Pokey the Fire-Fighting Robot

A Logical Design Using Digital and Analog Circuitry

Submitted by Group 7: Gerald Weed

Michael Schumacher

Shawn McVay

Jack Landes

Submitted on: May 11th, 1999

ABSTRACT

"Pokey the Fire-Fighting Robot"

By: Gerald Weed, Michael Schumacher,
Shawn McVay, and Jack Landes

Electrical engineering majors at New Mexico Tech are required to take a course in introductory design that introduces students to the idea of building a free-moving robotic system. This system must be completely autonomous, navigate a maze without touching the walls, and extinguish a candle that is located in one of four "rooms". Another restraint on the system is that it must fit within a 12.25"x12.25"x12.25" cube.

This report describes one method used to accomplish the above task. Using a Motorola 68HC11 microcontroller, a variety of sensors, and software programming written in the C language, the robot will move around the maze, speeding up and slowing down so as to not touch the walls. Contained within this report are the details of the method with special attention given to subsystem design and integration. Special focus should be given to the motor control, fire suppression, and modified differential drive subsystems, as they were unique designs within the class.

Keywords: motors, LMD18200 H-bridges, pulse-width modulation (PWM), encoders, decoders, Altera, programmable logic device (PLD), duty cycle, sensors, GP2D12, optical, infrared, OPT101, Motorola 68HC11, EPM7128ALC84-12, monopulse radar.

Table of Contents

Table of Figures

1 Introduction

The overall objective of the robot is to be able to navigate a robot through a standardized maze, detect a candle flame, and extinguish it. To this end, there are four different subsystems that enable the robot to view its environs and aid it in maneuvering the maze to accomplish its task. In order to discover where the robot is relative to its environs, the wall following subsystem has been implemented. To aid the robot, there are white lines in the maze that signify doorways, the starting and finishing circle for the robot, and an area of radius one foot that is centered on the candle. In order to make the most use of these lines, there is a line detection subsystem to aid the robot in traversing the maze. A motor control system must be implemented to maneuver the robot through the maze based on its interpretation of its position. Distance from walls, white lines, and the distance to the fire determine the direction and speed that the wheels must turn. Our innovative altered differential drive system also adds more stability to our design. The center of gravity of the robot and the three points of contact with the floor imply that it is inherently difficult to make the robot turn over. With this robust design, our robot can move around the maze very efficiently. Finally, the overall task of the robot is to find the candle and extinguish it. The robot uses two types of systems in order to detect the flame. First, for long-distance flame detection, there is a fire ranging sensor. While this system is very effective for this application, it has a narrow beam width and may be too unreliable to use for short- and medium-range flame detection. In this case, we have implemented another flame sensor that is based on a radar system known as monopulse radar. These four systems guide the robot in its mission to be a successful firefighter.

2 Theory of Operation

2.1 SENSOR SYSTEMS

Sensors are the eyes and ears of the robot. Without sensors, as with virtually every other system, the robot is useless. If no sensors are on the robot, the robot blindly follows its own will, often ending up in a heap of parts alongside a wall, at the bottom of a large pit, or well. Of course, there is a vested interest in keeping the robot intact and able to do its job, so it is very worthwhile to have a fully functional sensor system.

All of the systems presented here have their strengths and weaknesses. The advantages and disadvantages of the individual parts of each subsystem will be explained throughout this section. While the electrical and functional aspects of the sensor system will be discussed in detail, the practical aspects of circuit board construction will be briefly analyzed near the end of this section.

2.1.1 Wall Following Subsystem

The land area of Australia is about 7.7 million square kilometers, which is almost the same as the land area of the United States. (http://www.aiaa.org/calendar/iaf98cfp.html) We throw in this fact not because it is an interesting tidbit, but to make a point. While there is a lot of land on this earth, it does not make much difference if one's location is unknown. This is as important for a robot as it is for a human. While this particular robot is not going to be roaming across the Australian Outback, it will be navigating a standardized 98-inch by 98-inch maze. The maze is set up with walls partitioning it into rooms and halls. Since the robot cannot use dead reckoning (a method of navigation by which the robot has a preset map in its memory), we must have a subsystem to navigate the robot through the maze.

Figure 2.1.1 below shows a schematic of one unit of the wall following subsystem. Note for the wall following subsystem, there are *three identical units*, which compose the entire subsystem. We include the schematic of one sensor unit to minimize confusion.

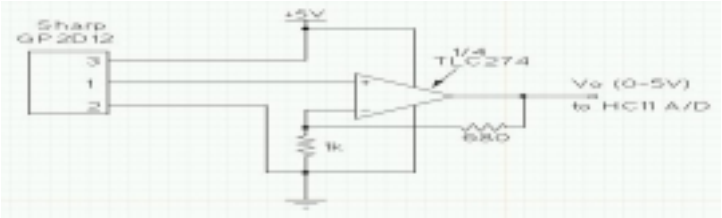


Figure 2.1.1: Wall Following Subsystem Unit Schematic for One Sensor

A breakdown of the wall following unit follows. The primary component of the wall following subsystem is the Sharp GP2D12 distance-measuring sensor. This sensor uses triangulation in order to measure the distance to a specific target, usually in the 10- to 80-cm range. This sensor has two unique advantages. First, it is non-specific to any type of wall coloring, provided it is not too reflective. Second, the output of the sensor is an analog voltage that typically ranges from 1 volt at 80 cm to 3 volts at 10 cm. While a two-volt variation over a 70-centimeter range is acceptable for the robot, the HC11 has an analog-to-digital voltage converter that can only accept voltages in the range of zero to five volts. In order to maximize the A/D converter capabilities, a linear non-inverting amplifier of gain 1.6 was added between the HC11 and the Sharp sensor in order to set the range of the wall measuring sensor for a 0-5 volt range. The linear amplifier uses a Texas Instruments TLC274 14-pin quad single supply op-amp, to amplify the signal. This operational amplifier is used for three reasons. The most important reason is that these op-amps use a single voltage supply instead of a dual voltage supply as with standard op-amps. These reduce the overall complexity of the robot by removing the circuitry necessary for a negative voltage supply. The second reason is that the

wall following subsystem uses only one op-amp for all three sensors, reducing circuit complexity, parts count, and power consumption. Additionally, we were able to obtain these operational amplifiers as samples, reducing the overall cost of the robot. Regardless, the output of the op-amp is fed into the HC11 analog-to-digital converter, where an 8-bit value is used by the HC11 to determine where the robot is relative to that wall.

The placement of the sensors will be discussed in the chassis design section of this paper.

