EE-382 Junior Design

Final Report

May 9, 2000

Submitted to:

New Mexico Tech

Electrical Engineering Dept.

Dr. S. Bruder

And

Dr. K Wedeward

Submitted By:

Design Team 1

L. Noice

H. Vahle

J. Mares

E. Szpindor

Table of Contents

List of Appendices

List of Figures

List of Tables

1.0 Introduction

1.1 Scope

The document starts with a System Overview to familiarize you with the requirement the drove the final design. Then each subsystem is described in detail including theory of operation for respective electronic circuits. This is all followed by a description of the robot's Initial Operating Capabilities (IOC). The last two sections of the document provide a cost summary followed by conclusions and lessons learned. Following the text sections of this document are appendices that include circuit board schematics and artwork as well as mechanical drawings.

1.2 Purpose

The purpose of this document is to provide a complete technical description of "Vader's Pet".

2.0 System Overview

The requirements for this project were governed in part by the rules of the International Fire Fighting Competition held annually at Trinity College. Additional requirements were place on the design by the Electrical Engineering Department at New Mexico Tech. The added requirements were to mandate that all motion control must be closed-loop and utilize, at a minimum**,** proportional control. In addition to these requirements the design team self-imposed addition requirements. The self-imposed requirements facilitated the team's goals to produce a design that would successfully pass optional challenges in the competition for the purpose of gaining a competitive advantage. The two specific optional challenges that drove the projects design requirements were the use of ramps and furniture. A simplified function block diagram of the system mission is provided in FIG. 1.

FIG. 1 System Mission Flow

2.1 System Level Requirements and Design Goals

The requirements were allocated to logical subsystem with a few remaining that affect the design as a whole. These system level requirements are listed in Table-1. The robot was required to operate un-tethered and autonomously. The nature of the project lends itself to the use of optical sensors. However, the sensors must have a high level of immunity to a verity of lighting conditions that may have the ability to interfere with proper sensor operation. Additional environmental considerations include temperature and humidity for both New Mexico and Connecticut where the international competition takes place. Maintainability and safety were given extremely high priority throughout the design process.

Table-1 System Level Requirements

Description	Requirement	Goal
Navigation and Control	autonomous	
Spectral Noise Immunity		fluorescent/sodium vapor
Operating Temperature	50° to 100° F	
Operate Humidity (non-condensing)	0 to 95%	
Auto Start		3.5 kHz tone
Design for Reliability and Maintainability	X	
Designed for Safety	X	

2.2 Function Description of Subsystems

The system was divided into the following four logical subsystem:

1. System Control Subsystem
2. Power Subsystem
3. Motion Control Subsystem
4. Sensory Subsystem

The System Control Subsystem is responsible for the hardware and software integration of the Motion Control Subsystem and Sensory Subsystem. The subsystem consists of two primary elements: an embedded micro controller and its associated code.

The Power Subsystem includes the power source, associated voltage regulation, over current protection, filtering, and power distribution.

The Motion Control Subsystem consists of those components and circuits that provide locomotion for the robot. Specifically this subsystem includes two compact dc motors and their associated gear heads as well as the optical encoder attached to each motor that are used for velocity and acceleration feedback. This subsystem also includes the power amplifier circuits to drive the motors.

The Sensory Subsystem is a myriad of sensors used to avoid collision with obstacles, detect white lines, detect flames, provide high resolution direction information to direct the robot to the flame, and extinguish the flame.

2.3 Allocation of Resources

To facilitate the research, design, develop, and testing of the four subsystems mentions above, specific responsibilities were assigned to the team members. In addition to the tasks related to the four subsystems, additional efforts were required for the packaging design and general system engineering responsibilities. After a review of each team member's strengths and interests, each member agreed to specific responsibilities to contribute to the overall design. These assignments are summarized in FIG. 2.

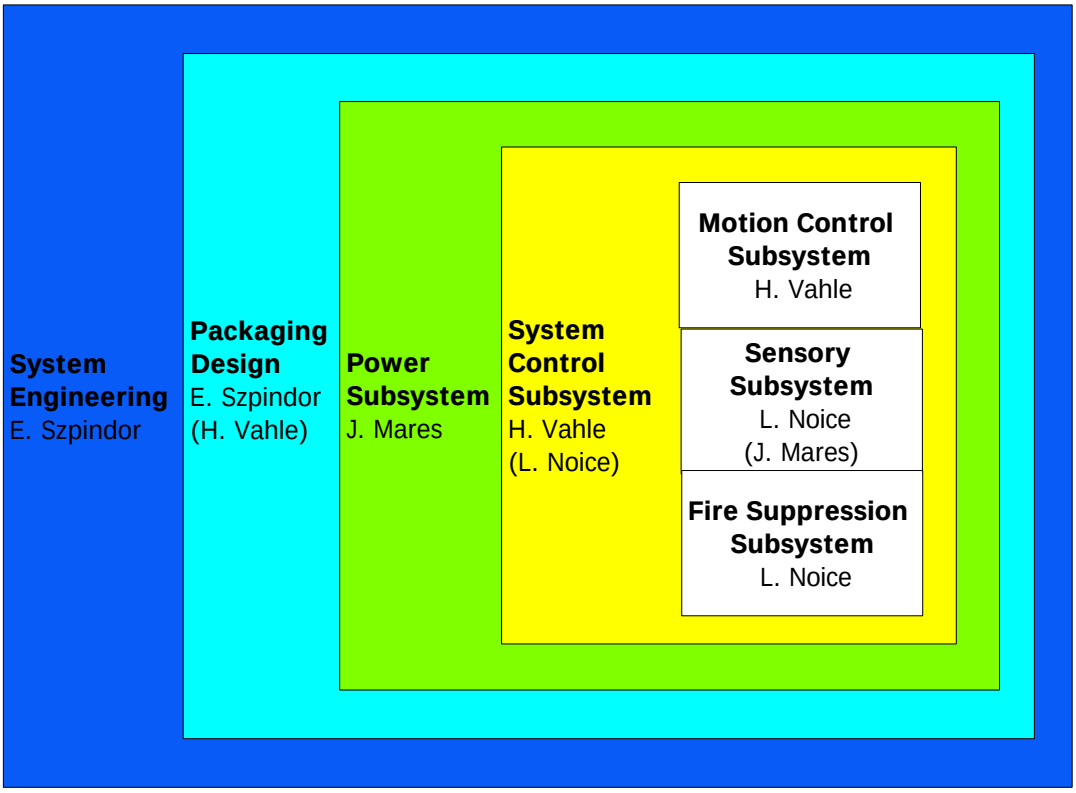


FIG. 2 Resource Allocations

2.4 Printed Wiring Board Summary

Good engineering practices determined the placement of assemblies on the robot. Analog signal lines were kept as short as possible and they were kept separated from inherently noisy or digital lines. Several printed wiring boards (PWB's) were manufactured to accommodate required electronic circuits. The same principles used to determine the placement of assemblies determined the grouping of circuits on the PWB's. Three PWB's were designed and fabricated for this project. The majority of components used on these boards were surface mount. The decisions to use surface mount parts facilitated small boards. The relationship between the circuits on each board, the sensors, fire extinguisher, and embedded controller are shown in FIG. 3.

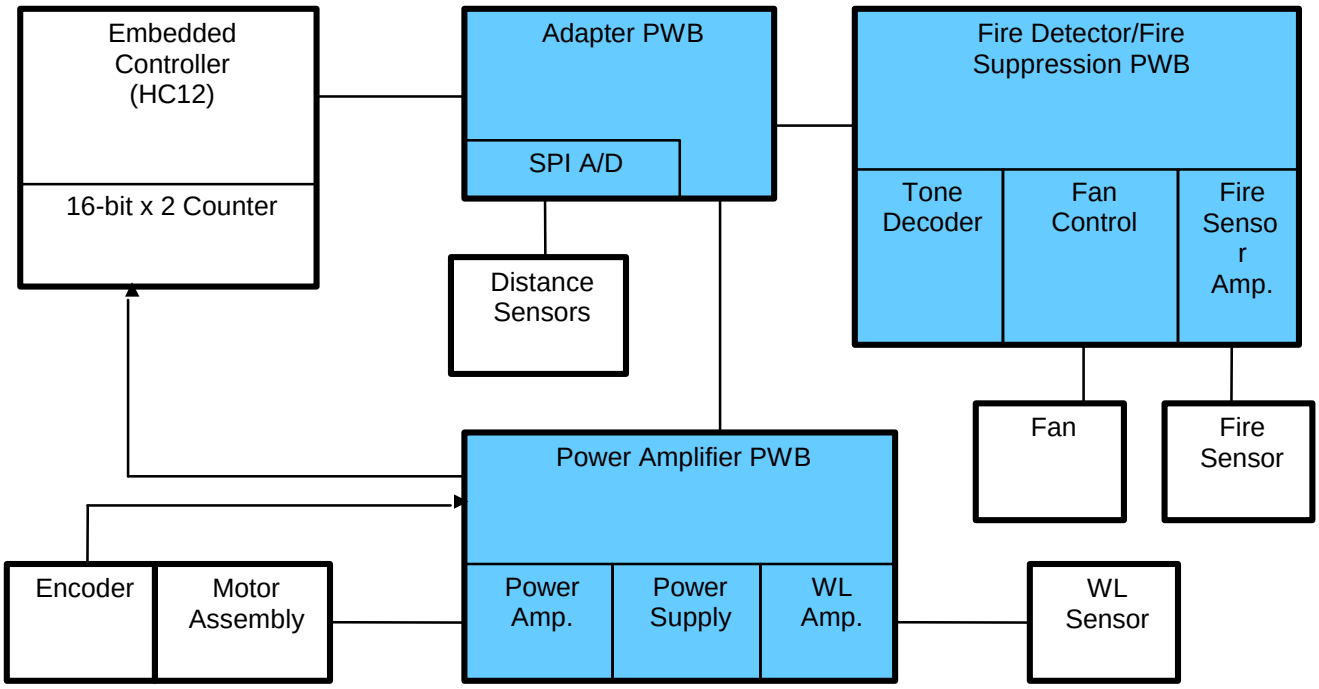


FIG. 3 Printed Wiring Board Summary

Starting in the upper left-hand corner of FIG. 3, the embedded controller consists of a modified Motorola HC12 evaluation board. The modifications include an Altera 7128 PLD. This part is programmed to have two 16-bit counters that are used to count one

of two encoder output lines per motor. The encoders used are quadrature encoders; however, there was no requirement to determine whether the robot was moving forward or backward. Additional details on motion control are included in the sections that follow. Encoder signal were transparently passed through the Power Amplifier PWB to facilitate the use of a single wiring harness to connect lower layer of the robot to the upper layer.

Continuing the discussion by moving to the right to the blue block tilted “Adapter PWB”, the Adapter PWB is the first of three boards designed for this project. The purpose of the Adapter is to provide a reliable interface from the embedded controller to the electronic circuits of the robot. All signals to and from the embedded controller pass through the Adapter with the exception of the encoder signals as mentioned. The Adapter PWB is a mezzanine board by design and it plugs into the embedded controller board. The Adapter board compensated for the poor I/O design of the HC12 evaluation board by providing robust connectorized I/O and distributing ground to each connector to accommodate separating signal lines by ground lines in cables and cable shields if necessary. The Adapter also includes an ADC used to digitize signal from distance sensors, which are used for collision avoidance. This ADC provided additional ADC channels beyond those inherent to the HC12 to accommodate the 11 channels required for this design. The unique feature of this ADC is its glue-less serial interface to the HC12. The ADC is designed to use the Motorola trade marked SPI (Serial Peripheral Interface) for communication to the controller.

In the upper right-hand corner of FIG. 3 is the second board designed for this project. It is again shown in blue and it is called the “Fire Detection/Fire Suppression PWB”. This board holds three groups of circuits. The first circuit, a Tone Decoder, detects and conditions a 3.5KHz tone used for starting the robot remotely. The second circuits amplification of signal from the Fire Sensor Subassembly. The last circuit on this board provides an optically isolated control line to turn on and off a fan motor used to extinguish the fire.

The last board designed for this project is located in the lower central portion of FIG. 3. It is called “Power Amplifier PWB”. This board consists of three mutually exclusive circuits. The first circuit is a power amplifier that provides direction control and the current necessary to for the motor to deliver the required torque. The next circuit is the robots power supply. This circuit connects to the power source and provides over current protection and voltage regulation as well as functioning as a single point ground for the system. The final circuit implemented on this board provides signal conditioning and amplification for the White Line Sensor Subassembly.

2.5 Locating Assemblies

As stated above, the placement of assemblies was chosen for ease of integration; moreover, the placement was chosen to reduce affects of EMI/RFI as will as inductively coupled noise. This becomes increasingly important due to our overall very small design and the close proximity of components. These efforts appear to be quite successful; hence, after hours of testing, no anomalies or degraded of functionality were detected. Furthermore, no shielded cables were used except to connect the “Fire Detection/Fire Suppression PWB” to the fire extinguishing fan motor. FIG. 4 can be used to see the placement of all major assemblies on the robot.

