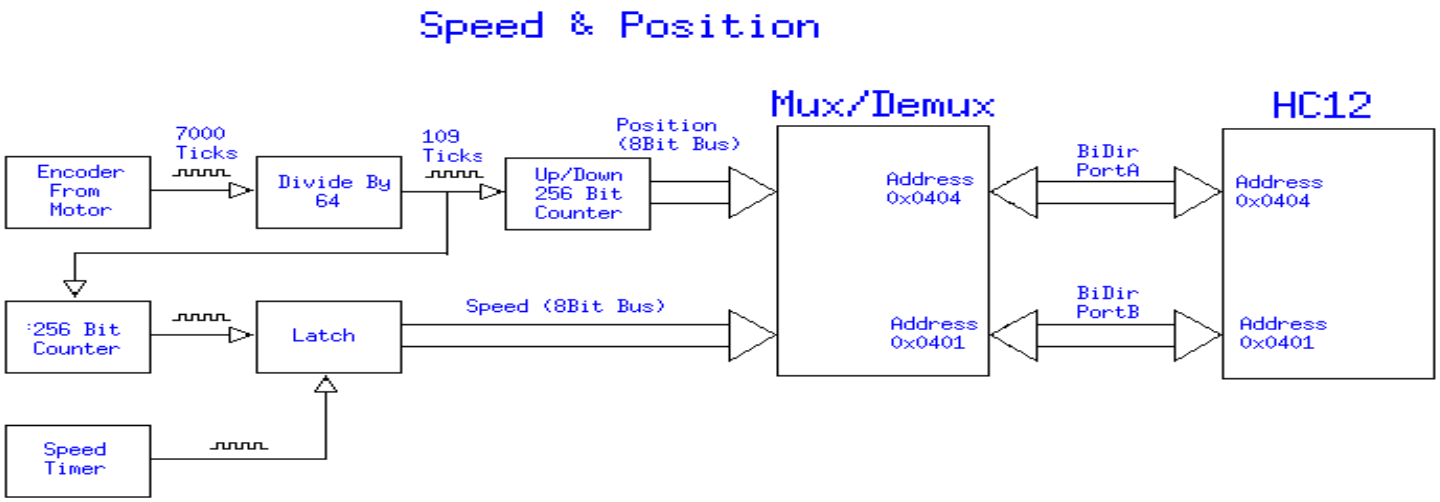

Figure 6. Altera Block Diagram

C Program Description

This is the brief description of the state diagram of the system (see figure 7).

The state diagram is our guide in formulating the code of our system once the code is downloaded and run into the robot, the registers and interrupts are set up. After completing the initial set up, the pr**o**gram enters a start algorithm that has a delay to have time to put the robot in the maze. The next state is push-button subroutine whe**r**e it wil**l** remain until the button is pushed at which point it will **j**ump out of the loop and into main routine. In the main routine robot gets its instruction to activate the motors and b**r**ing them up to a desired speed.

Once the motors are started, the robot will left wall follow. For the left wall following algorithm, the speed measurements and distance measurements are utilized for error corrections to keep the robot follow the left wall and moving with a constant speed.
If robot sees opening to the left it will make a left turn after which it will continue the left wall following looking for a white line. It moves into the scan state after it detects the white line

In the scan state, robot moves past the white line for a given distance, pauses, pivots for 270 degrees of a revolution while getting the measurements from the heat sensor.

If the heat sensor value indicates that there is fire in the room, it moves into homing on fire state. It positions itself directly lining up with the candle and moves towards the candle.

It stops after detecting the white line that surrounds the candle to move into the fire suppression state.

If the fire is not detected in the 1^{st} room, it exits from the room by left wall following until it detects white line in the 2^{nd} room.

The same scanning routine is performed in rooms 2 and 3.

After it comes out of the 3^{rd} room, it continues left wall follow a predetermined distance, after which it is allowed to go straight to cross the hallway to detect the front wall. At which point it does right wall follow to detect the white line in room 4 to perform the scan subroutine.

After each of the room maneuvers the robot will return to the home starting circle position.

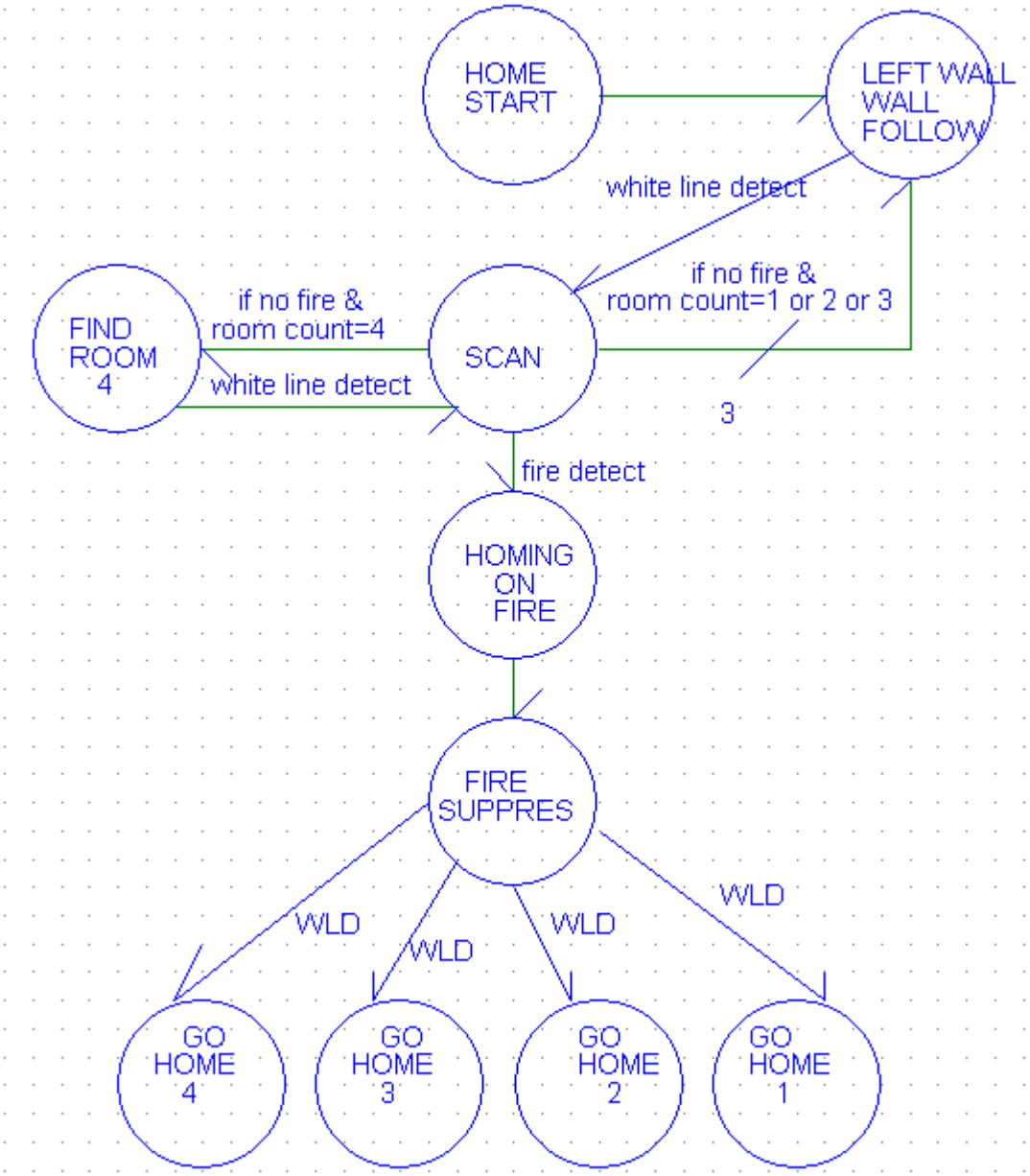

Figure 7. Program State Diagram

Budget

Here is a list of all components used in the construction of REGG1, their approximate cost and how they fit into our allotted budget.