2.1.2 Line Detection Subsystem

The rules for the competition contain a few regulations concerning the placement of white lines in the maze. White lines are placed where there is a doorway into a room and to signify an arc of radius one-foot, in the center of which the candle lies. A circle approximately one-foot in diameter is placed in the maze to act as a starting and stopping point for the robot. There are several good reasons why these lines and circles are in the maze. The doorway lines are important as it allows for algorithms to scan the room for a fire without actually entering it, saving precious time. The one-foot radius arc is important as part of the robot must be within it to legally extinguish the fire. Finally, the one-foot diameter “home-circle” is important as an algorithm can be constructed in order to make the robot return to this circle. With all of the information these white lines and circles convey, it is extremely worthwhile to have some method of detecting these events. Hence, the robot contains a line detection subsystem.

Figure 2.1.2 below shows the schematic for one sensor for the line detection subsystem. The robot currently carries *two identical sensor implementations*.

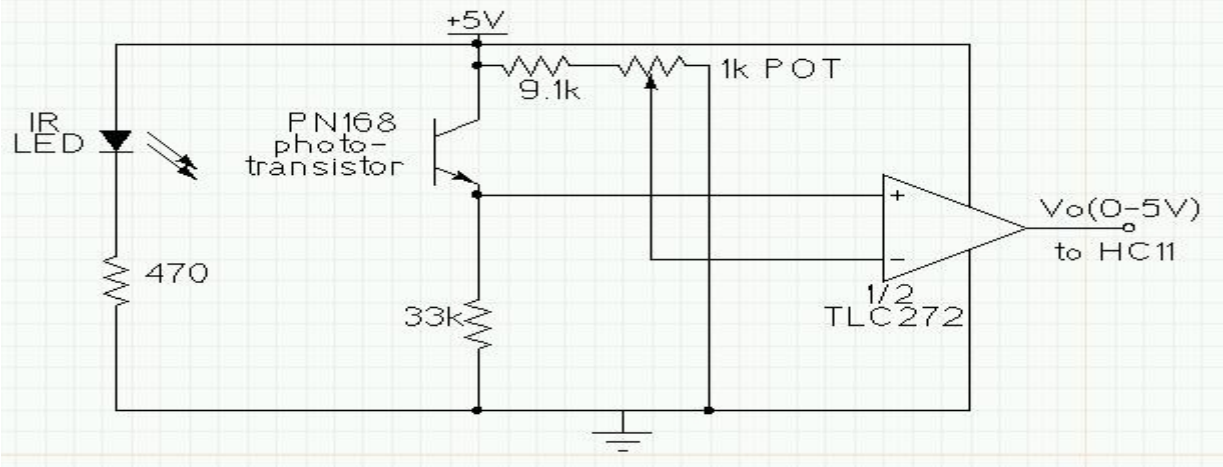


Figure 2.1.2: Schematic for Line Detection Sensor

This system is slightly more complex than the wall following subsystem. For reliability reasons, we decided to construct an infrared emitter/detector pair. Having an active line detection sensor is important because we did not want to be dependent on ambient light for detecting the white line. The emitter we used is a QT Optoelectronics CQX-15 IR emitter. We used this due to its high output and its capacity to be connected to +12 volts if the line sensors required more power. A current limiting resistor was placed between the cathode and ground to avoid shorting out the emitter. The detector is made of a PN168 phototransistor that uses a piece of floppy disk material as an optical filter. The emitter voltage of the PN168 is fed into the positive terminal of a comparator. This comparator is built using a TI TLC272 op-amp, which has the same characteristics of the TLC274 except that the TLC272 is a dual op-amp. The negative terminal of the comparator is connected to an adjustable voltage source that can be varied between 0 and 5 volts. This reference voltage is made adjustable so one can adjust the comparison for varying light conditions in the maze. The circuit works as follows. The detector's emitter voltage is low when it sees a black floor and high when it sees a white line. If the detector senses a black floor, the voltage reference is higher than the emitter voltage, so the op-amp outputs a logic low voltage. However, if a white line is detected, the emitter

voltage is higher than the adjustable voltage reference, so the comparator outputs a logic high voltage. The output of the comparator is fed into a port on the HC11 to trigger an interrupt when a rising edge is detected.

This circuit works nicely except for two small problems. The first problem with this circuit is that it is extremely difficult to adjust. The circuit as it stands has a problem of either railing out at 0 volts or near 5 volts if one does not take several minutes adjusting the voltage reference to changing maze conditions. Second, the output may be a bit unstable, especially if it detects a large amount of white particles on the floor. White particles could result from some other contestant overlubricating his tires or treads and leaving a nice glossy reflective coating along the maze floor. Adding Schmitt trigger inverters between the output of the comparator and the HC11 would go a long way toward resolving this problem; however, this was not implemented due to time restrictions.

2.1.3 Fire Sensors

The overall goal of this robot is to find and extinguish a candle somewhere within a maze. We implemented two different fire sensors for the robot. The first is a fire ranging sensor that is more suited to detect the candle at long distances. The second sensor is based on a well-known implementation known as monopulse radar. This sensor arrangement has some unusual capabilities that will be discussed in this section.

2.1.3.1 Fire Ranging Sensor

The maximum diagonal length of any one room in the maze is approximately 1.5 meters. In order to save time while searching for the candle, it is desirable to have the robot

enter the room, stop just inside, and perform a scan of it. Thus, a long distance sensor for detecting the flame is very useful.

Figure 2.1.3 below shows the final schematic for detecting the flame at long distances.

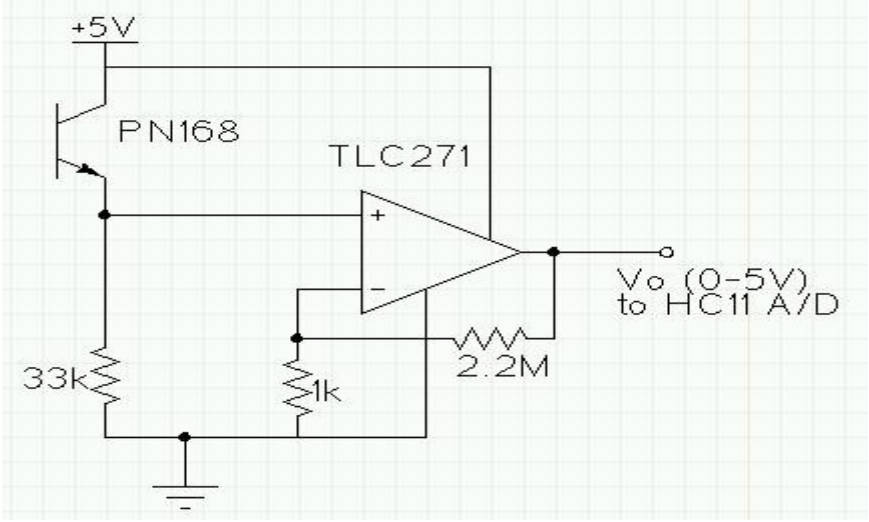


Figure 2.1.3: Schematic of a Long Distance Fire Sensor

To detect the emissions from the candle, we used a PN168 phototransistor that had an optical filter made from part of a floppy disk. The material used to construct a floppy disk will suppress higher wavelength visual light while allowing red, and infrared, light to pass through. The phototransistor is set up in a fashion similar to the one used with the line sensors. The emitter voltage is wired to a non-inverting amplifier of gain 2200. The high gain is necessary to be able to discern a flame at 1.5 meters within a room. The op-amp used in the amplifier is a TI TLC271, which has the same single-supply capabilities as a TLC274 except it accepts only one pair of inputs. The output of the amplifier is passed into the HC11 A/D converter for processing.