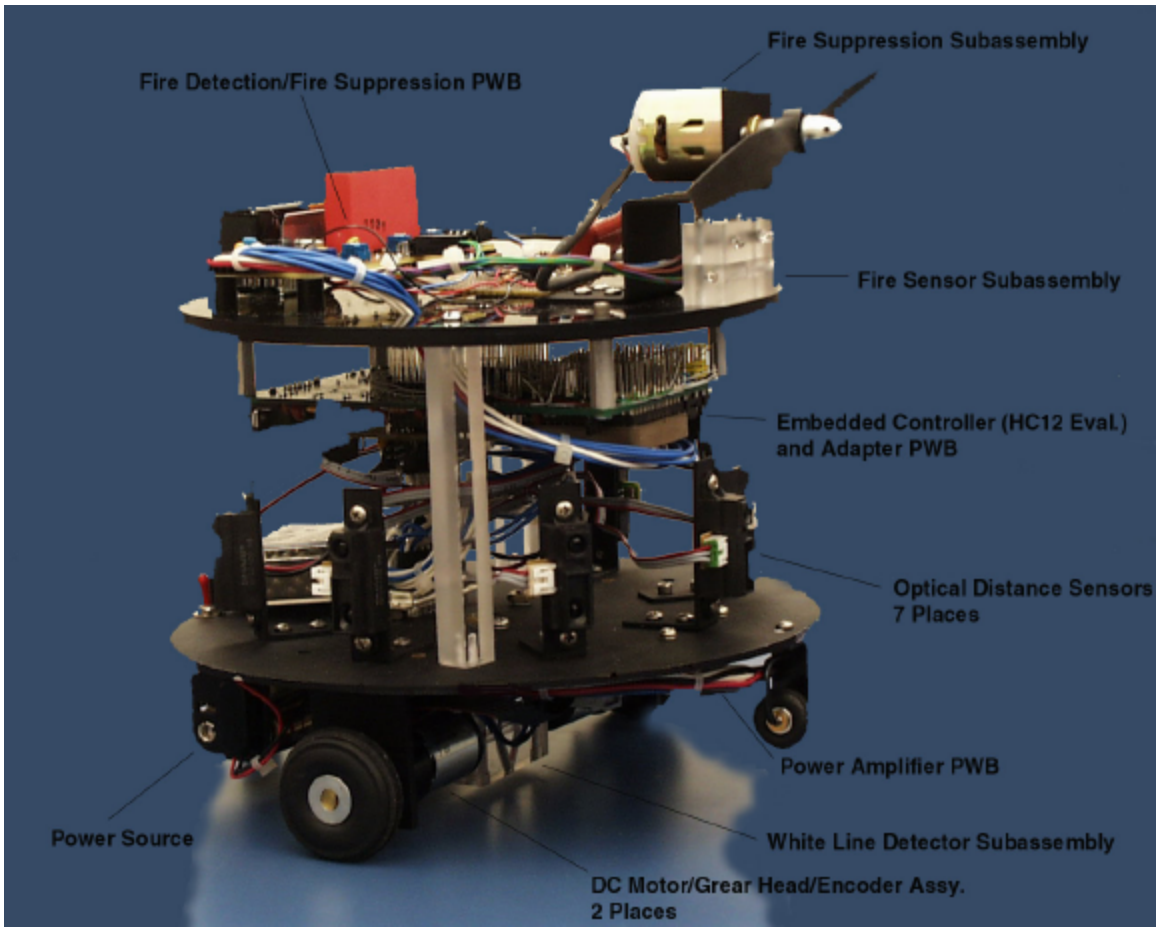


FIG. 4 Location of Main Assemblies

3.0 System Control Subsystem

3.1 Overview

This section contains information about the main flow of the software and the variables and algorithms used in the navigation of the robot through the maze and its search for the candle. Additional information on the software interface to specific subsystems is found in the following sections of this document.

The basic requirements were to navigate through a maze, find a small fire (a lit candle), navigate to the candle, and then extinguish the candle. Additional, requirements are shown in Table 2.

Table 2– System Control Requirements

Description	Requirement	Goal
Search Reliability	search all rooms	
Navigation	no dead reckoning	
Navigation Technique	sensory feed-back	center of hall/doorways
Motion Control	closed-loop	
Max Speed		2 ft/s
Max Torque		transverse ramp
Avoid Obstacles	> 4"	
Max Range Before Fire Suppression	> 12"	
Return Home After Fire Suppression	avoid rooms	shortest path

There were self-imposed requirements for the robot. For example, tone start, furniture avoidance and return home. At this time, the robot has the basic set requirements with one self-imposed requirement; tone start. The function block diagram in FIG. 5 shows the approach taken to initially approach the coding problems. Refer to Appendix 1 for the complete print out of the code used.

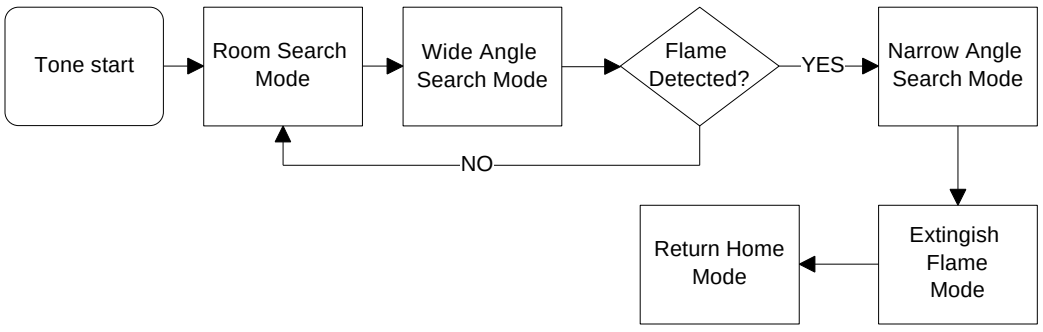


FIG. 5 – Function Block Diagram

3.2 Remote Start

Once the code is downloaded into the robot and the code is run, registers and interrupts are setup. After completing the initial setup, the program enters a tone detection subroutine where it will remain until a tone is detected. A high on PP6 indicated that a tone has been detected. The program checks for a high on this pin twice in 5ms in order to limit the possibility of other sounds setting the robot off. If it is the correct signal then variable *ts* is set high which allows the program to jump out of the loop and enter the main subroutine.

3.3 Speed Control

The main subroutine is where the robot gets its instructions to activate the motors and brings them up to a desired speed using the following algorithm:

lduty = (KL*DS)/100 + KPL*(DS - lms); //Dutycycle for channel 0
rduty = (KR*DS)/100 - KPR*(DS - rms); //Dutycycle for channel 1

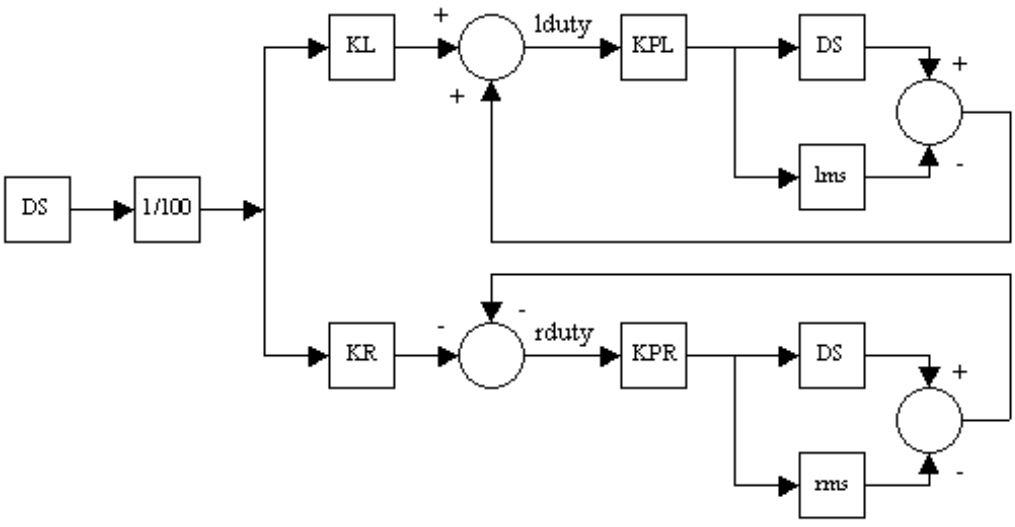


FIG. 6 – Closed-loop Speed Control Algorithm Block Diagram

FIG. 6 is a block diagram of the Closed-loop Speed Control algorithm. *DS* is the desired speed or max speed of the robot. *Lms* and *rms* are the measured speed from

the 16-bit encoder counters, these values are in terms of encoder counts. The measured speed is determined by the RTIF interrupt interrupting every 2ms and reading the information from 16-bit counters programmed in the Altera 7128 on the HC12 Evaluation board. *KPL* and *KPR* are the constants gain terms used to determine the rate at which the algorithm compensates for errors. *KL* and *KR* are the open loop gain constants of the motors, which are multiplied with the desired speed to get a desired duty cycle.

3.4 Left Wall Following

Once the motors are started, the robot will left wall follow using the following algorithm;
lduty = (KL*DS)/100 -(KWL*(DS_3 - s3))/10; //Dutycycle channel 0
rduty = (KR*DS)/100 + (KWR*(DS_3 - s3))/10; //Dutycycle channel 1

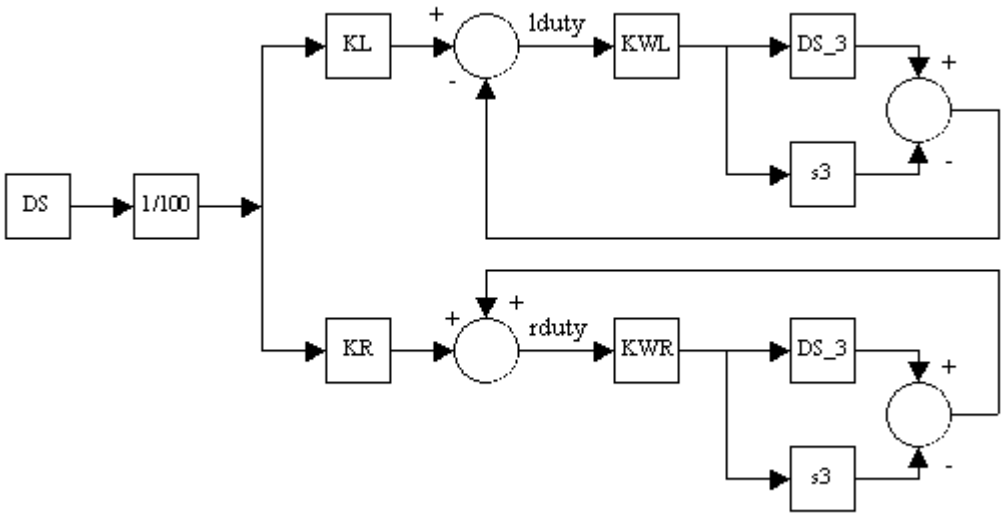


FIG. 7 – Wall Following Algorithm Block Diagram

FIG. 7 is a block diagram of the wall following algorithm. *DS_3* is the desired distance from the wall for the robot and *s3* is the measured distance (from Sharp sensor) from the wall. *KL*DS* and *KR*DS* are the desired speed for the motors. KWL and KWR are the constant gain terms that determine the rate at which the algorithm responds to errors. The main part of the program will run until it either sees:

1. A wall in front of the robot, in which case it turns right.
2. An opening on the left, in which case it will turn left.

3.5 Right Turn

If the robot sees a wall in front of it, it will go into the turn_right subroutine. This in turn will apply a high to PP2 which will reverses one motor. Then using the 16-bit counters the robot will track how many encoder counts to a predetermined count and then stop. Then PP2 will be set low to set the motor back forward and then the robot will continue with the room search.

3.6 Left Turn

If the robot sees an opening to the left it will make a left turn using the following algorithm;

lduty = (KL1*DS1)/100 + (KPL*(DS1 - lms))/10; // DC channel 0
rduty = (KR2*DS2)/100 - (KPR*(DS2 - rms))/10; // DC channel 1

This is similar to the left wall following algorithm except that *KL1*DS1* and *KR2*DS2* are for a slower turn. After the robot has made the left turn it will continue with left wall following. At the same time the robot is looking for a white line which indicates that it has entered a room. Upon seeing the white line the code will go to the *STOP_ROBOT* routine.

3.7 Stop

In the *STOP_ROBOT* routine, the robot will come to a stop by applying a low to PP4 and PP5, which applies a dynamic brake on the “Power Amplifier PWB”. Then the code samples the four fire sensors 16 times and then takes the average of each sensor. The code then takes all four averages and sums them and compares them to *CAL_SUM*, which is a calibrated threshold level of the light from a candle in the largest

room. If the sum of the four sensors is lower than the *CAL_SUM*, then there is no candle located in the room. If the candle is not in the room the robot will make a 180° turn. The 180° turn is accomplished by changing the direction of the right motor. This is done by applying a high to PP2, which will reverse the motor. The robot uses the 16-bit counters to track the encoder counts to a predetermined amount. Then setting PP2 low to switch the right motor to the forward direction will continue left wall following.

3.8 Fire Scan

Once the robot has found a candle in the room, it goes into the *FIRE_SCAN* subroutine. This subroutine will guide the robot to the candle. The algorithm for this is similar to the one used for wall following, but instead of using the sharp sensors it uses the data from the two center flame sensors to guide the robot towards the candle. The algorithm is presented below, and block diagram is shown in FIG. 8.

lduty = (KL*DS)/100 - (KWL*(centerL_sensor - centerR_sensor))/10;
rduty = (KR*DS)/100 + (KWR*(centerL_sensor - centerR_sensor))/10;

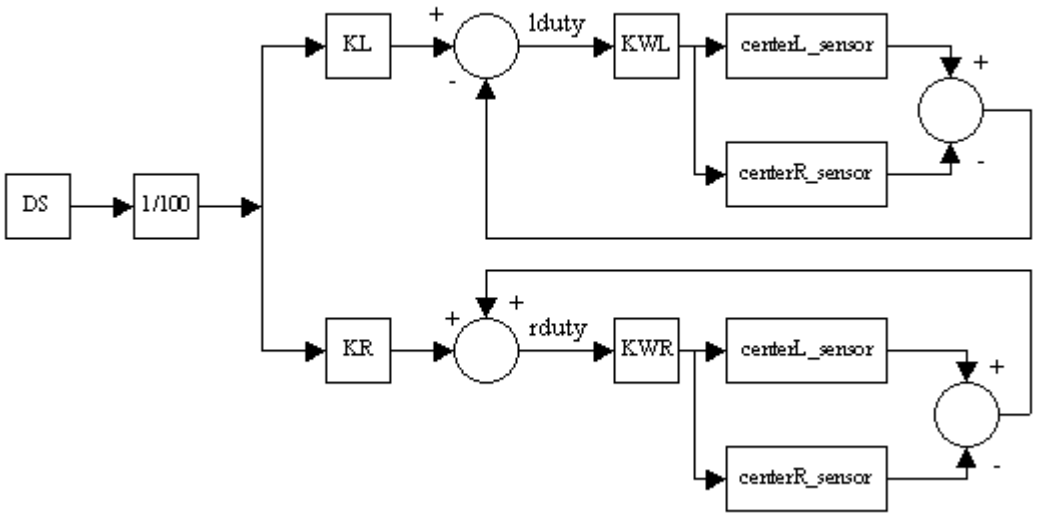


FIG 8 – Fire Location Algorithm Block Diagram

While traveling towards the flame, the robot will be looking for the white line surrounding the candle. Once it arrives at the white line in front of the candle, the program will instruct the fan to turn fan for 5 sec. At which time the flame will be extinguish.