H-Bridge (Allegro)UDN2998W	$7.00	1	$7.00		
Schmitt trigger optoisolators	$1.50	9	$13.50	fried 5	
5V regulators: 7805	$0.75	3	$2.25		
3/16” shrink tubing (price per inch)	$0.10	5	$0.50		
IR distance sensors:connectors:	$0.80	5	$4.00		
crimping wire connectors-motors	$0.10	6	$0.60		
GMA fuses(1,2,4,6A):	$0.50	1	$0.50		
Wire-wrap DIP socket 2x10-pin:	$0.75	1	$0.75		
Wire-wrap DIP socket2x8-pin:	$0.50	3	$1.50		
Boxed headers2X8-pin (male)	$1.50	2	$3.00		
crimp contacts: (i.e. pins for above)	$0.10	50	$5.00		
Antex solid-state opto-isolated2.5A DC relays:	$5.00	1	$5.00		
Reflective IR emitter/detector pair:	$1.25	1	$1.25		
Misc. capacitors:	$0.10	2	$0.20	5 analog/ digital lab	
2.6mm screws		6	$0.00	?	
Battery Holder: 8-AA holder	$2.00	1	$2.00		
Schmitt trigger inverter	$0.75	1	$0.75		
Misc. spacers/standoffs 6-32 spacers:	$0.10	2	$0.20	Rest salvaged	
Inserts	$0.20	2	$0.40		
all-thread rod	$1.00	1	$1.00		
9V battery Charger	$9.00	1	$9.00		
Ribbon Cable	$0.50	4	$2.00		
Caster wheel assembly	$10.00	2	$20.00	Broke 1	
TOTAL			$80.40		
Maxon 6-Watt 22mm motors (green body):	$50.00	2	$100.00		
12V 2.0Ahr lead battery:	$30.00	1	$30.00		
Rechargeable NiMH batteries: Rayovac	$2.00	0	$0.00	donated	
Battery Holder: 8-AA holder	$2.00	0	$0.00	donated	
IR distance sensors:0-30cm: GP2D120:	$8.00	4	$32.00		
El-tec Pyroelectric sensor	$49.95	1	$49.95		
Hamamatsu sensor	$57.72	0	$0.00	NOT USED	
TOTAL			$211.95		
					saved
GRAND TOTAL			$292.35		$7.65

Free and Salvaged Stuff

Fan Blade	$0.50	1	$0.50	donated
9V battery connectors	$0.60	5	$3.00	donated
polarized Lock pin connectors	$0.60	5	$3.00	donated
Polarized housing:2-pin housing:	$0.00	2	$0.00	salvaged
10” dam. 1/16” (thick) Al. disks:	$7.00	1	$7.00	salvaged
crimping wire connectors-power supply	$0.12	18	$2.16	donated
12V lead-acid charger:	$9.00	1	$9.00	donated
Fuse holder :	$1.00	2	$2.00	salvaged
10-pin male header:	$0.20	5	$1.00	salvaged
Misc. resistors		15	$0.00	analog/digital labs
Misc. resistors		2	$0.00	stock room
Misc. capacitors:	$0.10	2	$0.20	5 analog/digital lab
Misc. switches:	$0.50	3	$1.50	salvaged
PC wire-wrapping boards	$3.00	3	$9.00	salvaged
1.5” rubber wheels:	$2.50	2	$5.00	salvaged
Misc. wiring from Norton's shop and leftovers	$0.00	2	$0.00	free/salvaged
Misc. connectors	$0.00	2	$0.00	free/salvaged
Misc. screws nuts and bolts	$5.00	1	$5.00	free/salvaged
Misc. brackets	$5.00	1	$5.00	scrap/salvaged
TOTAL Not included in Budget			$53.36	

Conclusion

The process involved in constructing REGG1 from start to finish seemed complicated. The integration of all the hardware with the software was most distressing. By dividing the various tasks, with teamwork and cooperation, our group was able to design and build an autonomously functional robot. We overcame many problems with C-code instability, budget constraints, signal conditioning, and hardware manipulation. There was a lot of inter-team cooperation as well, which helped considerably. But, having completed the overall purpose of this junior design project we are quite pleased with having put together a robot that was unique in its function.

REFERENCES

Books

Mobile Robotics-Inspiration to Implementation,

Joseph L. Jones–Anita M. Flynn–Bruce A. Seiger. A. K. Peters Publishing.

Natick, Massachusetts. Second Edition, 1999

Web Sites

"www.ee.nmt.edu/~bruder"

Professor: Dr. Stephen Bruder

"www.ee.nmt.edu/~wedeward"

Professor: Dr. Kevin Wedeward

El –Tec Pyroelectric sensor
www.acroname.com.

Sharp GP2D120 distance sensors
smaecom1.sharpsec.com/

Maxim Schmitt-Trigger
www.maxim-ic.com.

List of figures:

Appendix

C program

```
#include "hc12.h"
#include "dbug12.h"

#define enable() _asm("cli")
#define TRUE 1
#define PERIOD 175   //Set period for PWM
#define DS 52        //Desired speed
#define KL 146       //Left motor conversion constant
#define KR 136       //right motor conversion contstant
#define KW 9         //Wall follow constant
#define KWL 10       //Wall follow constant, left motor
#define KWR 10       //Wall follow constant, right motor
#define DSL 15       //Left motor speed for left turn
#define DSR 25       //Right motor speed for right turn
#define KPL 150            //Left turn conversion constant, left motor
#define KPR 150            //Left turn conversion constant, right motor

#define SENSOR1 52   //Desired distance for sensor 1, not used
#define SENSOR2 54   //Desired distance for sensor 2, left wall
#define SENSOR3 70   //Desired distance for sensor 3, front wall
#define SENSOR4 60   //Desired distance for sensor 4, right wall
#define SENSOR5 112  //Desired value for sensor 5, heat sensor

void delay(unsigned int num);    //Function delay for a time of num*1ms
void fan();          //Function to turn on fan when fire is found
void speed();        //Function to look at the speed of the individual motors
void right_turn();   //Function to do a right turn basd on the front&rightfront sensor
void stop();         //Function to stop the robot
void scan();         //Function to scan the rooms, then jump to the appropriate function
void room4manuver(); //Function to manipulate the island room
void left_turn();    //Function to do a left turn based on the fron&leftfront sensor
void rightWallFollow();  //Function to do right wall following
void leftWallFollow();  //Function to do left wall following
void room1exit();    //Function to return home after candle out in room 1
void room2exit();    //Function to return home after candle out in room 2
void room3exit();    //Function to return home after candle out in room 3
void room4exit();    //Function to return home after candle out in room 4
void roomexit();     //Function to to determine which room to exit from

void leftturn(unsigned int num); //Input num to do a leftturn based on wheel encoder ticks
void rightturn(unsigned int num); //Input num to do a rightturn based on wheel encoder
ticks
void advance(unsigned int num); //Input a num to move straight based on wheel encoder
ticks
```

```
//Setup the appropriate variables that will be needed for the program
volatile unsigned char s1=0,s2=0,s3=0,s4=0,s5=0,s6=0, lposition, rposition,
lpos_speed,num2;
volatile unsigned char rpos_speed, SDL, SDR, whiteline,
room_number,lmotorspeed,rmotorspeed;
volatile signed int linecount, K1=0, K2=0, fire=0,fire_out=0,startcircle=0;
volatile unsigned int lduty, rduty, wheel_turn,;


main()
{
//Enable bit 0,1 output: others input
 DDRT = 0x23;
 DDRT = DDRT & ~0x04;
 room_number =0;

//Setup for PWM on Channels 0 & 1 & 2
 PWCLK = 0x00;              //8-bit mode
 PWCTL = 0x00;              //Left-aligned
 PWPOL = 0x0F;              //Clock mode=0, pos. polarity, all channels
 PWSCAL1 = 9;
 PWCLK = PWCLK | 0x1F;      //N=3 for Channels 0 & 1 & 2
 PWPER0 = PERIOD - 1;
 PWPER1 = PERIOD - 1;
 PWPER3 = 149;
 PWEN = PWEN | 0x0F;        //Enable PWM on Channels 0 & 1
 PWDTY0 = 255;              //Left wheel
 PWDTY1 = 255;              //Right wheel
 PWDTY3 = 20;               //PWM for Altera clock time

//Setup for A/D conversion on all channels
 ATDCTL2 = ATDCTL2 | 0x80;   //Power up A/D
 ATDCTL4 = 0X01;             //2MHZ E-clock, 9us conversion
 ATDCTL5 = 0X70;             //8-bit convert, continuous convert, multichannel

//Setup for real time interrupt
 RTICTL = 0x82;             //Set interrupt rate at 2ms
 RTIFLG = 0x80;             //Clear source of interrupt

 enable();                  //Enable interrupts

 delay(10000);              //Set a delay time before start
 advance(800);              //Move forward num amount to avoid homecircle

 while(TRUE)
```

```
 {

  leftWallFollow();         //Function call for left wall following

  if(whiteline == 0 && room_number <= 4) //if we see a white line....
   {
    stop();   //Stop robot
    scan();   //Jump to scanning room
    room_number++;   //Assigns room numbers for subroutines
   }
  else if(room_number == 3)  //If we exit out of room three...
   {
    room4manuver();  //Jump to the the room 4 function
   }
  else if( room_number >= 4)  //If no flame found in room 4, just exit
   {
     fire_out= 1;
     room4exit();
   }
 }
}

//To do left wall following
void leftWallFollow()
{
 if(s3 >= SENSOR3)    //Object in front
  {
    stop();          //Stop robot
    right_turn();    //Turn right if object in front
  }
 else if(s2 == SENSOR2)  //If moving straight, adjust speed of wheels
  {
    speed();
    lduty = (DS*KL)/100 + ((DS - lpos_speed)*KWL)/10;
    rduty = (DS*KR)/100 + ((DS - rpos_speed)*KWR)/10;
    PWDTY0 = ((lduty*PERIOD)/100) - 1;  //Left wheel duty cycle
    PWDTY1 = ((rduty*PERIOD)/100) - 1;  //Right wheel duty cycle
  }
 else if(s2 <= 20)    //Opening to the left detected
  {
    lduty = 15;
    rduty = 145;
    PWDTY0 = ((lduty*PERIOD)/100) - 1;  //Left motor duty cycle
    PWDTY1 = ((rduty*PERIOD)/100) - 1;  //Right motor duty cycle
  }
 else if(s2 > SENSOR2)    //Too close to left wall
```

```
   {
    lduty = (DS*KL)/100 - ((SENSOR2 - s2)*KW)/10;
    rduty = (DS*KR)/100 + ((SENSOR2 - s2)*KWR)/10;
    PWDTY0 = ((lduty*PERIOD)/100) - 1;   //Left motor duty ccycle
    PWDTY1 = ((rduty*PERIOD)/100) - 1;   //Right motor duty cycle
   }
  else if(s2 < SENSOR2)    //Too far from left wall
   {
    lduty = (DS*KL)/100 - ((SENSOR2 - s2)*KWL)/10;
    rduty = (DS*KR)/100 + ((SENSOR2 - s2)*KW)/10;
    PWDTY0 = ((lduty*PERIOD)/100) - 1;   //Left motor duty cycle
    PWDTY1 = ((rduty*PERIOD)/100) - 1;   //Right motor duty cycle
   }
}