This circuit works very well at distances over 1 meter, but it has some problems. First, a flame at 1.5 meters is barely discernable compared to no flame in the room. With an

amplification of 2200, it is very hard to set levels in the HC11 to determine if there is a flame in the room. The beam width of the phototransistor is relatively narrow, so it might be possible to not detect a flame if a thorough scan is not done. In addition, a one-stage amplifier for such a high gain may not be desirable. Instead it might be worthwhile to use two stages of lesser gains and cascade them in order to achieve the required gain should one decide to use this circuit.

2.1.3.2 Fire Bearing Subsystem

The second method of fire detection is known as the fire bearing subsystem. We decided to implement two systems in order to add redundancy and reliability to the robot. This system is present to detecting the flame at medium and short distances. Also, this system is based on a radar implementation known as monopulse radar.

A brief discussion on the basics of monopulse radar is necessary to understand the reasons behind implementing such a system. Consider the following system, composed of a directional emitter or detector, and constructed to have an optimal radiation pattern shown in Figure 2.1.4.

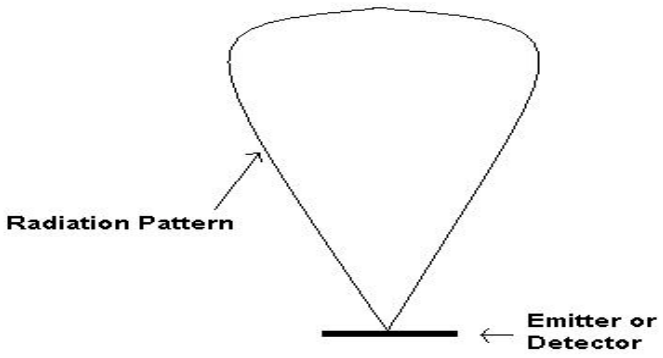


Figure 2.1.4: Ideal Radiation Pattern for an Emitter or Detector

A monopulse system is constructed using two emitters or detectors separated by a very small distance, as shown in Figure 2.1.5 below. In fact, the separation can be so small that the overall radiation patters might overlap each other. This can be very useful.

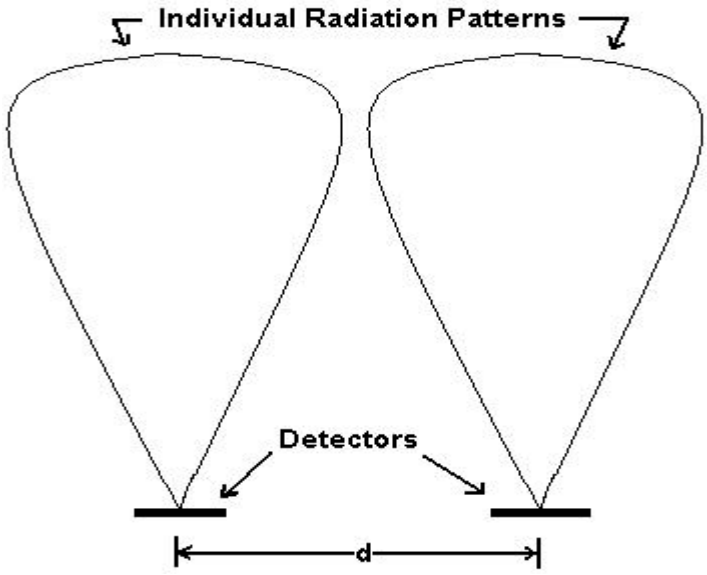


Figure 2.1.5: Monopulse System Setup Diagram

For the fire bearing subsystem, we are using two detectors to sense the fire and using the HC11 to compute where the fire is. There are two feasible ways of combining the signals. If the individual radiation patterns shown in Figure 2.1.5 are added, the resulting radiation pattern is a huge lobe (shown in Figure 2.1.6). This pattern is essentially a one-lobe system with a flatter radiation pattern than a single detector. It has the advantage of being able to find the flame as with the flame detection system described earlier, so this system does add redundancy in that respect.

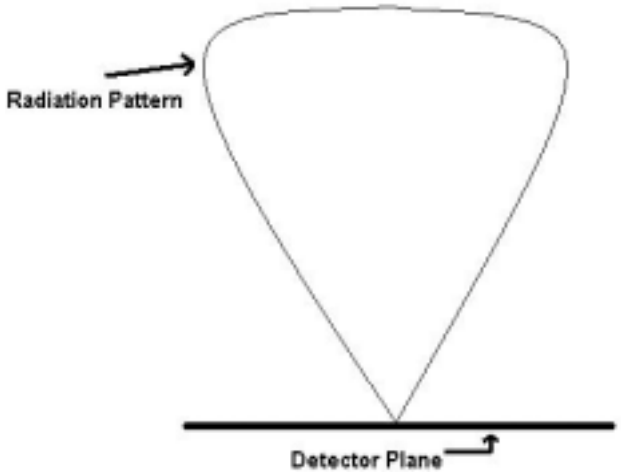


Figure 2.1.6: Monopulse Radiation Pattern from Summation of Detector Patterns

What happens when the two signals coming in from the detectors are *subtracted*? A very unique radiation pattern is realized (shown in Figure 2.1.7 below).

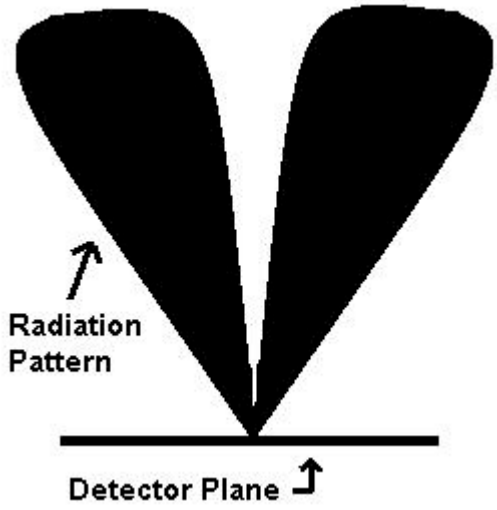


Figure 2.1.7: Monopulse Radiation Pattern from Subtraction of Detector Patterns

A very deep null is formed in the radiation pattern as a result. If the candle is directly ahead of the robot, the candle will be in the null and the robot moves straight ahead. But what happens if the candle is to the right of the robot? In this case, the intensity for the right detector will register higher than the one on the left. Thus, the robot will know that the candle is on the

right, and will move appropriately. The same argument holds true if the candle is to the left of the robot.