3.9 End_of _Game

At the end of the 5 sec., the fan is turned off and the program then goes to a *end_of_game* routine. Once in this sub-routine, the program is in an infinite loop until the HC12 is reset.

4.0 Power Subsystem

4.1 Overview

The Power Subsystem consists of a power source and associated circuitry that provides over current protection and voltage regulation. The circuit also functions as a single point ground for the system. The requirements for the power subsystem are shown in Table 3, and the system power budget is shown in Table 4.

Table 3 - Power Subsystem Requirements

Description	Requirement	Goal
Power Source	Battery	
Power Source Availability & Versatility		standard cell sizes
Charger Availability & Versatility		standard cell COTS
Capacity (w/ operator intervention)		one day (10 trials)
Capacity (w/o operator intervention)	6 minutes	
Power Source Current	exceed sum of all surge and average current requirements	
Equipment and Personnel Safety	over-current protection, and design w/ margin	
Power Supply Voltage(s)	match all subsystem voltage(s)	

Table 4 - Power Budget

SUBSYSTEM	DESCRIPTION	QTY. REQ.	Current		Voltage (V)
			AVG. (mA)	Peak (mA)	
Motion and System Control Subsystem	Assy., Embedded Controller		250		5
	Assy., Motor				
	Motors	2	800	2300	9
	Encoder	2	16		5
	Assy., Power Amplifier Board PWB	1	63		9
Sensory Subsystem	Assy., Distance Sensor	3	43		5
	Assy., White-Line Sensor	1	90		5
	Assy., Fire Detection Sensor	4	50		
	Adapter Board	1	53		
Fire Suppression Subsystem	Fan Motor	1	986		9
	Fire Suppression/Fire Extinguishing Board	1	20		N/A
		Total	**2371**	**4671**	

4.2 Primary Power Source

The system power source is a battery pack consisting of eight AA size cells. Nickel Metal Hydride rechargeable batteries were selected for their high current density of 1500 mAH. With eight 1.2 V cells connected in series, the total voltage of the battery pack is nominally 9.6 volts.

4.3 Circuitry

This battery pack is connected to an on/off switch that is in turn connects to the “Power Amplifier PWB” on J4. The schematic for this board is in Appendix 2 page 1. The 9.6 V line is passed through re-settable fuse F1, filtered by bypass caps C3 and C4, and applied to power amplifiers U1 and U2. After filtering, the 9.6V line is also connected to a voltage monitoring circuit who’s output is fed to the HC12 via J5. This interface allows the HC12 to determine when batteries should be replaced. The voltage monitor circuit is a set of resistors used to divide the voltage and limit the current to a range acceptable for the HC12’s internal A/D converters. Finally, the 9.6 V line feeds a voltage regulator circuit that provides 5 V for distribution throughout the robot. The regulator, U3, is a 7805 capable of delivering 500 mA. The output of the voltage regulator is connected to the “Adapter PWB” via J7. Power is distributed to all 5V

subsystems from the “Adapter PWB”. The power distribution to other subsystems is shown in Figure 9.

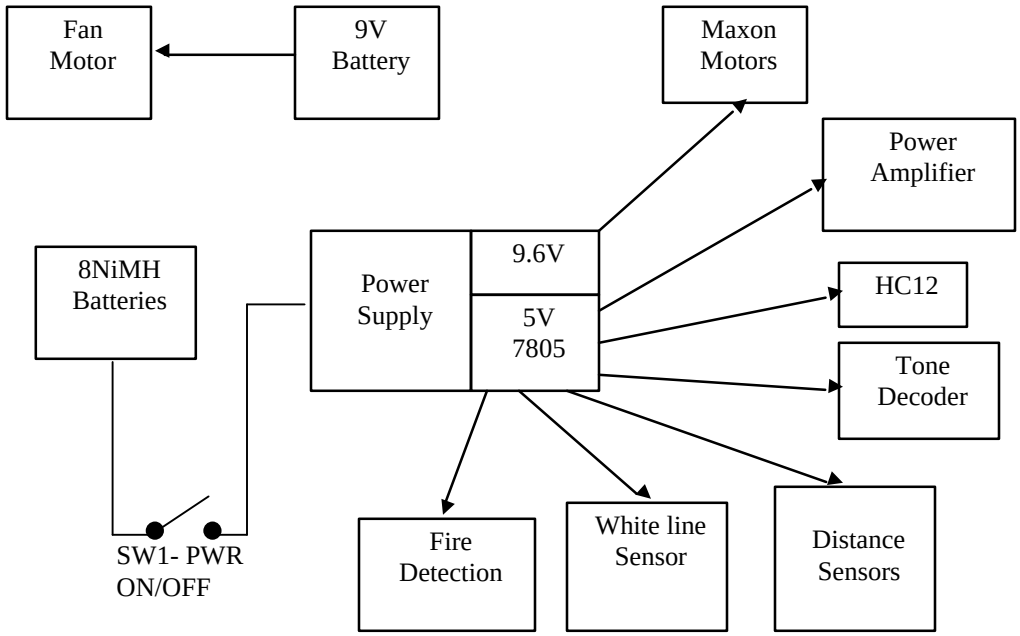


FIG. 9 – Power Distribution Block Diagram

4.4 Charge/Discharge Characteristics

This robot has a 2-hour operation life, which is defined as a 2V drop with respect to full charge. The battery charger has a quick charge mode that recharges all eight batteries in approximately 1.5 hours. The battery change is not integrated into the robot. Two battery packs are used such that a charged pack is always available.

4.5 Fan Power

The fan motor is powered from a separate 9V alkaline battery. The decision to use a separate power source for the fan motor was based on a desire to reduce the load on

the primary power source. Voltage is applied to the motor through an optically isolated relay controlled by a logic line from the System Controller (HC12).

5.0 Motion Control Subsystem

5.1 Overview

One of the requirements of the design was to have a closed loop Motion Control Subsystem. The design uses differential drive. The Motion control subsystem is made up of 3 major components; the Embedded Controller Assembly, the Maxon Motor Assembly, and the Power Amplifier Assembly as shown in FIG. 10. The Maxon Motor Assembly is shipped from the factory as one unit, but it consists of a motor, gear head, and encoder.

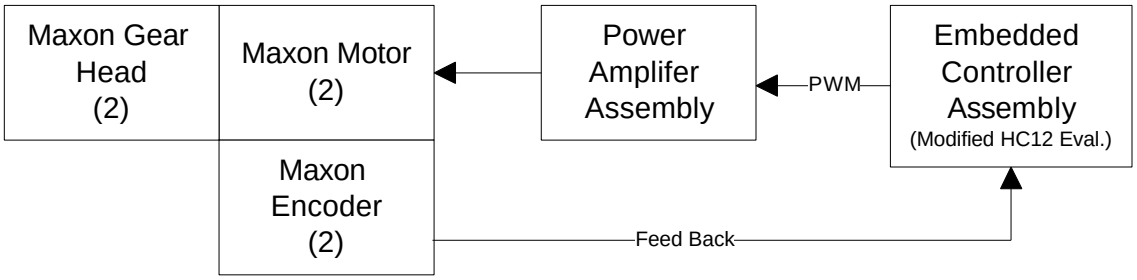


FIG. 10 – Motion Control subsystem

5.2 Signals and Circuits

The HC12 generates two Pulse Width Modulation (PWM) signals; one for the left motor, and one for the right motor. The signals are called PWM_IN_A and PWM_IN_B. The PWM_IN A&B signals are output on I/O pins PP0 and PP1 respectively. The two signals connect to the “Power Amplifier PWB” on J1 pins 6 & 10 after passing through the “Adapter PWB”. The Power “Amplifier PWB” schematic can be found in Appendix 2 page 1. PWM_IN A&B connect to pin 6, the /ENABLE input, of U1 and U2 respectively. U1 and U2 are Allegro A3952SLB full-bridges. The /ENABLE inputs are used to pulse width modulate the high current drive to the two motors. The output pins 10 & 15 of U1 and U2 are connected to J2-1 & 6 and J3-1 & 6 respectively.

J2 and J3 connect to the Maxon Motor Assemblies using a ribbon cable that is part of the Motor Assemblies.

The encoders on the motors provide a quadrature output (two signals 90 deg out of phase) with 16 counts per shaft revolution. It was not necessary in this design to use the quadrature information, so only one of the two encoder signals is used per motor. With the motor turning, the encoders send a pulsed signal back up the ribbon cable to J2-3 ENCODE_A and J3-3 ENCODE_B on the “Power Amplifier PWB”. The ENCODE_A & B signals simply pass through this board and exit on J1 pin 1 and 9 respectively. The ENCODE_A & B signals are brought to the HC12 Evaluation board’s J1 pin 19 and 20 respectively after passing through the “Adapter PWB”. J1 pin 19 and 20 connect to an Altera 70128 PLD mounted on the HC12 Evaluation board. The Altera part is programmed with two 16-bit counters; one for the left motor, and one for the right motor.

The Motion Control part of the software running on the HC12 samples the two 16-bit counters every 2ms reading memory mapped registers on the Altera 7128. The count values are used to determine how fast/slow the robot is going. The motion control algorithm uses this information, as discussed in the System Control section of this document, to adjust the PWM and maintain a desired speed. This gives the robot closed loop motor control.

In addition the phase pin of the Allegro part on the “Power Amplifier PWB” is used to control the direction of the motors. This signal is used to perform 180 degree turns when entering a room and finding no candle. The brake function was also used on the Allegro part to provide rapid stops and minimize overshooting of white lines.

One of the unusual things about the Allegro A3952SLB was that the output of the chip was inverted from the input. This caused a 100% duty cycle applied to the input of the Allegro part to be output as 0%, so this had to be taken in to account when adjusting the PWM signal.

6.0 Sensory Subsystem

6.1 Tone Detector Subsystem

The robot has the ability to be started remotely. To accommodate the International Fire Fighting Robot rules, the remote start functionality was implemented using a 3.5 kHz tone detector. The tone detection circuit consists of a microphone, an amplifier, a commercially available tone decoder. The tone decoder selected was a National Semiconductor LM567. The part requires external components to set the center frequency and the bandwidth of the sweep oscillator. The initial values of the external component selected for a desired fc = 3.5 kHz and BW = 10% * fc using equations from the manufacturers data sheet. After tuning, the actual values used varied slightly to account for component tolerances and stray reactance in the circuit layout. The circuit was incorporated into the “Fire Detection/Fire Suppression PWB”, and the schematic can be found in Appendix 2 pg. 4

6.2 Distance Sensor Subsystem

6.2.1 Hardware

One of the most important aspects of this project is being able to navigate through the maze. In order to do this, reliable distance sensors needed to be attained. After reviewing other distance sensor technologies, the GP2D120 was the down-selected sensor. The GP2D120 were made available to this team as well the entire junior design class due to the efforts of Herald Vahle’s research last summer.

Seven sensors were used for obstacle avoidance to include walls and furniture. The arrangement of the distance sensors is shown in FIG.11. Only three were used for navigation (Sensors 3, 4 and 5). The remaining sensors (Sensors

1,2,6 and 7) were intended to provide additional information for furniture avoidance. This feature was not fully implemented through in the robot's IOC.

The output of the distance sensors could be fed directly into the HC12's A/D without signal conditioning. However, the HC12's internal A/D has only six of eight channels available. As a result, an external ADC was used to digitize the seven distance sensors. The GP2D120s were nicely packaged, and they are specified to run on a 5V source at 50ma. These sensors have an indeterminate region between zero and three centimeters that requires specific attention. An effective way to remove this anomaly is to offset each sensor three centimeters from the edge of the robot.

Sensor placement every 45° provides symmetry and nearly 360° coverage. S3 and S5 were chosen as opposed to s2 and S6 for navigation because outputs from sensors placed at 45° change less rapidly than sensors placed at 90° for the same change in distance and time. This effectively smoothes the output of the left and right navigational sensors. Another feature to placing the left and right navigational sensors at 45° is that it gives the sensors the ability to effectively see ahead. This allows the robot to navigate faster thought the maze because it allows the robot the anticipate turns.

The darkened sensors are the ones used for navigation. Each sensor was numbered from 1 thru 7 starting from left to right. Doing this helped keep track of the sensor values through software.