//Do a right turn based on sensor values
void right_turn()
{
   PORTT= 0x01;       //Change right wheel direction
   while(s3 >= SENSOR3 | s2 >= SENSOR2)  //Looks at sensor values
    {
     lduty = (DS*KL)/100;
     rduty = (DS*KR)/100;
     PWDTY0 = ((lduty*PERIOD)/100) - 1;
     PWDTY1 = ((rduty*PERIOD)/100) - 1;
    }
   PORTT = 0x00;
}

//Scans the room for the flame and outs it
void scan()
{
  int line = 0;
  int value = 0;
  int turnticks = 0;
  int t_tick = 0;
  int holdcount= 0;
  int turn = 0;

  advance(750);  //To drive past the whiteline
  stop();        //stop the robot
  leftturn(414); //Turn left 90 degrees to setup scan

  while (turnticks < 1600)   //Do scanning based on motor encoder ticks
    {
     s5 = ADR3H;
```

```
    PORTT = 0x01;
    PWDTY0 = 0x27;
    PWDTY1 = 0x30;
    s5 = ADR3H;
    if(LMPOS != holdcount)
      {
       holdcount = LMPOS;
       turnticks = t_tick++;
      }
    s5 = ADR3H;

    if(s5 < 30)          //Or if we see a candle....
      {
       stop();
       stop();      //stop the robot
       PORTT = 0X00;        //drive forward until...
       PWDTY0 = 90;         //left motor
       PWDTY1 = 100;         //right motor

       while(line == 0)      //we see a white line
         {
          value = 1;

          if(whiteline == 0)
              {
           line = 1;
           stop();  //Stop Robot
           stop();
           fan();   //Turn fan on
           fire_out =1;
           roomexit();  //Exit the room if we have seen & put out flame
          }
         }
       }
     s5 = ADR3H;
    }
  PORTT = 0x00;

  if(value == 0)//exit out of room if no flame found, continue left wall following
   {
     advance(750);    //exit room (drive forward past whiteline) and stop
     stop();
   }
}

//If flame found & put out, determine which room exit to go home
```

```
void roomexit()
{
  while(fire_out == 1)      //since the fire was put out
    {
   switch(room_number)       //??what room number is it??
    {
      case(0):              //room 1 because the scan was not completed
      {
       room1exit();           //calling go home routine
      }
      case(1):              //room 2 because the scan was not completed
      {
       room2exit();           //calling go home routine
      }
      case(2):              //room 3 because the scan was not completed
      {
       room3exit();           //calling go home routine
      }
      case(3):              //room 4 because the scan was not completed
      {
       room4exit();           //calling go home routine
      }
    }
   }
}

//Does left turn, num, based on encoder ticks
void leftturn(unsigned int num)
{
  int turnticks = 0;
  int t_tick = 0;
  int holdcount= 0;

  while (turnticks < num)
    {
          PORTT = 0x02;
          PWDTY0 = 0x50;
          PWDTY1 = 0x45;
     if(LMPOS != holdcount)
      {
        holdcount = LMPOS;
        turnticks = t_tick++;
      }
    }
}
```

```
//Turns fan on if the scan function found flame
void fan()
{
 PORTT = 0x20;
 PWDTY0 = 0;
 PWDTY1 = 0;
 delay(3000);
 PORTT = 0x00;
}

//After room three, does the island room, 4
void room4manuver()
{
 int timer_ticks = 0;
  int state = 1;

 TSCR |= 0x80; //enable timer
 TMSK2 = 0X05; //262.144ms overflow intervals

 while(room_number == 3) //while were doing room 4...
  {
   if(state == 1)  //left wall follows after exiting room 3
     {
      leftWallFollow();
     }
   else if(state == 2)  //ignores the opening to the left
     {
      speed();
      lduty = (DS*KL)/100 + ((DS - lpos_speed)*KWL)/10;
      rduty = (DS*KR)/100 + ((DS - rpos_speed)*KWR)/10;
      PWDTY0 = ((lduty*PERIOD)/100) - 1;
      PWDTY1 = ((rduty*PERIOD)/100) - 1;
     }
   else if(state == 3)  //when the front wall is found, turn left
     {
      stop();
      delay(100);
      left_turn();
     }
   else if(state == 4)  //right wall follow until whiteline at doorway is found
     {
      rightWallFollow();
     }
   else if(state == 5)  //when whiteline found, do room scan
     {
      stop();
```

```
      scan();
     }
   else if(state == 6)  //if candle found or not, assign room the number 4 to exit
     {
      room_number = 4;
     }

   //Chnages the state based on particular situation
   if(state == 1 && timer_ticks >= 7 && s2 <= 20) state++;
   else if(state == 2 && s3 >= SENSOR3) state++;
   else if(state == 3) state++;
   else if(state == 4 && whiteline == 0) state++;
   else if(state == 5) state++;
   else if(state == 6) state++;

   if((TFLG2 & 0x80)!= 0) //if timer overflow
     {
      timer_ticks++; //increment timer_ticks
      TFLG2 = 0x80; //clears overflow
     }
  }
}

//To do right wall following
void rightWallFollow()
{
   if(s3 >= SENSOR3)    //Object in front
    {
     stop();
     left_turn();
    }
   else if(s4 == SENSOR4)  //Moving straight
    {
     speed();
     lduty = (DS*KL)/100 + ((DS - lpos_speed)*KWL)/10;
     rduty = (DS*KR)/100 + ((DS - rpos_speed)*KWR)/10;
     PWDTY0 = ((lduty*PERIOD)/100) - 1;
     PWDTY1 = ((rduty*PERIOD)/100) - 1;
    }
   else if(s4 < 20)    //Opening to the right detected
    {
     lduty = 145;
     rduty = 17;
     PWDTY0 = ((lduty*PERIOD)/100) - 1;   // left motor duty cycle
     PWDTY1 = ((rduty*PERIOD)/100) - 1;   // right motor duty cycle
    }
```

```
  else if(s4 > SENSOR4)    //Too close to right wall
   {
    lduty = (DS*KL)/100 + ((SENSOR4 - s4)*KW)/10;
    rduty = (DS*KR)/100 - ((SENSOR4 - s4)*KWR)/10;
    PWDTY0 = ((lduty*PERIOD)/100) - 1;   // left motor duty cycle
    PWDTY1 = ((rduty*PERIOD)/100) - 1;   // right motor duty cycle
   }
  else if(s4 < SENSOR4)    //Too far from right wall
   {
    lduty = (DS*KL)/100 + ((SENSOR4 - s4)*KWL)/10;
    rduty = (DS*KR)/100 - ((SENSOR4 - s4)*KW)/10;
    PWDTY0 = ((lduty*PERIOD)/100) - 1;   // left motor duty cycle
    PWDTY1 = ((rduty*PERIOD)/100) - 1;   // right motor duty cycle