This is a superior way of detecting the candle for several reasons. Since the null is so sharp in the difference pattern, the robot can easily follow it. Second, this configuration is very easy to program in the HC11, as it requires only a subtraction and a few comparison statements. Most importantly, it beats the standard method of detection known as "sidewinding" by requiring less time to find the candle and guaranteeing a greater accuracy. Essentially, "sidewinding" is a method using only one sensor where one scans the room for fire and records the greatest signal strength in volts. The system then attempts to match the recorded highest voltage by scanning the room again in order to get a lock on the flame.

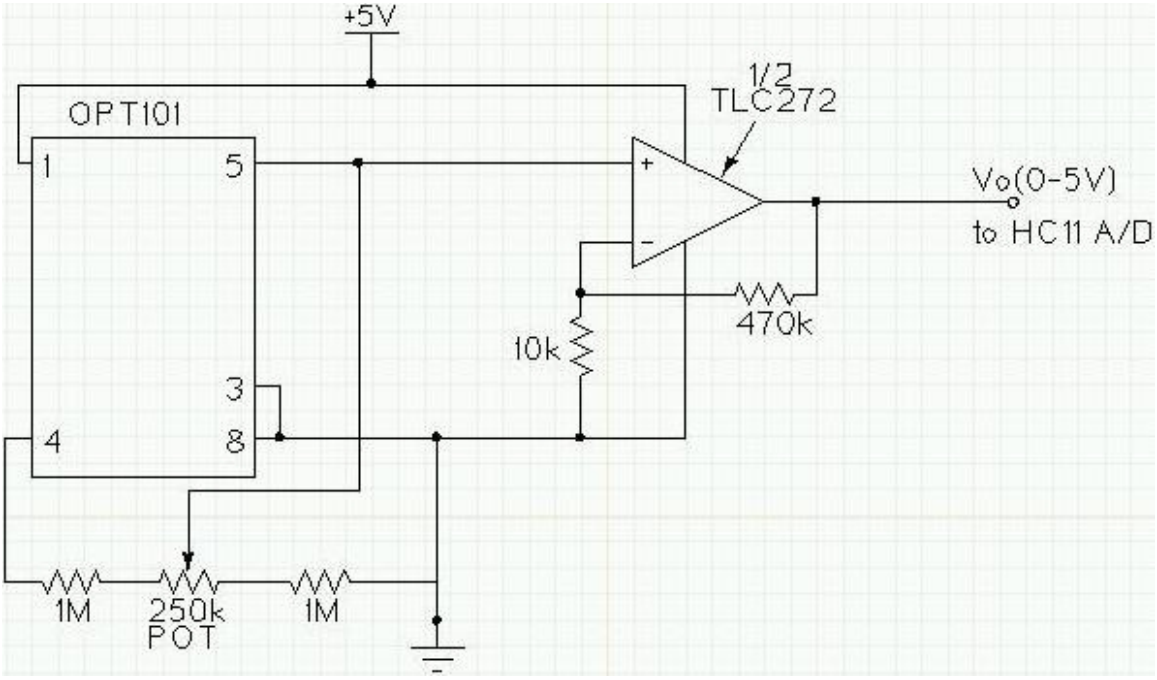


Figure 2.1.8: Schematic for One-half of Fire Bearing Subsystem

So how is the monopulse system constructed? The schematic for half of the monopulse system is shown in Figure 2.1.8 above. The entire system is composed of *two identical implementations.*

The central component of the fire bearing system is the Burr-Brown OPT101 photodetector and transconductance amplifier. We were fortunate enough to obtain samples of these photodetectors from Burr-Brown Corporation in order to construct this system. The OPT101 has a built-in photodetector that outputs a current proportional to light intensity. The OPT101 also includes a transconductance amplifier that converts the output current to a voltage. The implementation requires that the output of the OPT101 (pin 5) be connected to the 1-megohm feedback transconductance amplifier (pin 4) to obtain an output voltage. Since the sensors are not matched, we included a resistor-potentiometer adjustment system (shown above) in order to tune the output and to add more gain to the transconductance amplifier. The output of the OPT101 is fed into a non-inverting amplifier with a gain of about 47 to increase the output range of the sensors. The non-inverting amplifier is constructed using a TI TLC272 dual single-supply op-amp similar to the one used in the line detection subsystem. The output of the non-inverting amplifier is then fed into the HC11 A/D converter for processing into an 8-bit number. This 8-bit value is then used in the HC11 fire detection algorithms so the radiation patterns shown in Figures 2.1.6 and 2.1.7 are realized.

The fire bearing system works as implemented. However, there are some problems concerning this system as it is constructed. The physical construction of the system was very difficult, as it required shielding from all light except for a small slit for each sensor. Even then, there were optical filters made from a floppy disk in order to cut down on the visible light that the OPT101 can detect. The entire assembly was housed inside a Tic-Tac ™ box, as

this was the most discreet, efficient, and convenient method of containing the circuit board. Static-sensitive foam was added inside of the housing to cushion the sensors against vibrational shock. Even with all of these precautions, it is possible that light reflecting off of the *inside* of the housing could be detected by the OPT101, thereby modifying the detector voltages. Also, the adjustment assembly does not quite work as it is supposed to, as it is likely the Thevenin equivalent of the assembly, including the 1-megohm feedback transconductance amplifier, was modifying the output signal of the OPT101. For better adjustment, one could modify the adjustment scheme to mirror that of the voltage reference for the line detector amplifier in Figure 2.1.2 above.

From an electrical design standpoint, all sensor systems work as expected and to the specifications that each individual part was designed for. There are two real-world problems that the sensor system may encounter when the integration of the robot is completed. There could be a problem with noise occurring in the system while the robot is moving. The physical layout of the system on the chassis will help in reducing the possible noise. This matter is taken up in the discussion of the chassis layout later on in this report. The other problem with the sensor system is the layout of the sensors. The circuit boards were not efficiently designed for connectors to be used. This was due to a lack of experience on the part of the circuit board designer. If more time were allotted for the overall design, revisions in the circuit board layout would allow for connectors to be used in an efficient matter in accordance with the project specifications.