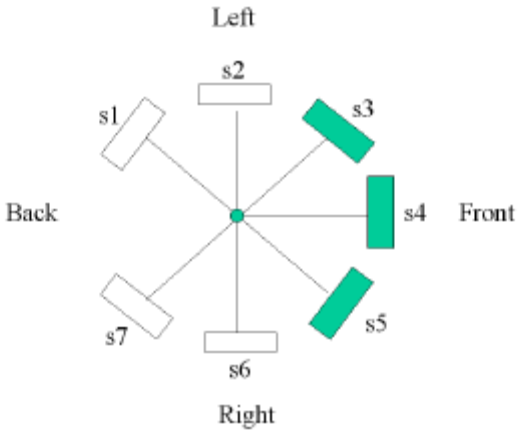


FIG. 11 - Placement of distance sensors

Since the transfer function of each sensor is unique, they were completely characterized and the results can be seen in Appendix 3 page 1. Although only seven sensors were used, nine sensors were characterized. The measured transfer function for each sensor is plotted against the manufactures characteristic transfer function for each sensor, and the entire data set is available in Appendix 3 page 1. One sensor was not used because its transfer function deviated grossly from the manufacture's characteristic curve as shown in FIG. 12. A second sensor was not used because it was unstable in nature.

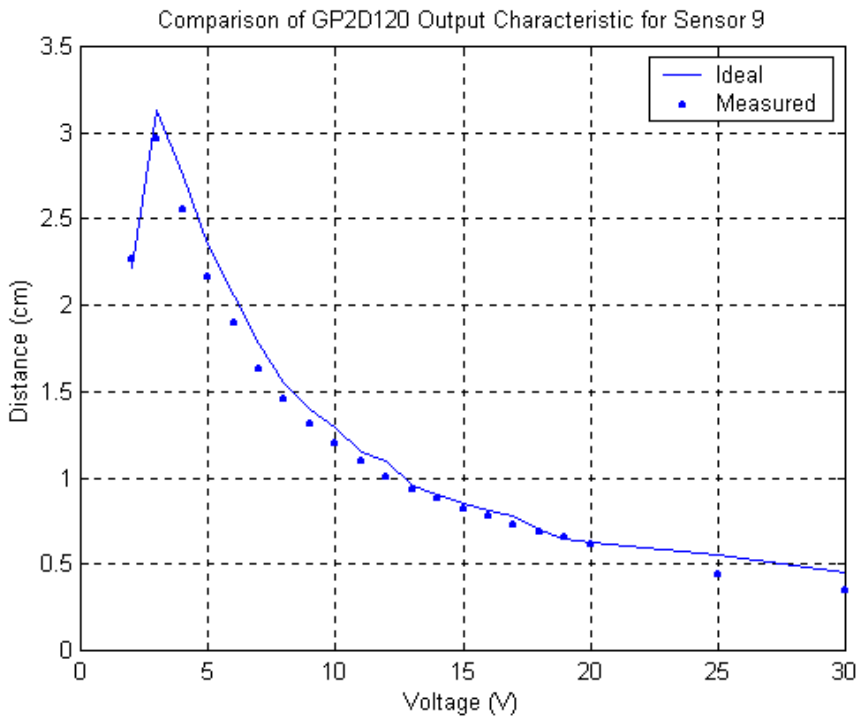


FIG. 12 – TF of Sensor 9

The data obtained from this characterization was used to match the transfer functions (TF) of complimentary sensor pairs such that:

1. TF S3 ≈ TF S5,
2. TF S2 ≈ TF S6,
3. TF S1 ≈ TF S7.

The same data was also used to determine threshold values used in software to maintain desired distances from objects.

6.2.2 Software

Since the robot is only eight inches in diameter, keeping it five inches away from walls allows it to always be centered in hallways and doorways. In order to do this, the data collected from the hardware characterization of the sensors

was used to determine software threshold values that correspond to a five inch distance. These threshold values were in A/D bit counts and were applied to the motion control algorithm described in the system control section of this document.

Although the original design goals were to include furniture avoidance, the software portion of this functionality was not implemented in the IOC.

6.3 White Line Sensor Subsystem

6.3.1 Hardware

One of the most important functions that the robot must be able to do is to detect white lines. The home base is the white circle from where the robot starts each run. Additionally, a white line is placed at each doorway, and a white line is placed around the candle.

The white line sensor must be immune to noise sources that include false detection from debris on the maze floor and spectral contamination from ambient lighting conditions. The immunity to false detection of debris was accomplished by manual calibration by adjusting the gain of a non-inverting amplifier. The immunity to spectral contamination was achieved by the over mechanical design and placement of the white line sensor.

The first technique employed to reduce contamination from ambient light was to place the white line sensor in the center of the robot. This provides an eight-inch aperture block for the sensor. Added benefits to placing the white line sensor in the center of the robot include:

1. When a doorway white line is detected, the robot is protruding halfway into the room. This allows the flame detector to not have its aperture blocked by the doorway itself, thus providing a wider field of view to the flame detector.

2. When the robot detects the white line surrounding the candle, the robots flame suppression assembly is four inches closer to the flame than if the white line sensor was placed at the leading edge of the robot.

The second technique employed to reduce contamination from ambient light was the mechanical packaging of the white line sensor. The photo emitting and photosensitive areas of the sensor are placed 0.2” from the floor. Placing it in a black tube reduces the field of view of the photo detector. In addition, the geometric relationship between the photo-emitter and photo-detector are set such that the angle of incidence to the floor is equal to the angle of reflection from the floor at a sensor height of 0.2”. This configuration is shown in FIG. 13. Finally, the placement off the floor, reducing the field of view of the detector, and angular relationship between the detector and emitter combine to reduce the possibility of contamination by ambient light to near zero.

The design used a PN168 phototransistor and a red LED, which were housed in a piece of milled clear lexan. The driver circuit for the emitter and signal conditioning for the detector were placed on the “Power Amplifier PWB” (See Appendix 2 page 1).

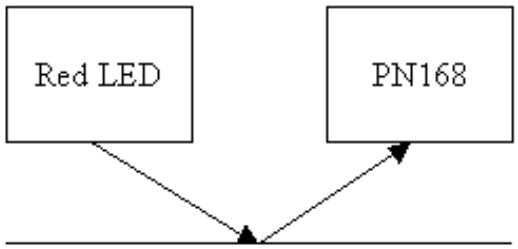


FIG. 13a - White line sensor

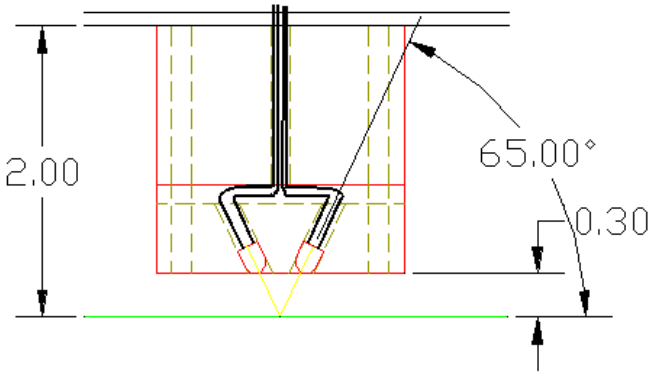


FIG. 13b - White Line Sensor

The emitter's anode was connected to J8 pin-4 that connects it to VCC through a 47Ω resistor used for current limiting (light intensity).

The phototransistor's collector was connected to VCC through J8 pin 1. The phototransistor's emitter is current limited by a 100KΩ resistor and connected to U6 via J8 pin 2. U6 is a non-inverting amplifier with an adjustable gain. The gain was adjusted such that when the robot was over a white line, the output of the op-amp was over 4V and when the white line sensor was not on a white line a voltage below 1V was read. The output of the op-amp was connected to an I/O port on the HC12. The gain of the op-amp was high enough to trigger the TTL I/O port of the HC12 such that no threshold devices were required for the interface.

The above design worked great. The white line sensor was able to be adjusted such that all debris was ignored. The white line sensor was tested in a room

that had a lot of wax drippings in it. The white line sensor designed passed with flying colors by never detecting the wax and best of all, the white line sensor was able to work in different lighting conditions. Because of the careful design of the white line sensor and its associated circuitry, the software implementation was simplified.

6.3.2 Software

The output of the white line sensor was fed to PT0 on PORTT. The RTIF was used to detect when PT0 was asserted. Once this bit was asserted, this would set the white_line variable high in the main while loop where the motion control code resided.

Once a white line has been detected, two other conditions must be met in order for the robot stop on doorway white line or a candle white line.

Normally, when the white line variable is asserted, this indicates that we've entered a room or have detected the white line surrounding the candle. Either of these two conditions should immediately stop the robot. However, two other conditions need to be taken into consideration because they will cause the robot to stop prematurely. The two conditions are:

1. Falsely detecting the home base at the beginning of run as a white line to a doorway.
2. Falsely detecting a white line when leaving a room.

The following algorithm was used to detect a white line taking in account of the two conditions mentioned above.

```
if (white_line == 1 & line_startl == 1 & line_startr == 1)
{
```

```
stop_robot(); // White line detected, stop robot
}
```

The variable line_start1 handles the first condition. False detection occurs at the beginning of the run when the robot is set down on the home base. It interprets the home base as a white line and thinks it has entered a room. To avoid this situation, line_startl bit was initialized low. While this bit is low, the white line will be ignored until this bit is set high. The only time that the line_startl bit is set high is when we make the first left turn and stays set high throughout the duration of the run.

The variable line_startr handles the second condition. False detection occurs after the robot detects a doorway white line and has determined that there is no fire in the room, the robot is programmed to make a 180° turn to search for the flame in the next room. Once the robot completes the 180° turn, it is still over the white line causing the robot to think it has gone into another room. To resolve this situation, the variable line_startr is used. This bit is initialized high and after the robot completes a 180° turn, the bit is pulled low. With the bit pulled low, the robot will ignore white lines until it makes a clockwise 90° turn. Once it has completed the 90° turn, line_startr is pulled high which prepares the robot to look for the next white line.

This technique worked extremely well, and it is repeatable on every test run.

6.4 Fire Sensor and Fire Suppression Subsystem

6.4.1 Hardware Fire Sensor

One of the main features that the robot must be able to do is that it must be able to accurately detect a flame in each of the four rooms. Much research was done to figure out which sensors would be the best to detect a flame. The best

sensors to detect a flame were the ultraviolet sensors. These are the ideal sensors because they are oblivious to ambient lighting, but the drawback to these sensors is that they are expensive. The other sensors of choice are the cheap infrared (IR) sensors.

It was decided that the IR sensors would be implemented for this project. There are a variety of IR sensors. To decide which sensor would be the best one for this project research was done on the lighting conditions that the robot would be operating under. The two lighting conditions that were of concern was the light emitted from HPS lamps and the light emitted from the florescent lights.

To figure out the affects of these lighting conditions, a spectral graph was attained (See Appendix 4). From this graph, it was learned that florescent light has about a 470nm wavelength and the HPS lights have a wavelength in the range of 570nm to 620nm. Taking this into consideration helped narrow down which IR sensors to use. The Panasonic PN168 NPN Phototransistor has a peak sensitivity wavelength at 800nm. This was way out of the spectral range that was of concern and this was the sensor implemented for this design.

The PN168s have a +/- 30º field of view measured at the 3dB point. To determine if a flame is present in a particular room one would like to have a field of view greater than 180º. Beyond the requirement for a wide field of view for flame detection, the robot must be able to accurately determine the direction to the flame. These two requirements lead to two specific modes of operation for the flame detector assembly. The designers called the two modes: wide-angle mode, and narrow-angle mode. Four PN168s were used to accommodate these two distinct modes of operation. The detailed discussion below identifies the differences between the two modes.

The phototransistor bias circuit and signal amplification circuitry is located on the “Fire Detection/Fire Suppression PWB”. For a schematic of this design,

please refer to Appendix 2 page 4. Each PN168s was implemented the same way. A 100? resistor was connected from the emitter to ground, the output was taken off the emitter and fed to a non-inverting op-amp with an adjustable gain, and the output of the op-amp was then fed to an A/D channel in the HC12.

Wide Angle

The wide angle search mode is a mode that enables the robot to search the entire room to determine if there is a fire in the room. To implement this design using PN168s, four PN168s were used as mentioned. The configuration of PN168s is shown in FIG. 14. Four PN168s were used because each PN168 has a half intensity beam width of 30º or a total beam width of 60º.

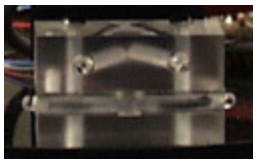


FIG. 14 – Flame Sensors

The self-imposed requirement was to have the robot position itself halfway into a doorway and quickly determine if there was a fire in the room without having to sweep the room to detect a fire. Positioning the robot halfway into a doorway is simplified by having the white line sensor assembly located in the center of the robot. Using four PN168s, equally spaced provides an aggregate sensor with a resulting field of view in excess of 220º. Certainly a challenging aspect to this design is the packaging of the sensors.

Narrow Angle

The narrow angle search mode is a mode that enables the robot to pinpoint the exact location of the flame in the room. The design goal was to provide high

angular accuracy and resolution with the PN168s and without the use of optical techniques to reduce the field of view.

To implement this design, a nulling technique was used. Since the beam pattern of a single PN168 was known, two PN168s were positioned such that there was a small overlap between the half angle/half power point creating a sharp null between them. This sharp null would allow the robot to precisely determine the exact location of the candle, while combined the two sensors have a field of view in excess of 120º.

The uniqueness of this design is that wide angle phototransistor were used to quickly acquire and maintain high-accuracy and high-resolution angular direction to the flame. No optical techniques were needed on the phototransistor or the flame detector sensor assembly as a whole to reduce the field of view for better angular resolution or to block out ambient light.

6.4.2 Software Fire Sensor

Wide Angle Mode

Once the robot has entered a room, the robot enters a wide scan mode. Sixteen samples are taken from each phototransistor and the average value is computed. By averaging 16 samples per phototransistor, noise fluctuations are reduced by a factor of 4. This value is compared real-time to a threshold value called, CAL_SUM. If the current measured average is greater than CAL_SUM, then a fire is detected. Otherwise, no fire is present and the robot enters the room search subroutine to look for a fire in a different room. This technique worked really well and detected a flame every time. The CAL_SUM value was determined by placing the robot in the largest room with the candle placed in the furthest corner. The robot then takes samples under current lighting condition. Again 16 samples per phototransistor are collected and averaged. The average is stored in the variable CAL_SUM.