   }
}

//To do left turn based on sensor values
void left_turn()
{
  PORTT= 0x02;
  while(s3 >= SENSOR3 | s4 >= SENSOR4)
   {
    lduty = (DS*KL)/100;
    rduty = (DS*KR)/100;
    PWDTY0 = ((lduty*PERIOD)/100) - 1;
    PWDTY1 = ((rduty*PERIOD)/100) - 1;
   }
  PORTT = 0x00;
}

//To do right turn based on encoder ticks
void rightturn(unsigned int num)
{
 int turnticks = 0;
 int t_tick = 0;
 int holdcount= 0;

 while (turnticks < num)
   {
        PORTT = 0x01;
        PWDTY0 = 0x50;
        PWDTY1 = 0x45;
    if(LMPOS != holdcount)
     {
      holdcount = LMPOS;
      turnticks = t_tick++;
```

```
      }
    }
  stop();
}

//Function to exit out of room 1 if candle found & put out
void room1exit()
{
  int white_count = 0;
  PORTT = 0X00;
  leftturn(457);  //turn away from candle circle num amount

  while(s3 < 85)  //move straight until front wall seen...
    {
     PORTT = 0x00;
     PWDTY0 = 85;
     PWDTY1 = 85;
    }

  while(fire_out == 1)  //right wall follow home while..
    {
     rightWallFollow();
     if(whiteline == 0)  //if whiteline at doorway found, move forward num amount
      {
       advance(50);
       room_number == 0;
       white_count = 1;
      }
    while(white_count == 1)  //right wall follow while home while...
      {
       rightWallFollow();
       if(whiteline == 0)  //if home circle found...do a dance
        {
         delay(50);
         stop();
         rightturn(914);
         stop();
         delay(10000);
         room_number == 0;
        }
      }
    }
}

//Function to exit out of room 2 if candle found & put out
void room2exit()
```

```
{
 int timer_ticks = 0;
 int state = 1;

 PORTT = 0X00;
 leftturn(500);  //turn away from candle circle num amount
 advance(500);  //move straight num amount
 PORTT = 0x00;

 TSCR |= 0x80; //enable timer
 TMSK2 = 0X05; //262.144ms overflow intervals


 while(fire_out == 1)  //to go home...
  {
   if(state == 1)  //right wall follow in room until..
    {
     rightWallFollow();
    }
   else if(state == 2)  //once whiteline found, move forward until front wall seen
    {
     PORTT = 0x00;
     PWDTY0 = 85;
     PWDTY1 = 85;
    }
   else if(state == 3)  //once front wall found, leftturn, move forward ignoring room 1
    {
     stop();
     delay(100);
     leftturn(400);
     advance(900);
     timer_ticks = 0;
    }
   else if(state == 4)  //once past room 1 opening, start leftwall follow while...
    {
     leftWallFollow();
    }
   else if(state == 5)  //start right wall follow home
    {
     rightWallFollow();
    }
   else if(state == 6)  //do a little dance at home
    {
     delay(50);
     stop();
     rightturn(914);
```

```
      stop();
      delay(10000);
      room_number == 0;
     }

  //Changes states based on loaction
  if(state == 1 && whiteline == 0 && s2 < 20 ) state++;
  else if(state == 2 && s3 >= SENSOR3) state++;
  else if(state == 3) state++;
  else if(state == 4 && timer_ticks > 5) state++;
  else if(state == 5 && whiteline == 0 && s4 <= 95) state++;

  if((TFLG2 & 0x80)!= 0) //if timer overflow
   {
   timer_ticks++;
   TFLG2 = 0x80; //clears overflow
   }
  }
}

//Function to exit out of room 3 if candle found & put out
void room3exit()
{
 int white_count = 0;
 rightturn(914);
 advance(500);

 while(fire_out == 1)  //left wall follow home while...
  {
   leftWallFollow();

   if(whiteline == 0 && s2 < 20)  //if doorway line found...
    {
     advance(500);
     room_number == 0;
     white_count = 1;
    }
   else if(whiteline == 0 && s4 < 95 && white_count == 1)  //if home circle found...
    {
     delay(200);
     stop();
     PORTT = 0x02;
     PWDTY0 = 85;
     PWDTY1 = 85;
     delay(10000);
     room_number == 0;
```

```
    }
  }
}

//Function to exit out of room 4 if candle found & put out
void room4exit()
{
 rightturn(914);

 while(fire_out == 1)  //left wall follow home while...
  {
   leftWallFollow();

   if(whiteline == 0 && s2 < 20)  //if doorway line found...
    {
     while(s3 < 95)  //move forward while looking at front sensor
      {
        PORTT = 0x00;
        PWDTY0 = 85;
        PWDTY1 = 85;
      }
     room_number == 0;
    }

   if(whiteline == 0 && s3 < 95)  //if home circle found...
    {
      delay(200);
      stop();
      PORTT = 0x02;
      PWDTY0 = 85;
      PWDTY1 = 85;
      delay(10000);
      room_number == 0;
    }
  }
}

//Function to implement a delay of num*1ms
void delay(unsigned int num)    //Creates a delay of (num*1ms)
{
 unsigned int i;

 while(num > 0)
 {
  i = 1000;
  while(i > 0)
```

```
   {
    i = i - 1;
   }
  num = num - 1;
 }
}

//Function to move forward based on encoder ticks
void advance(unsigned int num)
{
  int turnticks = 0;
  int t_tick = 0;
  int holdcount= 0;

  while(turnticks < num)
     {
      PORTT = 0x00;
      PWDTY0 = 0x67;//left motor
      PWDTY1 = 0x60;//right motor
      if(LMPOS != holdcount)
       {
         holdcount = LMPOS;
         turnticks = t_tick++;
       }
     }

  stop();       //stop the robot
}

//Function to stop robot
void stop()
{
 PORTT = 0x03;
 PWDTY0 = 30;
 PWDTY1 = 30;
 delay(50);
 PORTT = 0x00;
}

//Function to look at speed from expanded memory
void speed()
{
  lpos_speed = LSPED;          //left motor speed
  rpos_speed = RSPED;          //right motor speed
}
```

```
//Interrupt function to look at whiteline, distance sensors,
//motor speeds, heat sensor
@interrupt void rti_isr(void)   //Interrupt to tell when whiteline seen
{
 whiteline = PORTT & 0x04; //Looks for white line
 s2 = ADR6H;              //Looks at the left wall sensor
 s3 = ADR5H;              //Looks at front sensor
 s4 = ADR4H;              //Looks at the right wall sensor
 s5 = ADR3H;              //Looks at the heat sensor
 lpos_speed = LSPED;       //Looks at the left motor speed
 rpos_speed = RSPED;       //Looks at the right motor speed
 RTIFLG = 0x80;
}
```