2.2 MOTOR CONTROL SUBSYSTEM

Motor control will be discussed in this section. Motor Control consists of three major parts: pulse width modulation, H-bridges, and encoder processing. This section

will also discuss the attempt to run motor control independently in an Altera chip. The reasons for the failure of this system will be discussed, as well as solutions we attempted.

2.2.1 Pulse Width Modulation

This section will explain the two methods of generating a PWM signal for use in a small mobile robot. PWM, or Pulse Width Modulation, is used to control the robot's motors. The first method of generating such a signal uses the Motorola MC68HC11 microcontroller. The second method uses two Programmable Logic Devices (PLD) to generate the PWM signal. This report will show that the second method is the most viable.

PWM is a method of generating a variable DC signal from a non-variable DC source. The base of this PWM signal is a square wave (shown below, in Figure 2.2.1).

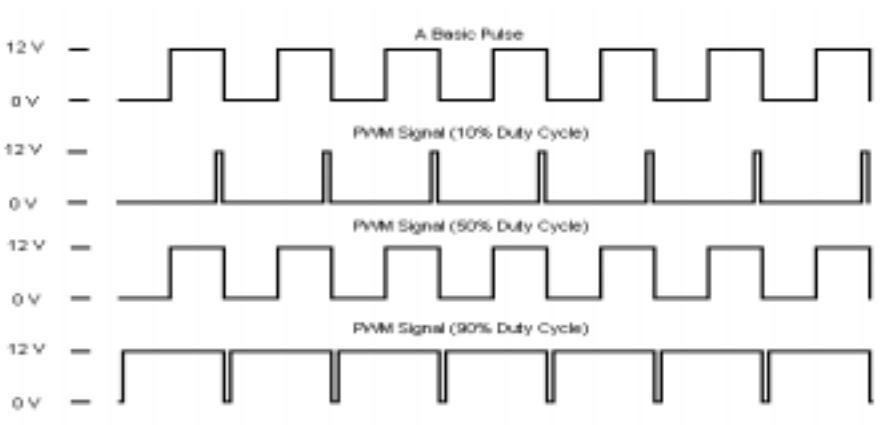


Figure 2.2.1: Comparison of PWM and Pulse Signal

As the above diagram shows, a PWM signal is a signal that is set to its maximum value for a variable amount of time. The amount of time, per period, that the signal is set at 12 volts is referred to as the duty cycle, and given as a percentage of the total period (or that period of time over which the signal does not repeat itself). In Figure 2.2.1 above, 10%, 50% and 90% duty cycles are shown. It is important to note four things:

1. A 50% duty cycle is identical to the basic pulse from which the PWM signal is derived.
2. Given a 90% duty cycle, the signal is at 12 Volts for the majority of the time.
3. A 0% duty cycle would result in a signal that was fixed at 0 Volts.
4. A 100% duty cycle would result in a signal fixed at 12 Volts.

The PWM signal used by this robot runs at 976.56 Hz, a consequence of the motors' construction, as well as programming restrictions. Altera, the company that manufactures the PLD chips used, only allows division by powers of two in Altera Hardware Design Language, or AHDL.

To determine this frequency, simple trial and error was used. The motors were connected to a function generator, which output an 80% duty cycle PWM signal. The frequency, starting at 100 kHz, was then varied by increments of first 10 kHz, then 100 Hz, and lastly 10 Hz, until the motors stopped emitting a high pitched whine. The resulting frequency was 1 kHz (1000 Hz), and 976.56 Hz was the closest frequency that could be generated. Dividing the HC11's E-Clock, which runs at 2 MHz, by 2048 (within the Altera chip), generated the signal.

Unfortunately, the implementation was not quite so easy. The first time the clock divider program was written, we wrote it incorrectly. The incorrect program is shown below.

```
SUBDESIGN divclk
(
        CLUCK3                  : INPUT;        % CLOCK is a reserved word %
        CLUCK2                  : OUTPUT;
)
VARIABLE
        DRACULA[11..0]          : DFF;  %  COUNT Dracula.  Get it?  %
```

```
BEGIN

        DRACULA[11..0].CLK = CLUCK3;

        IF DRACULA[11..0] < 2048 THEN
                DRACULA[11..0].d = DRACULA[11..0].q + 1;
        ELSE
                CLUCK2 = VCC;
        END IF;
END;
```

Figure 2.2.2: Incorrect Clock Divider Program

This program would have worked correctly, except we made a simple error by forgetting to reset the counter. The following, correct, program, shows this fix.

```
SUBDESIGN divclk
(
        CLUCK3                  : INPUT;        % CLOCK is a reserved word %
        CLUCK2                  : OUTPUT;
)

VARIABLE
        DRACULA[11..0]          : DFF;   %  COUNT Dracula.  Get it?  %
BEGIN

   DRACULA[11..0].CLK = CLUCK3;

        IF DRACULA[11..0] < 2048 THEN
                DRACULA[11..0].d = DRACULA[11..0].q + 1;
        ELSE
                CLUCK2 = VCC;
                DRACULA[11..0].d = GND;
        END IF;
END;
```

Figure 2.2.3: Correct Clock Divider Program

There was, however, insufficient time (after the bug was found) to implement this change.

Signal Generation:

There are two feasible methods of PWM signal generation. The first method involves using a Programmable Logic Device (or PLD). The second involves using the HC11's output compare subsystem. Both methods involve programming and use the same basic concepts.

The first method, writing a program to generate the PWM signal from an Altera chip, was deemed the most feasible. There are several reasons for this.

1. Altera's AHDL is used in a digital electronics course, so we are familiar with this language.
2. Altera chip programmers are already installed and used in the lab (especially the EPM7128ALC84-12 and EPM7032LC44-15).
3. Other PLD programming software may be difficult to find or have too steep of a learning curve for a one-semester design course.

The program itself is simple. Capable of generating a PWM signal for each of the two motors, it consists mainly of two counters, which count up to an input 8 bit value (a desired speed, sent from the HC11). While the value of the counter is less than or equal to the input value, a logic high (5 volts) is output on the PWM signal line. The moment the counter's value becomes higher than the input value, a logic low (0 volts) is output on the PWM signal line. The overall result of this is a PWM signal, with a resolution of $1/256^{th}$ of the total signal period. Since the clock signal input (a clock is necessary to run the counters) is set at 976.56 Hz, the following calculations apply. For signal period T:

$$T=1/f = 1/(976.56\ Hz) = 0.001024\ sec = 1.024\ ms$$

Since the period T of any signal is equal to 1 over the frequency of that signal, the period of the PWM signal is 1.024 milliseconds. This means that the signal repeats itself every 1.024 milliseconds. The signal resolution (or minimum amount by which the signal can change) is $1/256^{th}$ of the overall period, or Signal Resolution, R is:

$$R = (1/256)*(0.001024) = 0.000004\ sec = 4\ \mu s$$

Since the motors are capable of change only at a very slow rate, this is more than enough resolution. A consequence of this same principle is that the desired speed, fed as input from the HC11) is limited to a number between 0 and 255.