Narrow Angle Mode

Once the robot has determined that there is a fire present in the room, it will go into a narrow angle mode. This mode will find the exact position of the candle so that the robot can quickly and accurately move to the flame.

Only the two center fire sensors are used to find the flame. From here the value of the left center sensor is subtracted from value of the right center. When the difference of those two is negative, the left center sensor is seeing more light that the right center sensor, so robot will veer left. The opposite it true as well. When the difference of the sensors is negative, the right sensor is seeing more light than the left, and so the robot will veer right. When the difference of the two is equal to zero, then the two sensors are seeing an equal value and the robot will track straight.

6.4.3 Hardware Fire Suppression

Once the robot has successfully detected a flame and is ready to put it out, the robot must have a reliable fire extinguisher to put out the flame. There are many different ways to put out a flame. After reviewing the many different possibilities, a 9V DC motor with an airplane propeller blade was down selected. Although it is not practical to put out of fire by trying to blow it out, this was the method of choice since the fire the robot had to put out was only a small flame.

There are many benefits to using the fire extinguisher selected. The 9V DC motor used in this design was small, powerful, and implementing it through hardware was simple. An airplane propeller was easily installed onto the motor, with the only drawback being that the propeller had to be positioned vertically before each run or it would block the fire sensors.

Since the 9V motor only draws about 1A, an external 9V battery was added to power the 9V DC motor. Doing this would take off some of the power drain to our main batteries and would alleviate any noise caused by this noisy motor.

In order to drive the fan, an optically isolated relay was chosen. Using this relay would reduce the circuitry and would prevent any EMI produced from the DC motor from entering other circuitry. An I/O pin from the HC12 was connected to pin 3 of the optical relay. When this pin was pulled high by the I/O pin from the HC12, the motor would turn on, when pulled low, the motor would be turned off. The full implementation of this design was can be seen on the “Power Amplifier PWB” (See Appendix 2 page 1).

The 9V DC motor with the attached propeller proved to be an effective fire suppressor. The 9V battery was able to provide enough power to the DC motor, and the propeller used was able to provide enough air flow that such it provided a ±15° angle of extinguishing. With the combination of the accurate fire detection subsystem and effective fire suppression subsystem, the fire was put out on every trial.

6.4.4 Software Fire Suppression

Once the robot has detected a flame, driven up to the flame and has stopped on the white line surrounding the candle, the robot is ready to turn on its fire extinguisher.

To do so, as mentioned, the output of PT3 is fed to pin 3 on the optical relay. Once the white line surrounding the candle has been detected, PT3 is then set high for approximately five seconds using a simple delay subroutine. This causes the fan to stay on long enough to extinguish the flame. Once the period of five seconds is up, PT3 is then set low and the fan turns off. As mentioned above, because of the reliability of the fire sensors and fire extinguisher, no

other code was needed to compensate for the event that the robot failed to put out the flame.

7.0 Cost Summary

In general, the majority of components used on this robot have been donated by the team members or vendors/manufacturers. The design team stayed well below the $100.00 department budge. Most of the expenditure accrued on the department account were from the Instrument Shop and included expendable material. Table -5 is a summary of the cost incurred on the project as well as value the of donated material.

ITEM	DESCRIPTION	EE Department (Budgeted)			Donated Parts		
		QTY.	Unit Price	Extended Price	QTY.	Unit Price	Extended Price
A1	Assy., Embedded Controller			$0.00	1	$130.00	$130.00
A2	Assy., Motor			$0.00	2	$108.00	$216.00
A3	Assy., Power Amplifier PWB	1	$11.51	$11.51	1	$16.00	$16.00
A4	Assy., Tail Wheel	1	$1.50	$1.50			$0.00
1	Drive Wheel	2	$1.00	$2.00			$0.00
A1	Assy., Distance Sensor			$0.00	7	$8.00	$56.00
A2	Assy., White-Line Sensor			$0.00	1	$4.00	$4.00
A3	Assy., Fire Detection Sensor	2	$5.18	$10.36	2	$2.00	$4.00
1	Adapter PWB	1	$11.51	$11.51	1	$10.00	$10.00
A1	Assy., Motor/Fan (fire suppression)			$0.00	1	$14.50	$14.50
1	Fire Detection/Fire Suppression PWB	1	$15.51	$15.51	1	$38.00	$38.00
A1	Assy., Power Source			$0.00	16	$4.25	$68.00
1	Battery Charger			$0.00	1	$35.00	$35.00
A1	Assy., Cable- Adapter PWB to FDFS PWB			$0.00	1	$4.00	$4.00
A2	Assy., Cable - Adapter PWB to PWR Amp			$0.00	1	$10.00	$10.00
A3	Assy., Cable - Battery to PWB			$0.00	2	$1.00	$2.00
1	Mounting Plate			$0.00	1	$10.00	$10.00
2	Misc. Hardware	1	$4.48	$4.48	1	$30.00	$30.00
			Subtotal	$56.87		Subtotal	$647.50
						Total	$704.37

Table 5 - Summary of cost

8.0 Initial Operating Capabilities

After hours of test, the initial operating capabilities of the robot have been fully characterized. All projected directed requirements have successfully meet. In addition to the project requirements, the robot also has full remote start functionality. Two of the self-imposed goals were not functional at IOC testing. The first was the return home function. This function has been coded, but the code has not been integrated and tested. The other design goal was to take on furniture. Although all hardware have been implemented and test, the code has not been written. All of the current functionality is repeatable. Table 6 summarized the IOCs of the robot.

Item	% Functionality	Repeatability	Notes
Tone Detection	99	99	Noise Immunity
Autonomous Navigation	99	99	Closed-loop
Search all Rooms	99	99	without collisions
Locate White Lines	99	99	Noise Immunity
Detect Fire	99	99	from door of largest room
Locate Fire	70	99	high angular resolution
Extinguish Fire	99	99	wide angle
Return Home	0	0	

Table 6 - IOC SUMMARY

Appendix 1

```
#include "hc12.h"
#include "DBug12.h"

#define  TRUE 1
#define  PERIOD 200                // prf = 5KHz(prescaler=32)
#define  DS 25                     // Desired speed - 1.5"
#define  KL 216                    // Conversion constant for left motor
#define  KR 220                    // Conversion constant for right motor
#define  KPL 10                    // Proportional constant for left motor
#define  KPR 10                    // Proportional constant for right motor
#define  KWL 10                    // Wall follow constant for left motors
#define  KWR 10                    // Wall follow constant for right motors
#define  DS1 40                    // Desired left motor speed - left turn
#define  KL1 80                    // Conversion constant for left motor-left turn
#define  DS2 15                    // Desired right motor speed - left turn
#define  KR2 594                   // Conversion constant for right motor-left turn
#define  RTURN 1800                // Number of counts for 90 degree turn
#define  UTURN 3200                // Number of counts for 180 degree turn
#define  DS_3 40                             // Desired distance for sensor 1
#define  DS_2 48                             // Desired distance for sensor 2
#define  DS_4 65                             // Desired distance for sensor 4
#define  DS_5 30                             // Desired distance for sensor 5

#define CAL_SUM 16                           // light level constant


volatile unsigned int lms, rms, lduty, rduty, select ,sensornum, int_count,
turn_count, r_count; volatile unsigned int u_count, eog, centerL_sensor;
volatile unsigned int  bit1=0, bit2=0, temp1=0, temp2=0, volt = 0, turn, turncount;
volatile unsigned int centerR_sensor, outterR_sensor, sensor_sum, outterL_sensor,;
volatile signed int temp, count, center_sensor;
volatile unsigned char s1=0,s2=0,s3=0,s4=0,s5=0,s6=0, s7=0, tone_start, cal_count,
tone_start; volatile unsigned char t1, t2, ts, s_count, line_startr, r_enter=0,
l_enter=0;
volatile unsigned char white_line, fire_out, white_count=0, line_detected=0,
line_startl,;
volatile unsigned char tone_start, t1, t2, ts, s_count, centerL_avg, centerR_avg,
outterL_avg;
volatile unsigned char outterR_avg, n, twozero;

void sensor_select();
void getdistance();
void turn_left();
void turn_right();
void stop_robot();
void delay(unsigned int ms);
void tone();
void fire_scan();
void end_of_game();

main()
{
```

```
    DDRT = 0x04;                                    // Bit 0,1 input, bit 2 output
    DDRP = 0xbf;                                    // Bit 0,1,4,5,7 output
    DDRS = 0xf0;                                    // CS, SS, SCLK, MOSI outputs
    count = 0;
    lms = 0;                                        // Zero counter for left wheel
    rms = 0;                                        // Zero counter for right wheel
    select = 0;
    white_line = 0;
    line_startl = 0;
    line_startr = 1;
    r_count = 0;
    u_count = 0;
    s_count = 0;
    n = 4;                                          //number of averages
    twozero = 0x01;

    PORTT = 0x00;
    PORTP = 0x88;                                   // Set PP0-2,4,5 low, PP3 & 7 high

    // Setup for the A/D on HC12
    ATDCTL2 = ATDCTL2 | 0x80;                       // Power up A/D Converter
    ATDCTL4 = 0X01;
    ATDCTL5 = 0x70; /* 0 1 1 1 0 0 0 0
                       | | | \_____/
                       | | |    |
                       | | |     \___  Channel Select
                       | | \_________   Mult = 1 => Multi-channel
                       |  \___________  Scan = 1 => continuous conversion
                        \_____________  S8CM => 8-bit mode */

    // Setup for serial peripheral interface (SPI)

    SP0CR1 = 0x50;                                  // MSB 1st, multi-bytes w/SS asserted
                                                    // Master mode, 0 phase & polarity
    SP0CR2 = 0x00;                                  // Normal mode (not bi-directional)
    SP0BR = 0x02;                                   // 1MHz SPI clock
    PORTS = PORTS | 0x80;                           // Deselect slave

     // setup for pulse width modulator for channel 1 & 0
    PWCLK = 0x00;             // Choose 8-bit mode
    PWCTL = 0x00;             // Choose left-aligned
    PWPOL = 0x0f;             // Polarity & select clock mode 0 for all Channels
    PWCLK = PWCLK | 0x18;     // Select N = 3 for ChannelS 1 & 0
    PWPER0 = PERIOD - 1;      // Select period for Channel 0-left wheel
    PWPER1 = PERIOD - 1;      // Select period for Channel 1-right wheel
    PWEN = PWEN | 0x03;       // Enable PWM on Channel 1 & 0


// set up real time interrupt

    RTIFLG = 0x80;
    RTICTL = 0x82;                          //RTIE, set interrupt rate 2ms
    enable();                               // Enable interrupts
    tone();
    PORTP = PORTP | 0x30;                   // Disengage brakes on motor

    while (TRUE)
```

```
    {

      sensor_select();                    // get values for sensors

    if (white_line == 1 & line_startl == 1 & line_startr == 1) //found whiteline
      {
            stop_robot();
      }

    else if (s4 >= DS_4)                          // stop and pivot right
      {
            turn_right();
      }

    else if (s3 == DS_3 | count ==0)              // Robot is going straight
      {
      lduty = (KL*DS)/100 + (KPL*(DS - lms))/10;      // Dutycycle for channel 0
      rduty = (KR*DS)/100 - (KPR*(DS - rms))/10;      // Dutycycle for channel 1

      PWDTY0 = ((lduty*PERIOD)/100)-1;
      PWDTY1 = ((rduty*PERIOD)/100)-1;

      count = 1;
      }

    else if (s3 <= 10 & count == 1)               // Turn left at detected opening
      {
            turn_left();
      }

    else if (s3 < DS_3 | s3 > DS_3)       // Adjust robot from veering left or right
{

      lduty = (KL*DS)/100 - (KWL*(DS_3 - s3))/10;     // Dutycycle for channel 0
      rduty = (KR*DS)/100 + (KWR*(DS_3 - s3))/10;     // Dutycycle for channel 1

      if (lduty > 90)               // Sets duty cycle if exceeds 200 or less than 0
                  {
                  lduty = 90;
                                        Page 2
                  }
      if (lduty < 5)
                  {
                  lduty = 5;
                  }
      if (rduty > 90)
                  {
                  rduty = 90;
                  }
      if (rduty < 5)
                  {
                  rduty = 5;
                  }


            PWDTY0 = ((lduty*PERIOD)/100)-1;
            PWDTY1 = ((rduty*PERIOD)/100)-1;
```

```
      }

    }
}

void turn_left()
{
lduty = (KL1*DS1)/100 + (KPL*(DS1 - lms))/10;         // Dutycycle for channel 0
rduty = (KR2*DS2)/100 - (KPR*(DS2 - rms))/10;         // Dutycycle for channel 1


PWDTY0 = ((lduty*PERIOD)/100)-1;
PWDTY1 = ((rduty*PERIOD)/100)-1;

l_enter++;                                            // keeps track of left turns
line_startl= 1;
}


void stop_robot()
{
      r_count = 0;


      PORTP = PORTP & ~0X30;                                  // sets brake PP4,PP5

      centerL_sensor = 0;
      centerR_sensor = 0;
      outterL_sensor = 0;
      outterR_sensor = 0;
      count = 0;

      delay(100);

      centerL_sensor = ADR7H;                                 // Reads fire sensor
      centerR_sensor = ADR6H;
      outterL_sensor = ADR5H;
      outterR_sensor = ADR4H;

      while (s_count < 2)
      {
            while (cal_count < (twozero << n))
            {
                  centerL_sensor = centerL_sensor + ADR7H;
                  centerR_sensor = centerR_sensor + ADR6H;
                  outterL_sensor = outterL_sensor + ADR5H;
                  outterR_sensor = outterR_sensor + ADR4H;


                  cal_count++;
            }

                  centerL_avg = centerL_sensor >> n;
                  centerR_avg = centerR_sensor >> n;
                  outterL_avg = outterL_sensor >> n;
                  outterR_avg = outterR_sensor >> n;
```

```
                cal_count = 0;
                centerL_sensor = 0;
                centerR_sensor = 0;
                outterL_sensor = 0;
                outterR_sensor = 0;

          s_count++;
          sensor_sum = (centerL_avg + centerR_avg + outterL_avg + outterR_avg);

          if (sensor_sum > CAL_SUM)
          {
                fire_scan();
          }

     }

     s_count = 0;
     PORTP = PORTP | 0X04;                    // rev PP2

     PORTP = PORTP | 0X30;                    // sets brake PP4,PP5
     PORTP = PORTP & ~0x80;                   // Reset Altera counter set PP7 low
     int_count++;                             // Increment counter

     PORTP = PORTP | 0x80;                    // Set PP7 high
     turn_count = 0;

     while (turn_count <= UTURN)
     {
     lduty = (KL*DS)/100;                          // Dutycycle for channel 0
     rduty = (KR*DS)/100;                          // Dutycycle for channel 1

     PWDTY0 = ((lduty*PERIOD)/100)-1;
     PWDTY1 = ((rduty*PERIOD)/100)-1;
     }

     PORTP = PORTP & ~0x04;                              // forward PP2


     lduty = 50;
     rduty = 50;


     PWDTY0 = ((lduty*PERIOD)/100)-1;
     PWDTY1 = ((rduty*PERIOD)/100)-1;

     line_startl = 0;
     line_startr = 0;
     u_count++;                                    //counts number of u-turns

}

void turn_right()
{
     PORTP = PORTP | 0X04;                               // rev PP2


     PORTP = PORTP & ~0x80;                   // Reset Altera counter set PP7 low
```