Wiring Diagrams

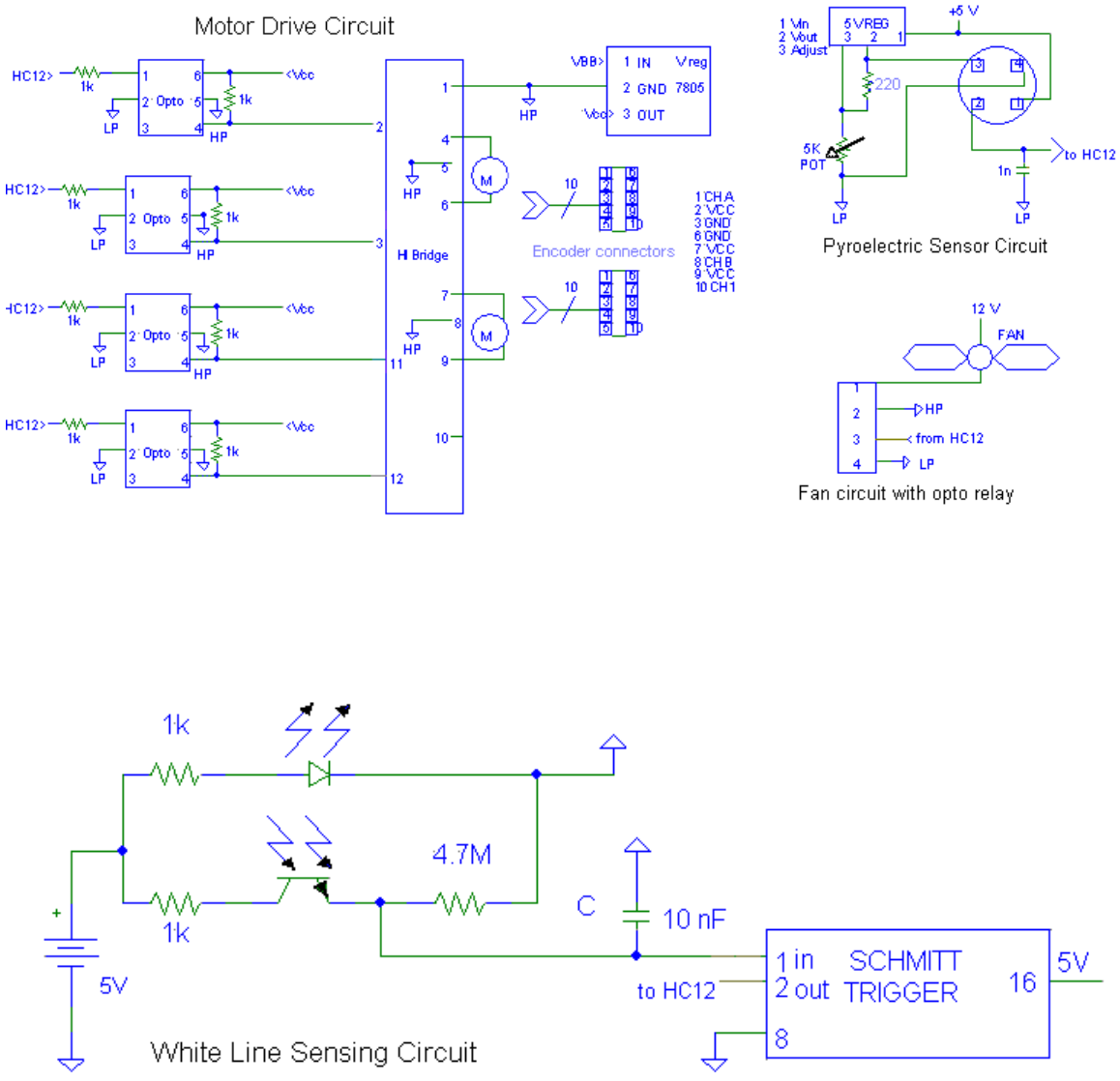

Figure 7. Subsystem Wiring Diagrams

Altera

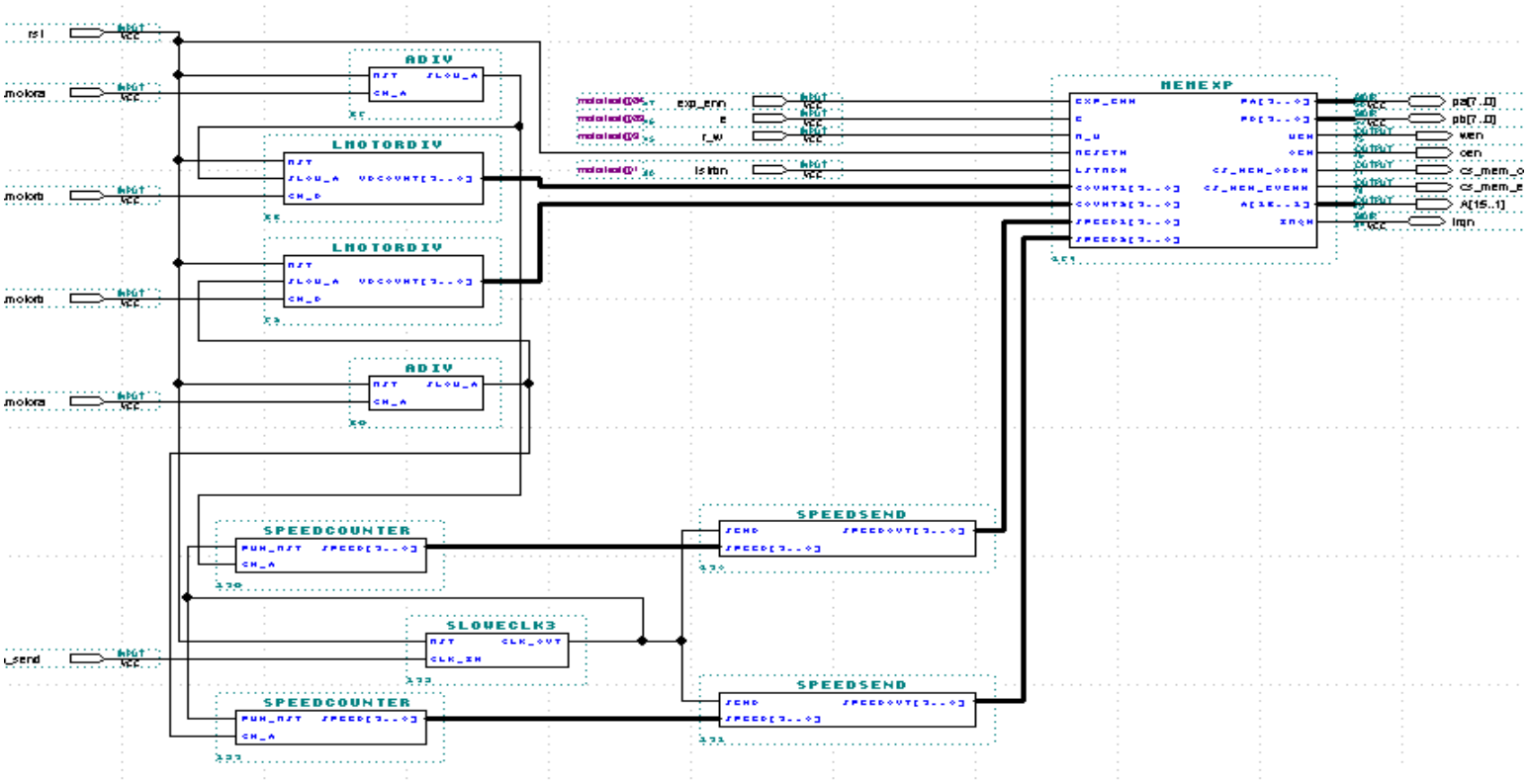

Figure 8. Altera Graphic Design File (.gdf) for counting ticks from the 500 count motor encoders.