The Altera PWM program is shown below.

```
% OPEN LOOP PWM %
SUBDESIGN open
(
        CLUCK2                          : INPUT;   % CLOCK is a reserved word %
        R_DES_SPEED[7..0]               : INPUT;
        L_DES_SPEED[7..0]               : INPUT;

        R_PWM                           : OUTPUT;
        L_PWM                           : OUTPUT;
)
VARIABLE
        DRACULA[7..0]                   : DFF;   %  COUNT Dracula.  Get it?  %
        CHOCULA[7..0]                   : DFF;   %  COUNT Chocula         %
BEGIN
        DRACULA[7..0].CLK = CLUCK2;
        CHOCULA[7..0].CLK = CLUCK2;

        DRACULA[7..0].d = DRACULA[7..0].q + 1;
        CHOCULA[7..0].d = CHOCULA[7..0].q + 1;

        IF DRACULA[7..0] <= R_DES_SPEED[7..0] THEN
                R_PWM = VCC;
        ELSE
                R_PWM = GND;
        END IF;
        IF CHOCULA[7..0] <= L_DES_SPEED[7..0] THEN
                L_PWM = VCC;
        ELSE
                L_PWM = GND;
        END IF;
END;
```

Figure 2.2.4: PWM Generation Program

The second method involves setting up the HC11's OCx (or Output Compare) pins. This subsystem takes a value, input to a register (a device capable of holding an 8 bit binary number) and compares it to a counter in much the same way as the Altera program. A logic high (5 volts) is output while the counter is less than or equal to the input value, and the OCx pin transitions to a logic low (0 volts) when the counter value goes above the input value.

A program that can be used to implement PWM signal generation on the HC11 is shown below.

```
#include <hc11.h>

#define TRUE  1
#define FALSE  0
#define PERIOD  250

void tic2_isr(void);
void toc1_isr(void);
void putint(unsigned int i);

unsigned int first, second, tr, done, DUTY_CYCLE;

main ( )
{
        done = FALSE;
        tr = 0;
        first = 0x00;
        second = 0x00;

        /* Setup IC2 */
        TIC2_JMP = JMP_OP_CODE;
        TIC2_VEC = tic2_isr;
        TCTL2 = (TCTL2 | 0x08) & ~0x04;      /* Capture Falling Edge */
        TMSK1 = TMSK1 | 0x02;
        TFLG1 = 0x02;

        /* Setup OC1 to bring OC2 high with interrupt */
        TOC1_JMP = JMP_OP_CODE;
        TOC1_VEC = toc1_isr;
        OC1M = OC1M | 0x40;
        OC1D = OC1D | 0x40;
        TMSK1 = TMSK1 | 0x80;
        TFLG1 = 0x80;
        TOC1 = PERIOD;

        /* Setup OC1 to bring OC2 low w/o interrupt */
        TCTL1 = ( TCTL1 | 0x80 ) & ~0x40;


        enable( ); /* General interrupt enable */

        while ( !done );
        tr = second - first;    /* Calculate Time Elapsed */
        putint(tr);
}

#pragma interrupt_handler tic2_isr
void tic2_isr (void)
{
        if ( !first)
        {
                first = TIC2;
        }
        else
        {
                second = TIC2;
        }
```

```
        if (second)
        {
                done = TRUE;
        }
        TFLG = 0x02;
}

#pragma interrupt_handler toc1_isr
void toc1_isr (void)
{
        unsigned int DUTY_CYCLE;

        DUTY_CYCLE = PORTC & 0x07;  /* Mask out lower 3 bits */
        DUTY_CYCLE = 25 * (DUTY_CYCLE + 2);
        TOC2 = TOC1 + DUTY_CYCLE;
        TOC1 = TOC1 + PERIOD;
        TFLG1 = 0x80;
}

void putint(unsigned int i)
{
        asm (“     jsr  0xffc1”);
        asm (“     jsr  0xffc4”);
}
```

Figure 2.2.5: Sample HC11 code for PWM Generation

The Altera PLD chip and the HC11 subsystem are the two most feasible methods of generating a PWM signal. Both involve programming. The Altera program, however, is much more simple, and contains much less that is capable of malfunctioning. In addition, the HC11 is very busy with other functions, and its workload is already approaching a dangerous level. Therefore, we decided to generate the PWM signal in Altera.

2.2.2 H-Bridge Motor Drivers

Upon working with our motor control subsystem, a suitable motor driver was necessary to be able to change motor speeds and directions. Looking through all of our options, we chose the LMD18200 H-bridge package from National Semiconductor. Although this is not a dual bridge package capable of running both motors from the same chip, the EE Department provided us two of these H-bridges. We decided to use them in order to minimize

cost. The H-bridge takes the PWM and direction signal inputs and outputs a signal to the motor that turns the wheels in both the direction and speed desired.

Our group decided to simulate PWM in Altera. This was discussed in the previous section. We generated one PWM signal for each motor out of Altera and a direction line for each motor out of the Motorola 68HC11 microcontroller. Each of these lines is a digital signal that is fed into the H-bridge. A 12V battery was attached to the H-bridge to supply power to the circuit and to convert the digital PWM signal into a 0-12V output to the motors. Based on the weight of our robust design (about 15 lbs.), the robot would not move unless the motors were supplied with this increase in PWM amplitude. The encoders then send back a signal to the HC11 to tell it how quickly the motors are spinning. Knowing this rotation speed, and that the encoders send out 500 pulses per rotation, Altera approximates the current actual speed. Using this actual speed measurement, we compare it to a desired speed number and determine whether we need to slow down or speed up. A block diagram of what the circuit entails is included below:

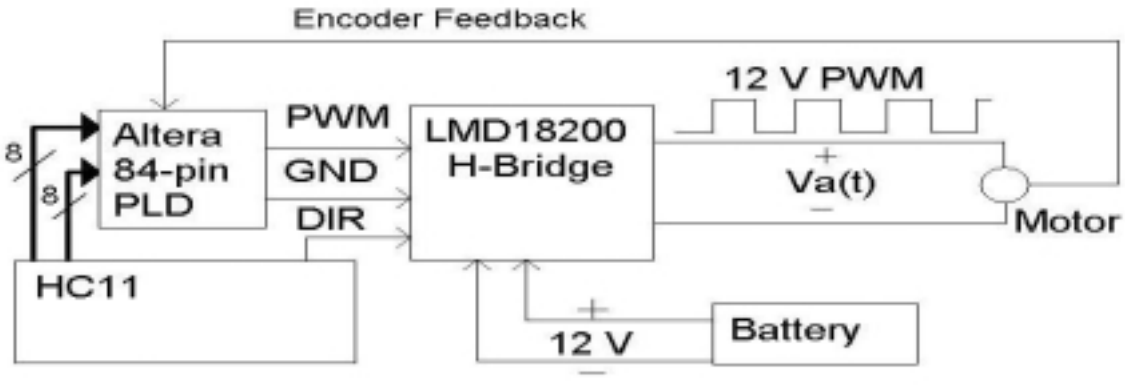


Figure 2.2.6: Block Diagram of Motor Control Subsystem

Next we have to consider that the H-bridge must be fit into the circuit properly, otherwise, it will not output what it is supposed to. Using the National Semiconductor

Data Sheet for the LMD18200, we constructed two boards using the circuit schematic shown below:

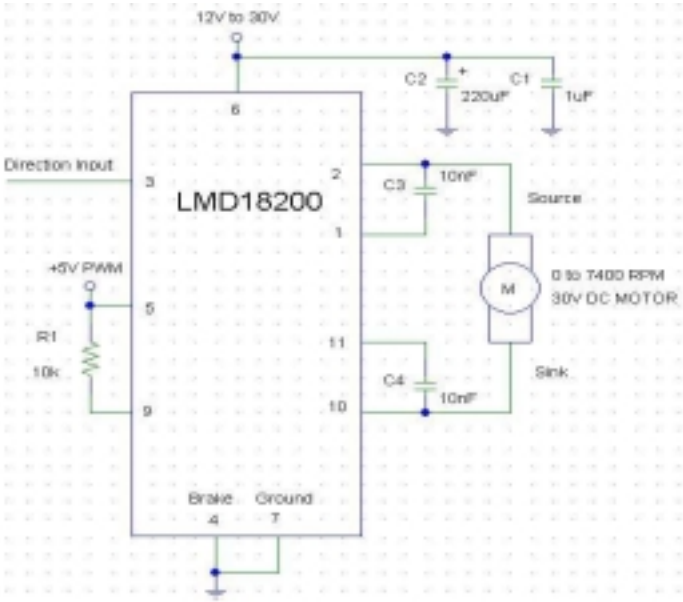


Figure 2.2.7: Schematic for LMD18200 H-bridge Motor Driver

Using this schematic and the block diagram from Figure 2.2.6, we constructed a feasible motor control subsystem. However, we had issues with the boards from the start. We had to design a printed circuit board that would hold the entire H-bridge circuit design. We drew up a board layout that would allow us to make the proper pin connections from the H-bridge to the resistors and capacitors on the board. Upon tracing the layout on the first board and etching it, we found that the board was etched backwards. The H-bridge had to be put on the copper side rather than the component side. This makes soldering the components to the board next to impossible. We mirror-imaged the layout such that the proper etching design would be achieved and etched another board. Upon doing this, we soldered all of the components onto the board and found it to be an almost flawless design.

We then hooked up the H-bridges to the motors to see if they would drive them properly. After only about a minute of testing, a component on the board exploded. The polarized filter capacitor had been soldered on the board backwards. After correcting this oversight, we tried again. This time we noticed we had a few loose soldering connections. One of them involved ground not being connected from the copper to the H-bridge. Without this ground reference, the H-bridge would not be able to determine that the power source was at a 12V potential. Upon fixing these soldering problems, and hooking the H-bridge up to all necessary input and output lines, they both worked flawlessly.

2.2.3 Encoder Processing

The initial design for encoder signal processing was two customized pulse accumulators, one for each motor, and a divider for the HC11's E-clock programmed into an Altera chip. The circuitry for each pulse accumulator consisted of an eight bit counter, an eight-bit shift register and an eight-bit latch. The E-clock divider consisted of a single eight bit counter, which divided the E-clock down to produce a clock signal at 7.8 kHz. The encoder signal was passed through the eight bit counter to reduce the number of pulses received during a single clock cycle. This reduced signal was then passed to a shift register to convert it from a serial signal to an eight bit parallel number. This parallel number was then passed to the latch, which was triggered by the clock signal. This eight-bit number and modified clock signal were then passed to the closed loop controller program, which was also programmed in Altera. The overall block diagram of the initial system is shown in Figure 2.2.8 below.

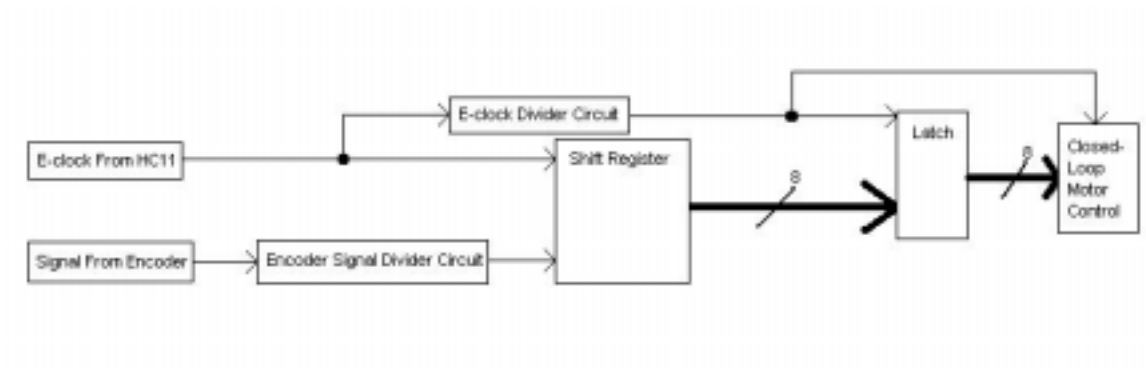


Figure 2.2.8: Encoder Signal Processing Design

This design proved less than ideal. The first problem arose from the fact that the latch and shift register were both triggered by the same clock signal at the same frequency. This problem was easily remedied; the clock line for the shift register was taken from a lower order bit on the E-clock's counter. This allowed the shift register to output the complete eight-bit number before it was latched. After this simple but effective modification the encoder signal processor worked as planned under simulation but it still failed to correctly interface with the actual systems of the robot. These interface problems were caused by a failure to meet the shift register's timing requirements.

In the second design, the shift register was removed and the needed data was taken directly from the counter. However, it was then discovered that the clock rate was incorrect for system requirements. To remedy this, the clock was replaced within the encoder signal processing circuitry by a chip select line ran from the HC11. This design, however, had too much resolution for the rest of the motor control components to process.