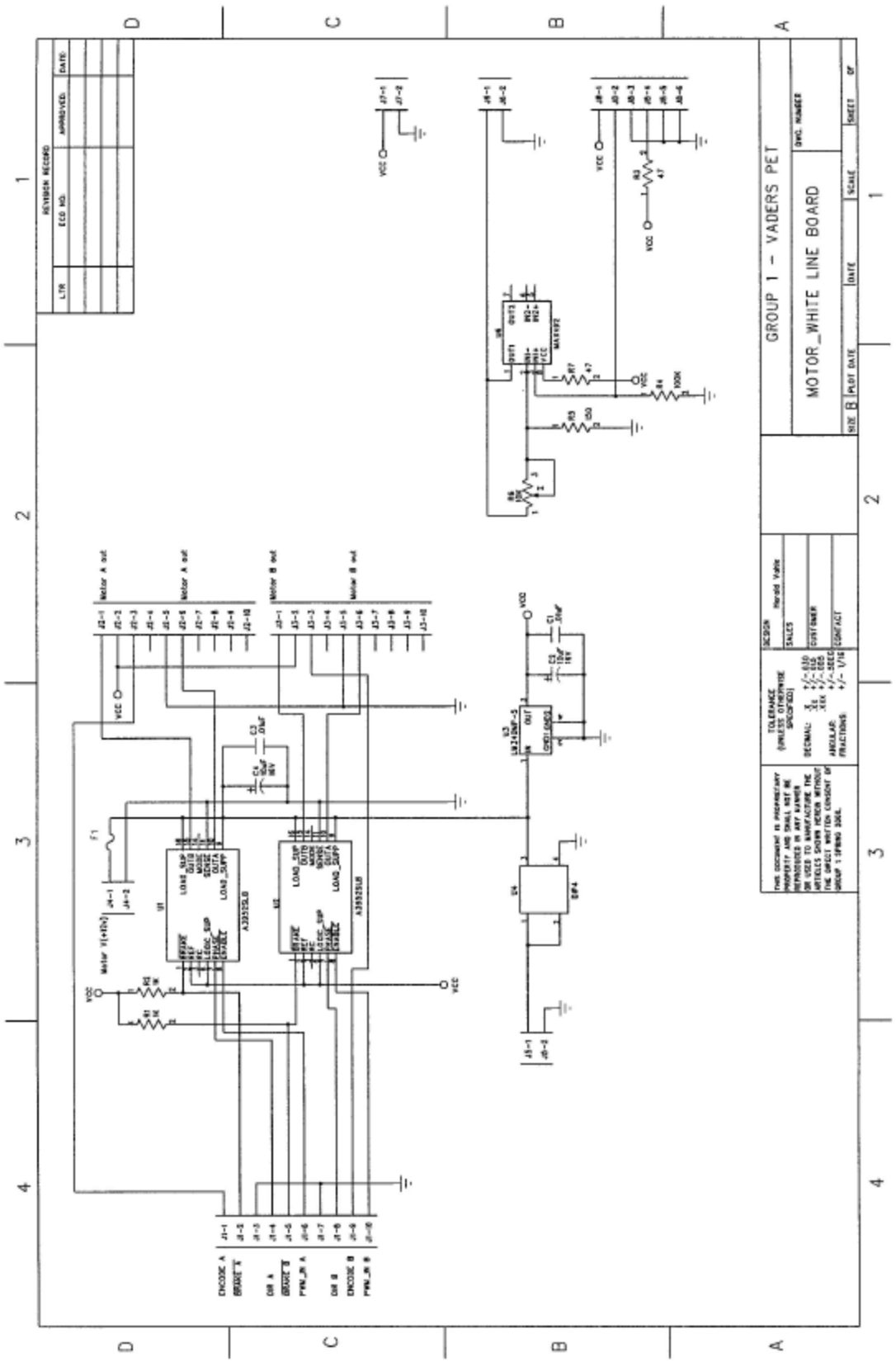
```
      int_count++;                              // Increment counter

      PORTP = PORTP | 0x80;                     // Set PP7 high
      turn_count = 0;
      line_startr = 1;

      while (turn_count <= RTURN)
      {
      lduty = (KL*DS)/100;                          // Dutycycle for channel 0
      rduty = (KR*DS)/100;                          // Dutycycle for channel 1

      PWDTY0 = ((lduty*PERIOD)/100)-1;
      PWDTY1 = ((rduty*PERIOD)/100)-1;
      }
      PORTP = PORTP & ~0x04;                              //forward PP2

}


void sensor_select()
{
      while (select <= 6)
        {


            if (select == 0)                    // select channel 1 of A/D
            {
                sensornum = 0xCF;               // getting measured value of sensor 2
            }
            if (select == 1)                    // select channel 3 of A/D
            {
                sensornum = 0xDF;               // getting measured value of sensor 4
            }
            if (select == 2)                    // select channel 5 of A/D
            {
                sensornum = 0xEF;               // getting measured value of sensor 6
            }
            if (select == 3)                    // select channel 0 of A/D
            {
                sensornum = 0x8F;               // getting measured value of sensor 1
            }
            if (select == 4)                    // select channel 2 of A/D
            {
                sensornum = 0x9F;               // getting measured value of sensor 3
            }
            if (select == 5)                    // select channel 4 of A/D
            {
                sensornum = 0xAF;               // getting measured value of sensor 5
            }
            if (select == 6)                    // select channel 6 of A/D
            {
                sensornum = 0xBF;               // getting measured value of sensor 7
            }
                getdistance(sensornum);
        }
           select = 0;
```

```
}

void getdistance()
{

   PORTS = PORTS & ~0x80;                    // Select slave
   SP0DR = sensornum;                      // Extclk, unipolar, 0,1,& 2 sel Start bit
   while ((SP0SR & 0x80) == 0);              // Wait for transmit complete
   SP0DR = 0;                                // Transmit byte of all zeros
   while ((SP0SR & 0x80) == 0);              // Wait for transmit complete
   bit1 = SP0DR;                             // receive byte one
   SP0DR = 0;                                // Transmit byte of all zeros
   while ((SP0SR & 0x80) == 0);              // Wait for transmit complete
   bit2 = SP0DR;                             // Receive byte two
   temp1 = bit1 << 2;                        // Bit shift left 2
   temp2 = bit2 >> 6;                        // Bit shift right 6
   volt = (temp1 | temp2);                   // Ored for sensor input
   PORTS = PORTS | 0x80;                     // Deselect slave

   if (select == 0)
      s2 = volt;                             // Measured distance from sensor 2

   if (select == 1)
      s4 = volt;                             // Measured distance from sensor 4

   if (select == 2)
      s6 = volt;                             // Measured distance from sensor 6

   if (select == 3)
      s1 = volt;                             // Measured distance from sensor 1

   if (select == 4)
      s3 = volt;                             // Measured distance from sensor 3

   if (select == 5)
      s5 = volt;                             // Measured distance from sensor 5

   if (select == 6)
      s7 = volt;                             // Measured distance from sensor 7

      select++;                              // select next channel and sensor

}

void tone(void)
{

      delay(25);
      tone_start = 1;
      ts = 0;

      while (ts == 0)
      {
      tone_start = PORTP & 0x40;             // Look at bit PP6 for tone input
      t1= tone_start;
      delay(5);
      tone_start = PORTP & 0x40;             // Look at bit PP6 for tone input
```

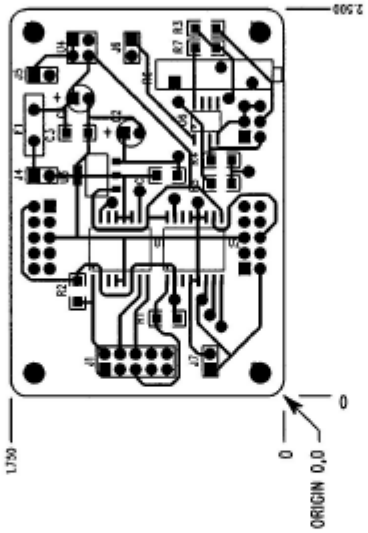
```
      t2 = tone_start;
      delay(200);
      if (t1 ==0 && t2 == 0)                                  // Tone detected
      {
            ts = 1;
      }
    }
}
void delay(unsigned int ms)                                   // set for delay of 1ms
{
      int i;
      while (ms > 0)
      {
          i = D_1MS;
          while (i >0)
          {
            i = i - 1;
          }
            ms = ms - 1;
      }
}

@interrupt void rtif_isr(void)
{
      white_line = PORTT & 0x01;                              // Looks for white line
high

      int_count = 0;                                          // Zero interrupt count
      lms = *(unsigned int *) 0x0404;                         // measured Left wheel
      rms = *(unsigned int *) 0x0406;                         // measured Right wheel
      turn_count = turn_count + lms;                          // count for U-turn and
right turn
      PORTP = PORTP & ~0x80;                                  // Reset Altera counter
set PP7 low
      int_count++;                                            // Increment counter

      PORTP = PORTP | 0x80;                                   // Set PP7 high
      RTIFLG=0x80;                                            // Clear source of
interrupt

}


void fire_scan()
{
      PORTP = PORTP | 0X30;

      fire_out = 0;

      while (fire_out == 0)
      {

            centerR_sensor = ADR6H;                    // Read center right sensor
            centerL_sensor = ADR7H;                    // Read center left sensor

            lduty = (KL*DS)/100 - (KWL*(centerL_sensor - centerR_sensor))/10;
            rduty = (KR*DS)/100 + (KWR*(centerL_sensor - centerR_sensor))/10;
```

```
            PWDTY0 = ((lduty*PERIOD)/100)-1;
            PWDTY1 = ((rduty*PERIOD)/100)-1;



            if (white_line == 1 )
                  {
                  PORTP = PORTP & ~0x30;                    // Sets brake on motor
                  PORTT = PORTT | 0x04;                     // Turn on Fan
                  delay(2000);
                  PORTT = PORTT & ~0x04;                    // Turn off Fan
                  fire_out = 1;

                  }
      }
      end_of_game();

}
void end_of_game()
      {
      eog = 1;
      while (eog = 1)
       {
            PORTP = PORTP & ~0x30;                          // Sets brake on motor
      }
}
```