Block Diagram

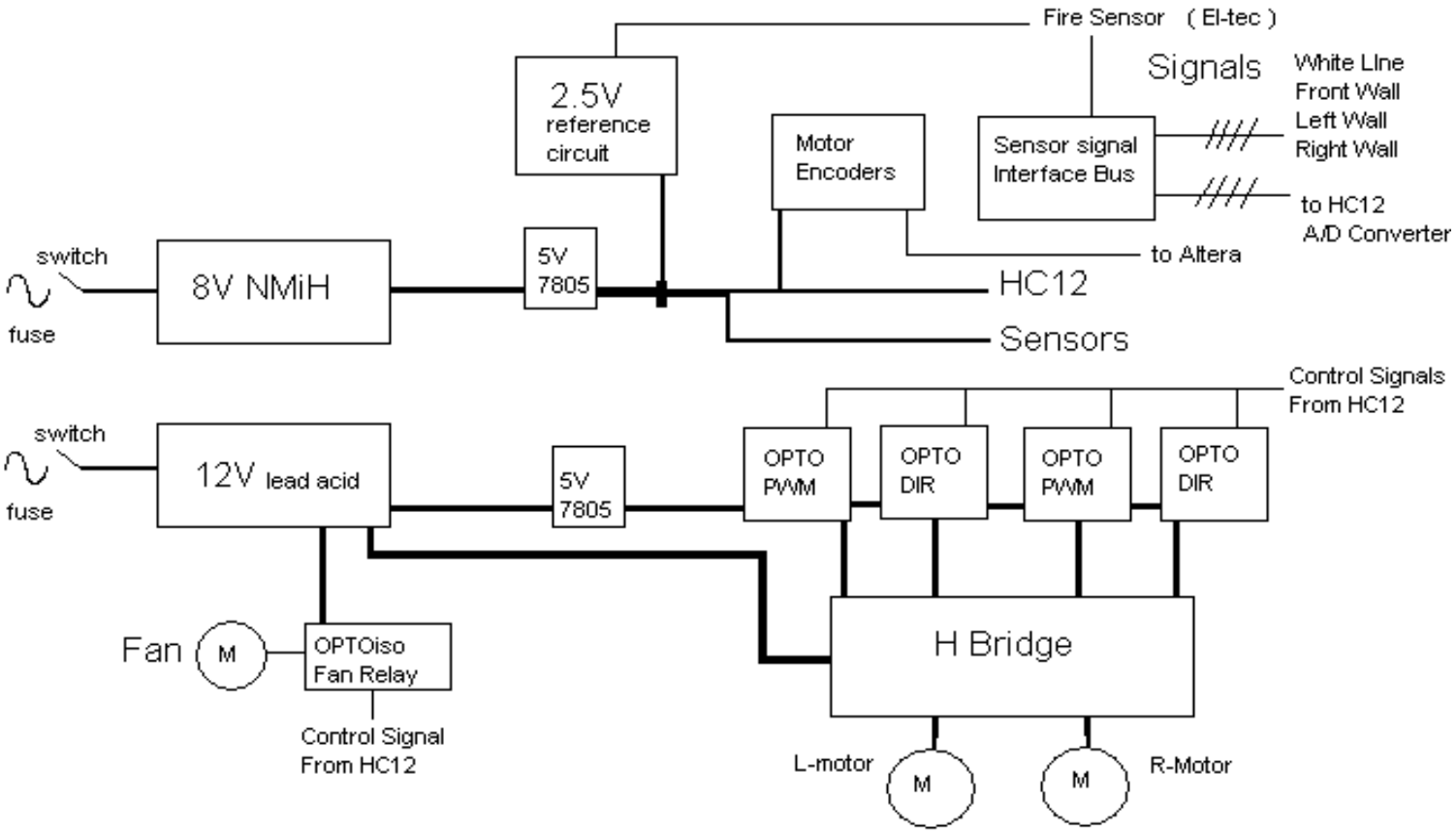

Figure 9.Block Diagram of Power and Signal distribution Board

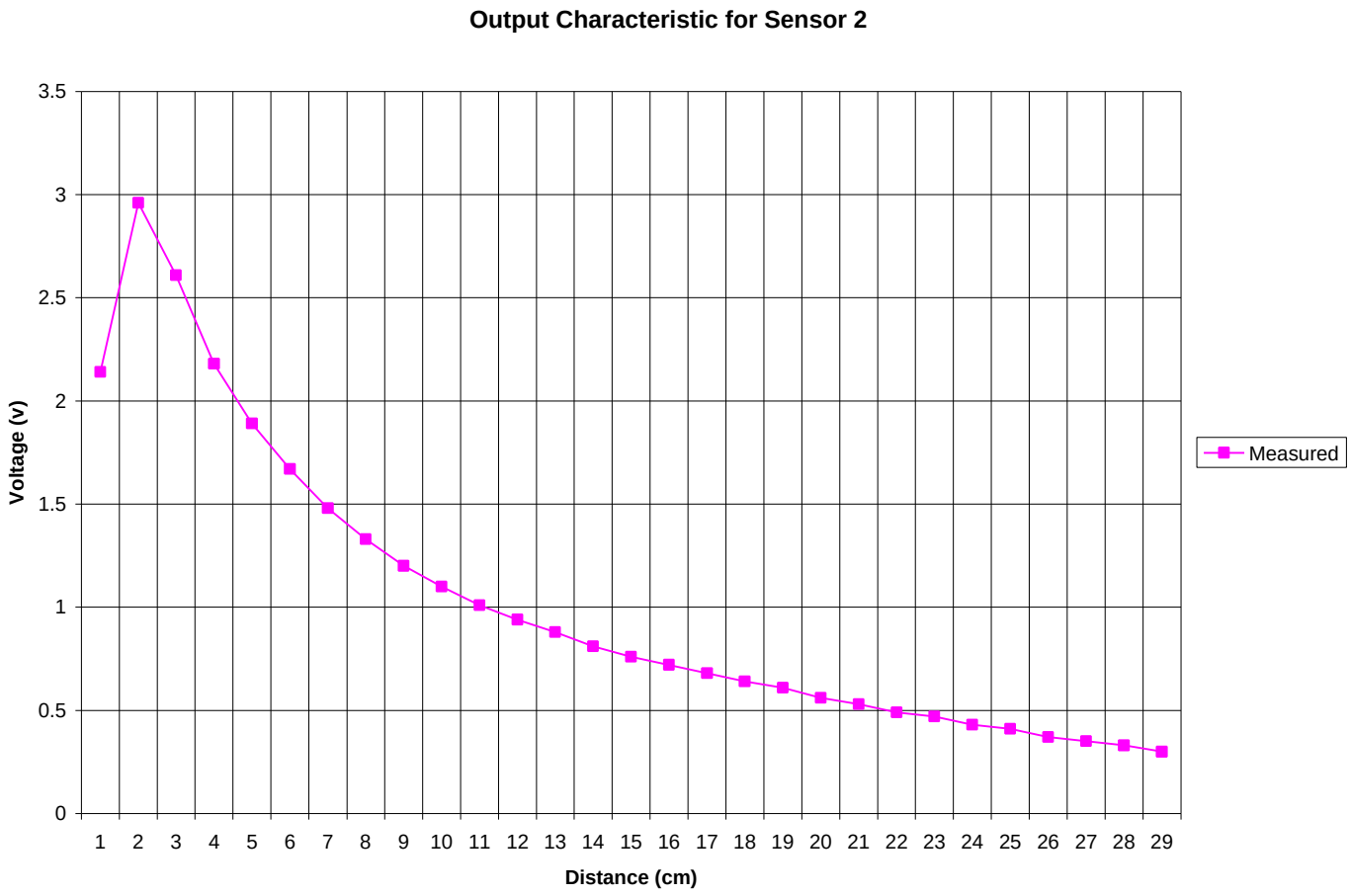

Figure 10. Voltage to Distance Curves

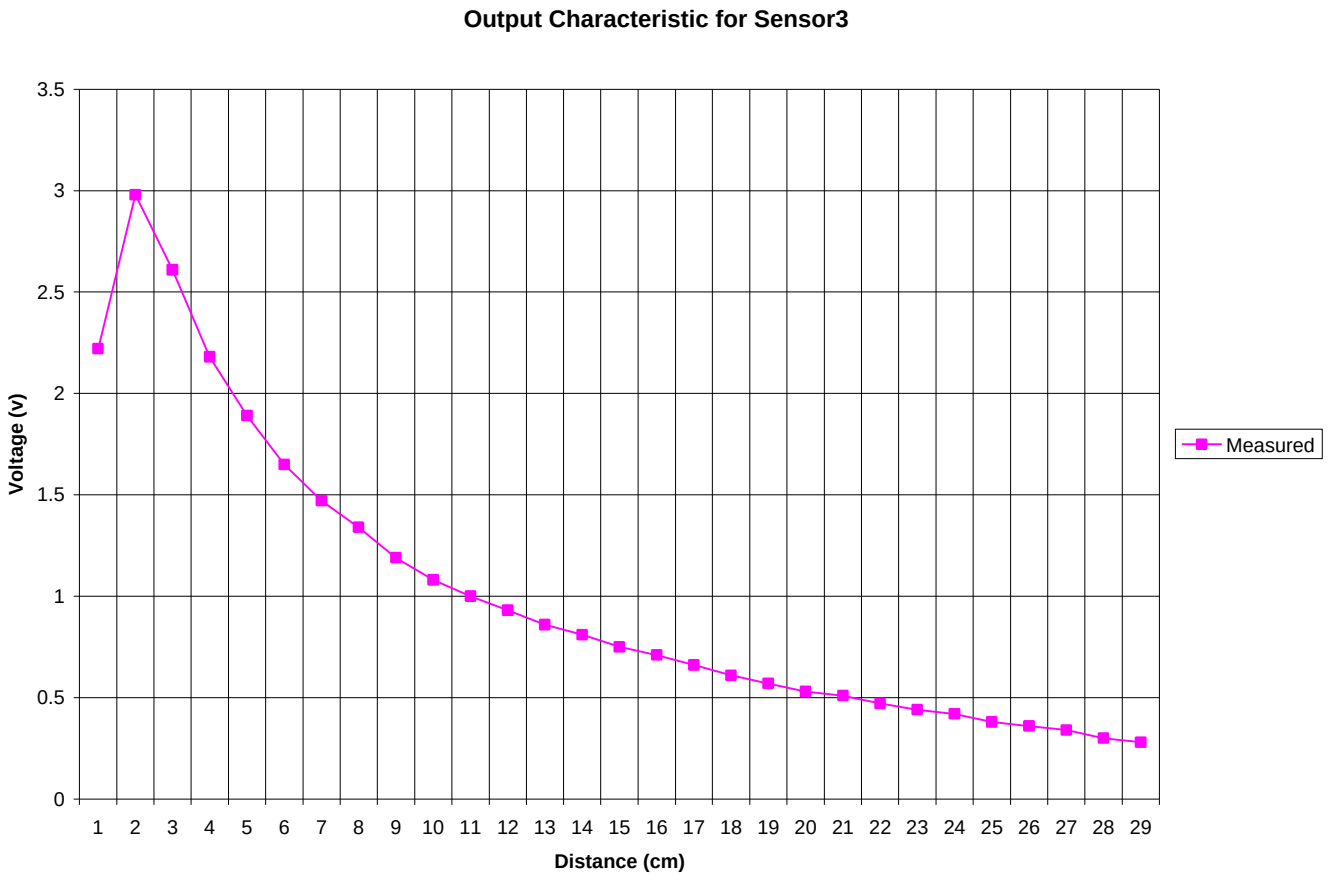

Figure 11. Voltage to Distance Curves

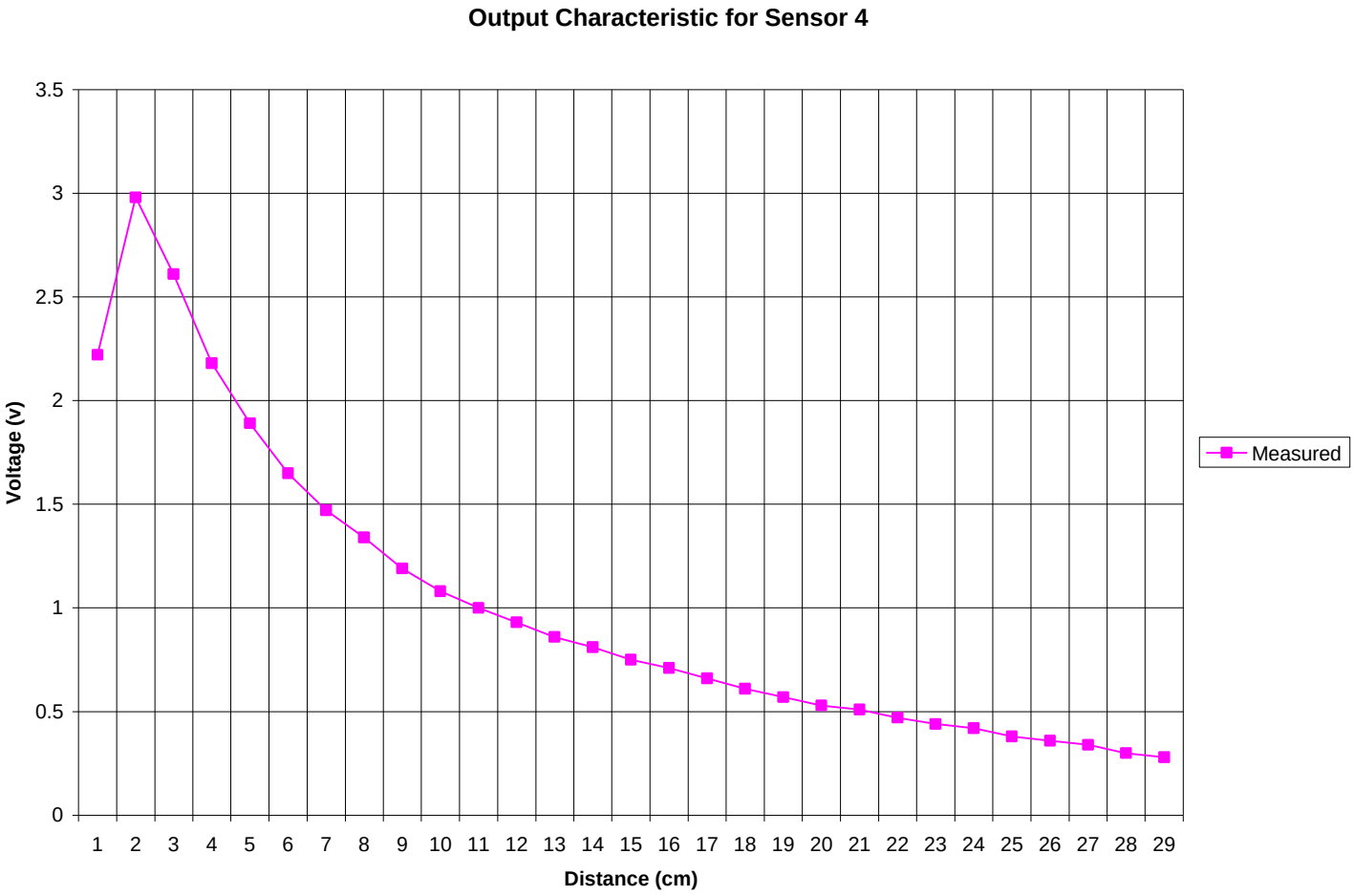

Figure 12. Voltage to Distance Curves

Production Time line for Design

Week1: Jan 16

Basic strategy, task breakdown, manpower allocation.

Week2: Jan 22

Some basic parts collecting, motors, batteries, list for
Establishing a preliminary budget, working on PDR presentation

Week3: Jan 29

Decided on name for robot, continued collecting parts,
Practiced PDR presentation,

Week4: Feb 5

Bench Testing: sensors, power distribution, motor control.
Fabrication: motor mounting brackets, sensor brackets

Week5: Feb 12

Collecting more parts (as many free ones as possible),
Started assembly, started building power distribution board

Week6: Feb 19

Finished assembly, started coding

Week7: Feb 26

REGG1 up and wall following (sort of, not really)

Week8: Mar 5

Functionality review, showed sensor systems working, got the
Altera code to read the encoders at the last minute.
Over budget (not good)

Week9: Mar 19

Showed our smooth wall following
Lost the Hamatmatsu Sensor due to budget constraints.
Also lost one of the 12V lead acid batteries
Acquired donations of rechargeable NiMH AA
Budget back in the black

Week10: Mar 26

Maze navigation coding, pivoting, turning, white line detection, scanning.
Changed wheel base more inboard

Week11: Apr 2

Code for counting rooms, finding room 4,
Candle finding routine doesn't work

Week12: Apr 9

Redesign voltage reference circuit.
Put the fan assembly and Pyroelectric sensor on a single stanchion

Week13: Apr 16

Fire scanning routine doesn't work with the whole program
Made shielded cables including connectors for everything

Week14: Apr 23

Set up return home routines
Relocated Pyroelectric so that it is adjustable to candle height
Replaced the shielded cable for the sensors; too much cross-talk

Week15: Apr 30

Functionality review went well
Documentation, code commenting, pictures
Final paper drafting; electronic and hard copy