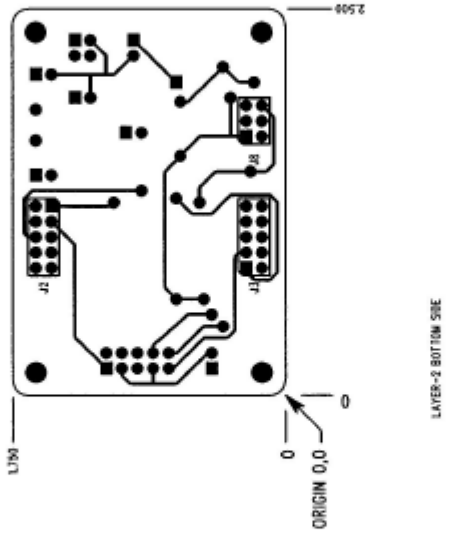
Appendix 2



Appendix 2

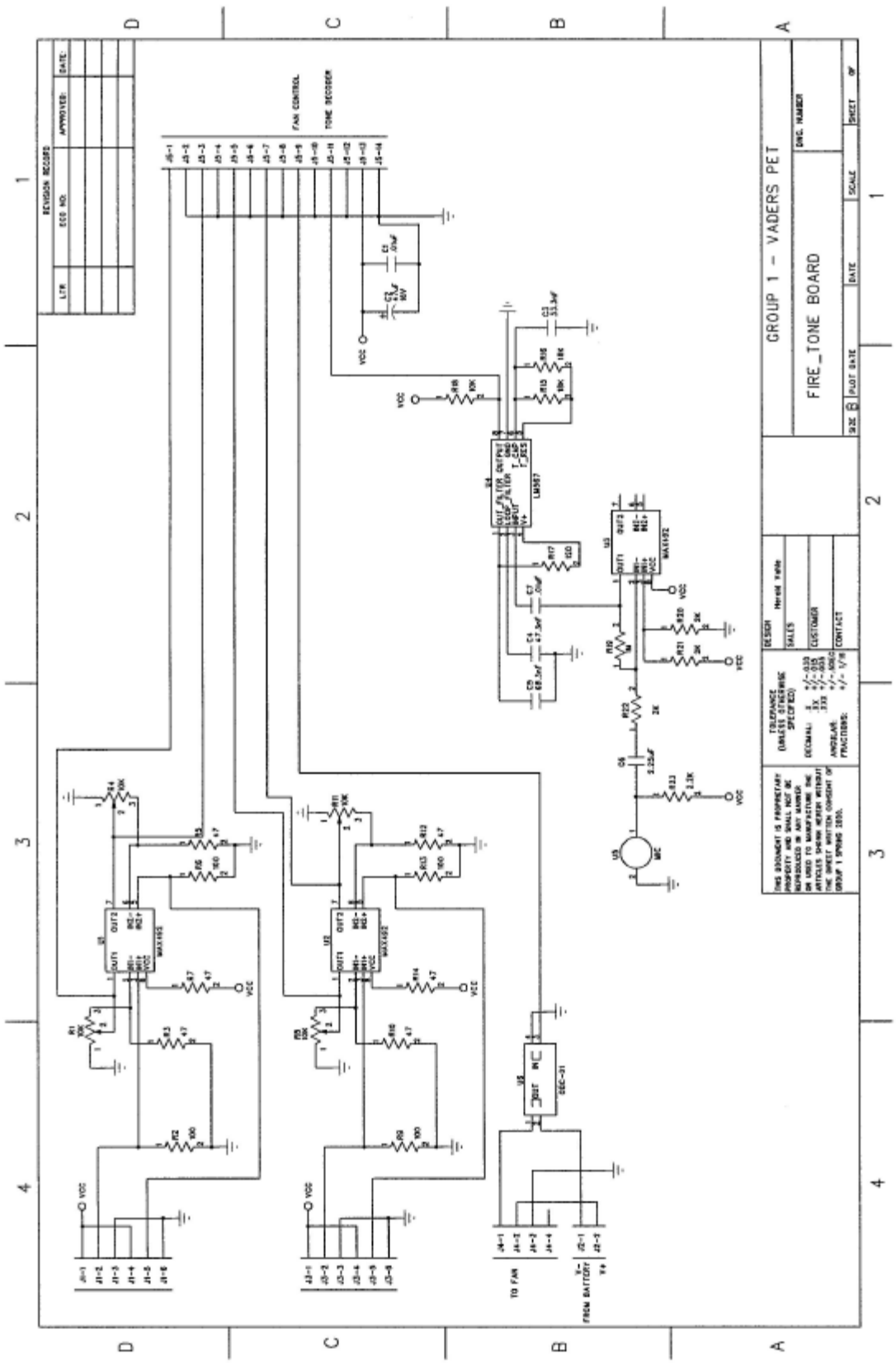

2.500
1.750
0
0
ORIGIN 0,0
LAYER-1 TOP SIDE
Motor_whiteline.pcb - Tue May 09 10:16:12 2000

Appendix 2

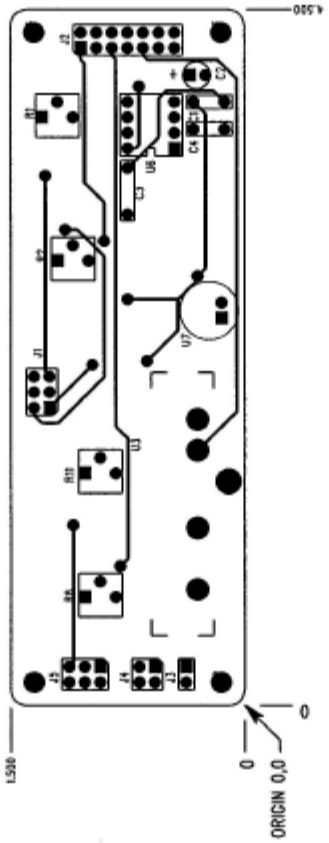


Motor_whiteline.pcb - Tue May 09 10:16:49 2000

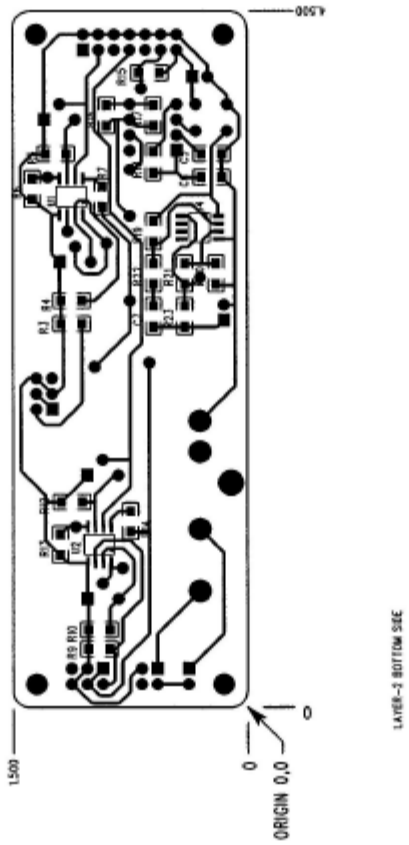
Appendix 2


GROUP 1 - VADERS PET
FIRE_TONE BOARD
FAN CONTROL
TONE DECODER
TO FAN
FROM BATTERY

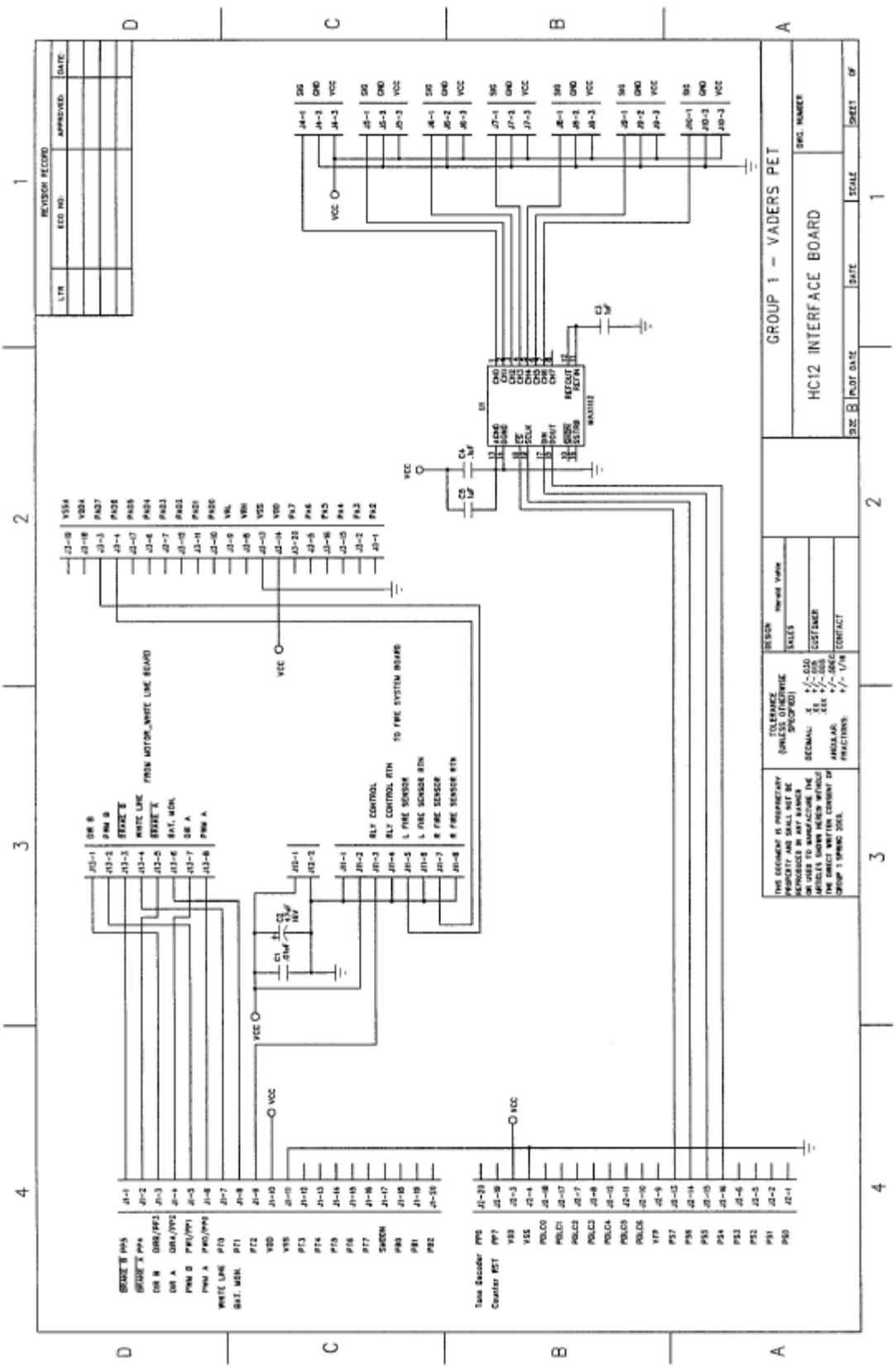
Appendix 2


4.500
5.500
0
0
ORIGIN 0,0
LAYER-1 TOP SIDE
Fire_tone.pcb - Tue May 09 10:13:58 2000

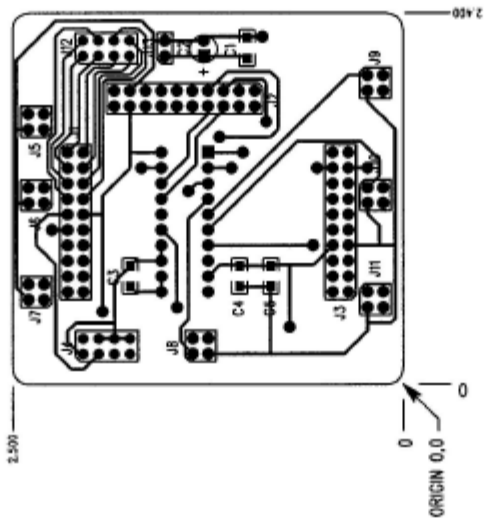
Appendix 2



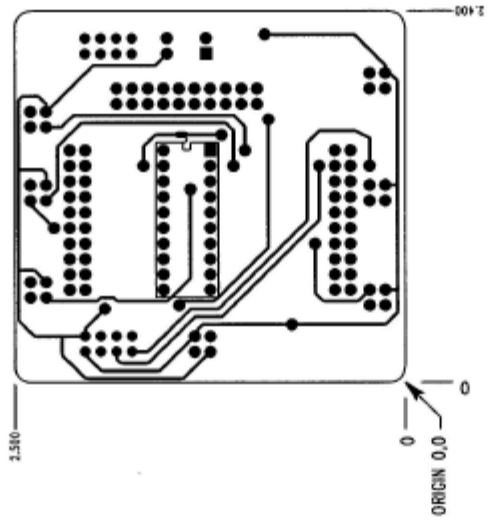
Appendix 2


GROUP 1 – VADERS PET
HC12 INTERFACE BOARD
REVISION RECORD
LTR
ECO NO
APPROVED
DATE
SIZE B
PLOT DATE
DATE
SCALE
SHEET
OF
DWG. NUMBER
DESIGN
Hamid Vahe
SALES
CUSTOMER
CONTACT
TOLERANCE (UNLESS OTHERWISE SPECIFIED)
DECIMALS:
ANGULAR:
FRACTIONS
THIS DOCUMENT IS PROPRIETARY PRODUCTS AND SHALL NOT BE REPRODUCED OR TRANSFERRED TO OTHERS OR USED TO MANUFACTURE THE ARTICLES SHOWN HEREIN WITHOUT THE DIRECT WRITTEN CONSENT OF GROUP 1 SPRING 2005.
VCC
GND
SIG
MOTOR_WHITE LINE BOARD
TO FIRE SYSTEM BOARD
RLY CONTROL
L FIRE SENSOR
R FIRE SENSOR
WHITE LINE
BAT. MON.
DIR A
PWM A
DIR B
PWM B
ENCODER A
ENCODER B
Tank Encoder
Counter RST
SHDN
RESET
XTAL
EXTAL

Appendix 2

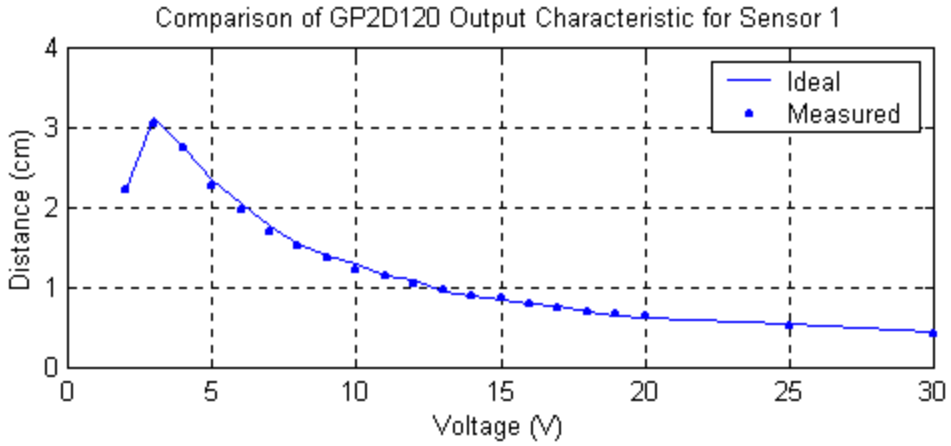

2.500
2.500
0
0
ORIGIN 0,0
LAYER-1 TOP SIDE
Hc12IF.pcb - Tue May 09 10:17:26 2000

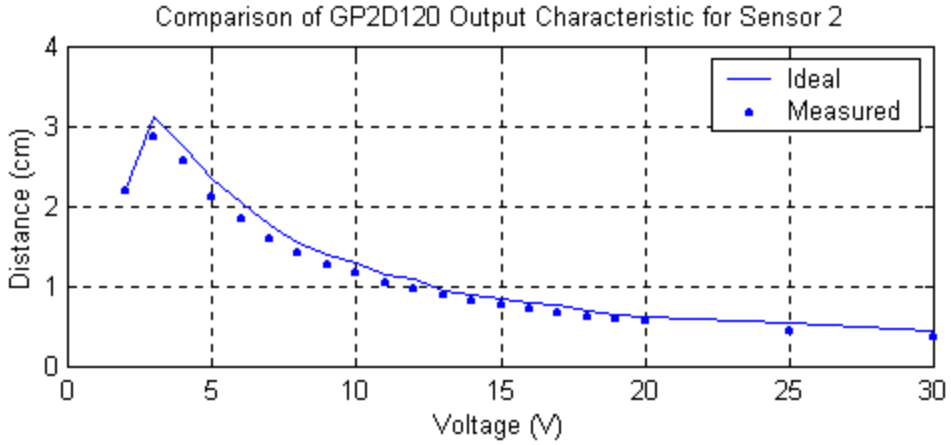
Appendix 2



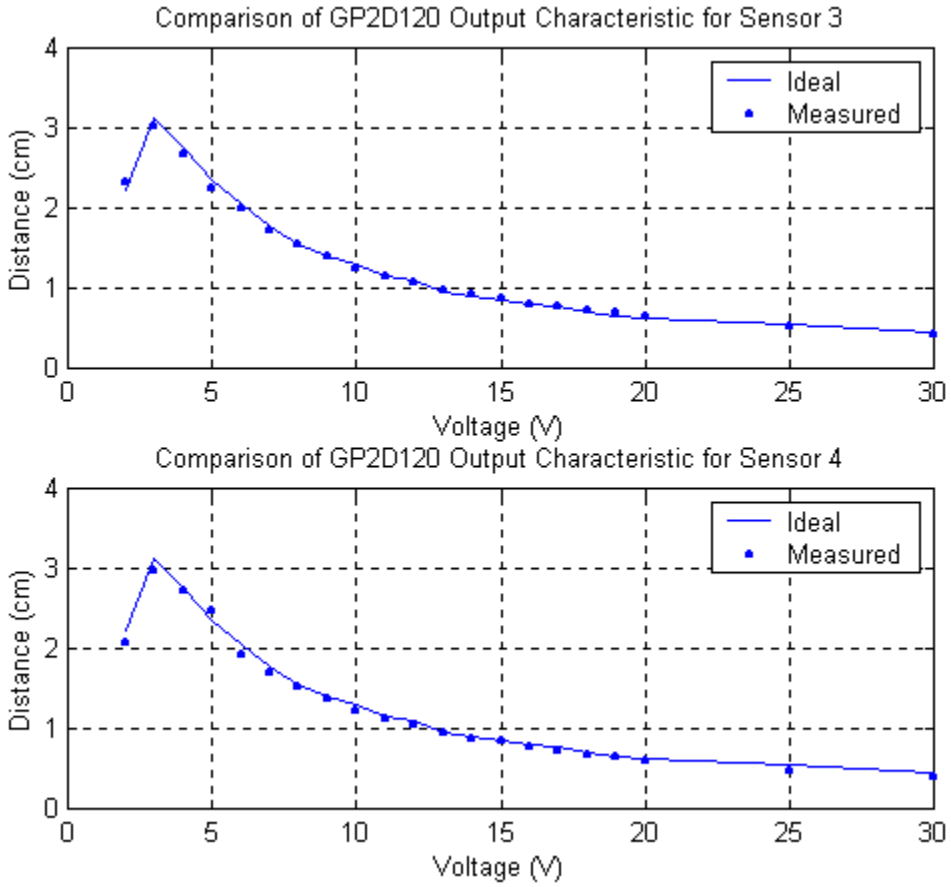
Hc12IF.pcb - Tue May 09 10:17:44 2000

Appendix 3

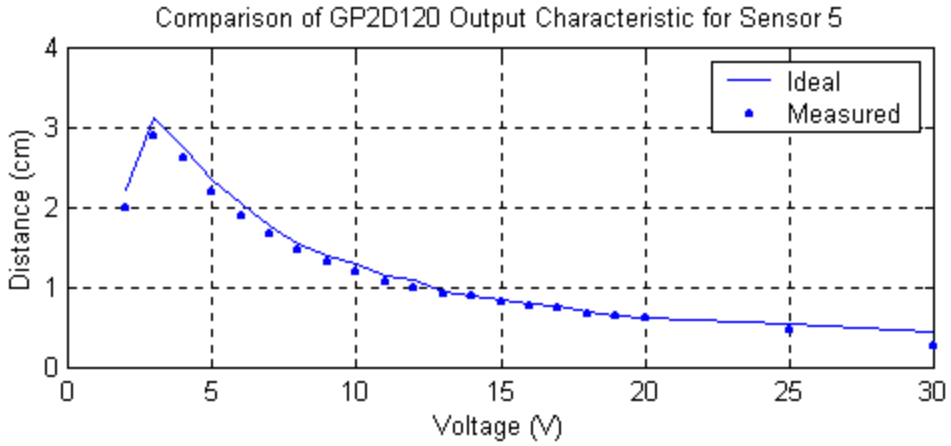

Comparison of GP2D120 Output Characteristic for Sensor 1
Ideal
Measured
Distance (cm)
Voltage (V)
0
1
2
3
4
0
5
10
15
20
25
30

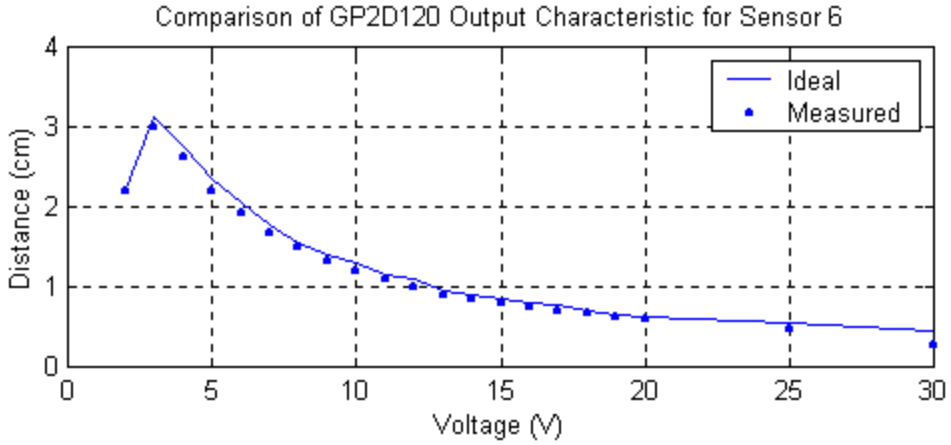

Comparison of GP2D120 Output Characteristic for Sensor 2
Ideal
Measured
Distance (cm)
Voltage (V)
0
1
2
3
4
0
5
10
15
20
25
30

Appendix 3

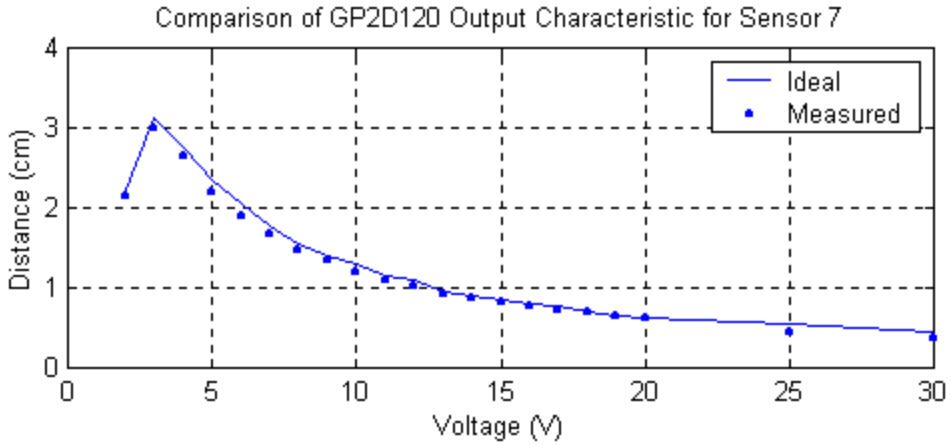

Comparison of GP2D120 Output Characteristic for Sensor 3
Ideal
Measured
Distance (cm)
Voltage (V)
0
1
2
3
4
0
5
10
15
20
25
30
Comparison of GP2D120 Output Characteristic for Sensor 4
Ideal
Measured
Distance (cm)
Voltage (V)

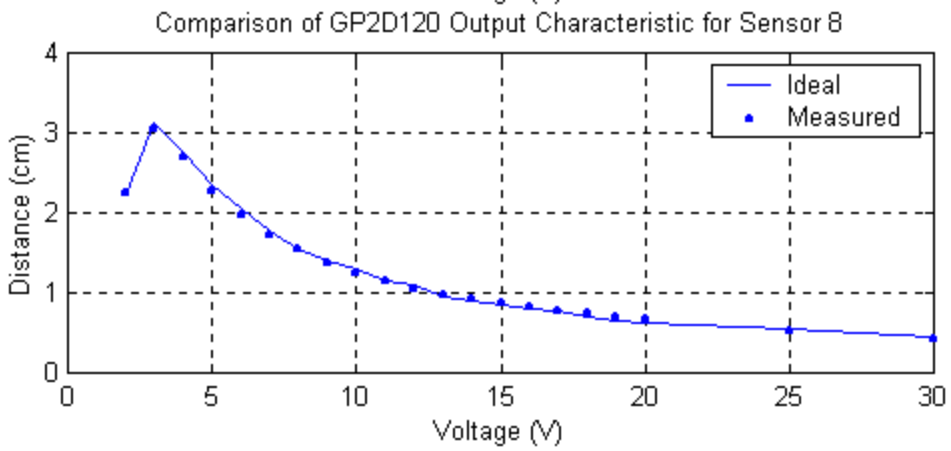
Appendix 3


Comparison of GP2D120 Output Characteristic for Sensor 5
Ideal
Measured
Distance (cm)
Voltage (V)
0
1
2
3
4
0
5
10
15
20
25
30


Comparison of GP2D120 Output Characteristic for Sensor 6
Ideal
Measured
Distance (cm)
Voltage (V)
0
1
2
3
4
0
5
10
15
20
25
30

Appendix 3


Comparison of GP2D120 Output Characteristic for Sensor 7
Ideal
Measured
Distance (cm)
Voltage (V)
0
1
2
3
4
0
5
10
15
20
25
30


Comparison of GP2D120 Output Characteristic for Sensor 8
Ideal
Measured
Distance (cm)
Voltage (V)
0
1
2
3
4
0
5
10
15
20
25
30

Appendix 3

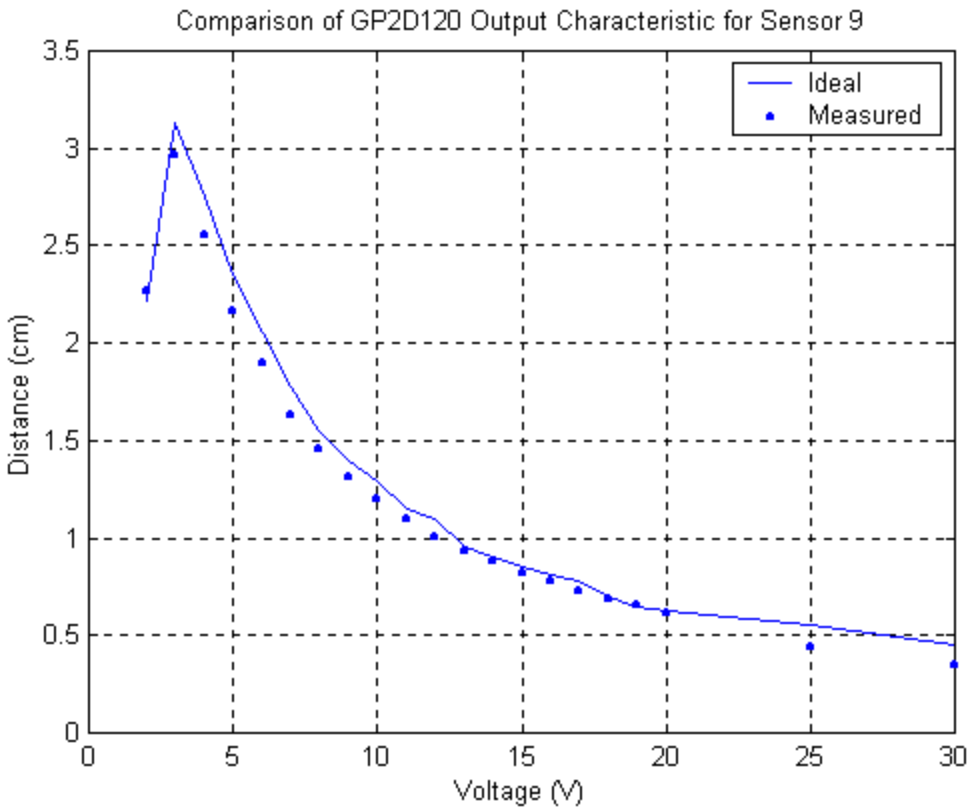

Comparison of GP2D120 Output Characteristic for Sensor 9
Ideal
Measured
Distance (cm)
Voltage (V)
0
0.5
1
1.5
2
2.5
3
3.5
0
5
10
15
20
25
30

Fig 9. Spectral power distribution of a HPS lamp at various burning times. Curve 1 was the distribution measured at 486 hours of operation, 2 at 906 hours, and 3 at 1165 hours.