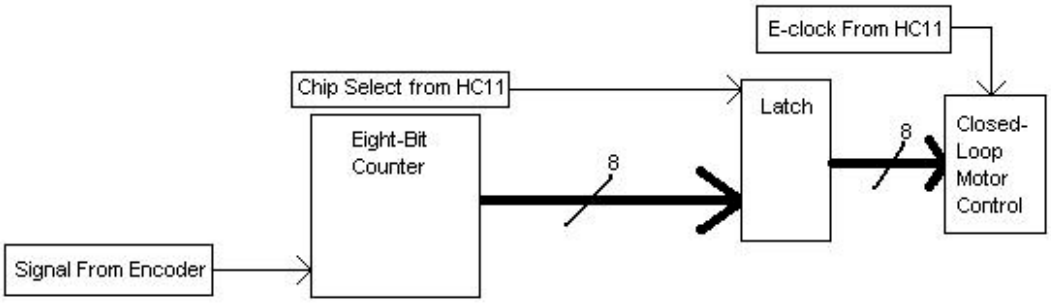


Figure 2.2.9: Second Encoder Processing Design

The next step in correcting the many problems with motor control was to reintroduce a divider to reduce the number of pulses received by encoder signal processing and the rest of the motor control system. It was soon discovered that even with reduced resolution, the encoder signal processing still inexplicably failed to function.

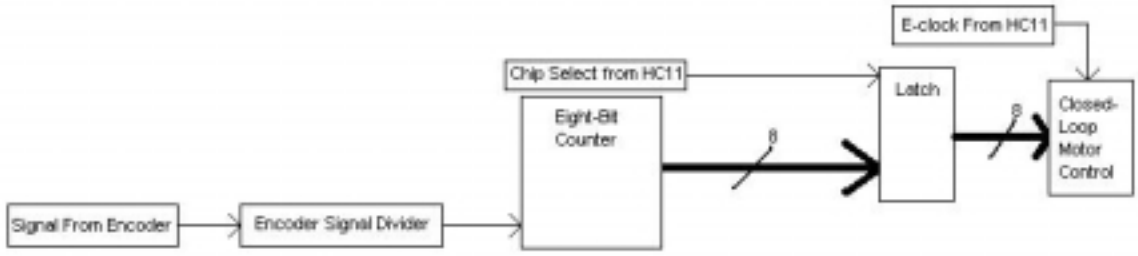


Figure 2.2.10: Third Encoder Processing Design

After weeks of designing, wire wrapping and testing each of these encoder signal processors the decision was made to attempt to program the closed loop portion of the motor control into the HC11. There are two reasons why we decided to move most of the motor control in the HC11. First, it would be difficult to modify motor constants. Second, it would be

extremely difficult to implement a proportional or integral motor control in Altera without first designing a floating-point processor and arithmetical logical unit for use in Altera. It was also decided to only use one 128-logic-cell Altera chip to run the PWM portion of the motor control.

Attempting to run motor control independent of the HC11 is not, in hindsight, a feasible possibility for a one-semester course. It is both easier and more efficient (as a system which does not work cannot be labeled efficient) to leave motor control to the HC11. Pulse Width Modulation, however, can easily be generated, and the encoder signals resolved into something usable, in a single EPM7128ALC84-12 Altera chip.

2.3 HC11 CONTROL CODE

The requirements laid out for the robot state that the control program be written in the C programming language for the HC11. The control code has been included in Appendix A for continuity reasons. A flowchart describing the overall HC11 control code is shown in Figure 2.3.1 below.

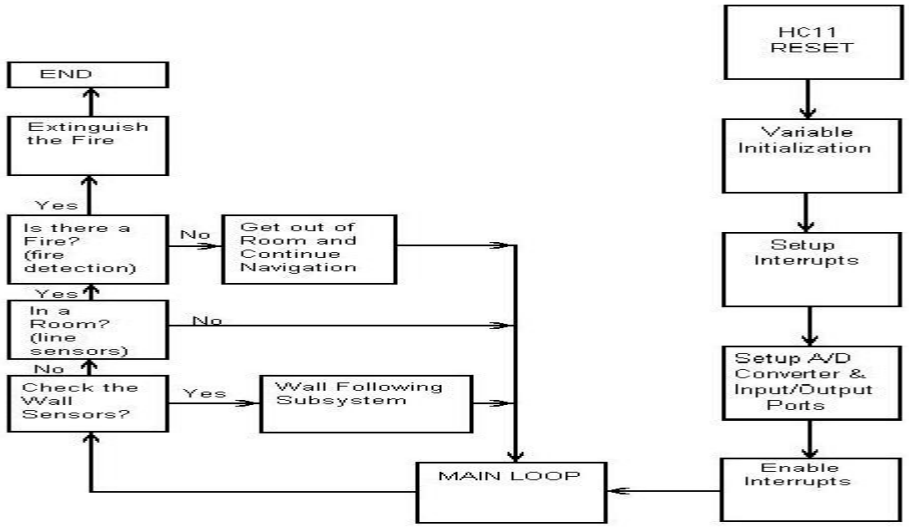


Figure 2.3.1: Flowchart for HC11 Control Code

The overall goal of the HC11 code is to be as modular as possible with minimal events occurring when interrupts are active. Currently, the control code is incomplete, as we were in the process of writing code to make the robot follow the right wall.

The status of the code is as follows:

- The code has not been tested or compiled.
- The initialization of most of the variables, interrupts, and functions are complete.
- The constants concerning most of the comparisons have not been tested and verified.
- The wall-following code is complete, but it will not detect the so-called “fourth room” where normal wall following bypasses it.
- Motor control is currently incomplete. As of this report, it was in the process of being refined to include as few floating-point operations as possible, since the HC11 does not contain a floating-point coprocessor.
- The line-detection code is complete.
- The fire detection code has been pseudo-coded but not translated into C code.
- The fire suppression code once the fire has been detected has been written but not yet tested.
- No optimization whatsoever concerning individual rooms has been completed.
- Any code to add additional features such as returning to the home circle once the fire was extinguished has not been written.

As of this report, several of the HC11 interrupts are being used to trigger key events for the robot. The Real Time Interrupt (RTI) is being triggered every 8 milliseconds to tell the HC11 to do the closed-loop motor adjustments and calculations.

The Timer Overflow Interrupt (TOF) is used to trigger a wall-following calculation every 32 milliseconds. The line sensors are tied to two Timer Input Capture Interrupts (TIC) so the HC11 can sense when the robot is going into a room. The TIC interrupt is set up to where the HC11 receives a rising edge on the input capture pins. Currently, the PIA expansion port is set to receive data off of some external connection, but one can use the Interrupt Request Interrupt (IRQ) in order to receive or send data to an external connection.

The basic wall following code is in place. Figure 2.3.2, below, shows the logic of the wall following process.

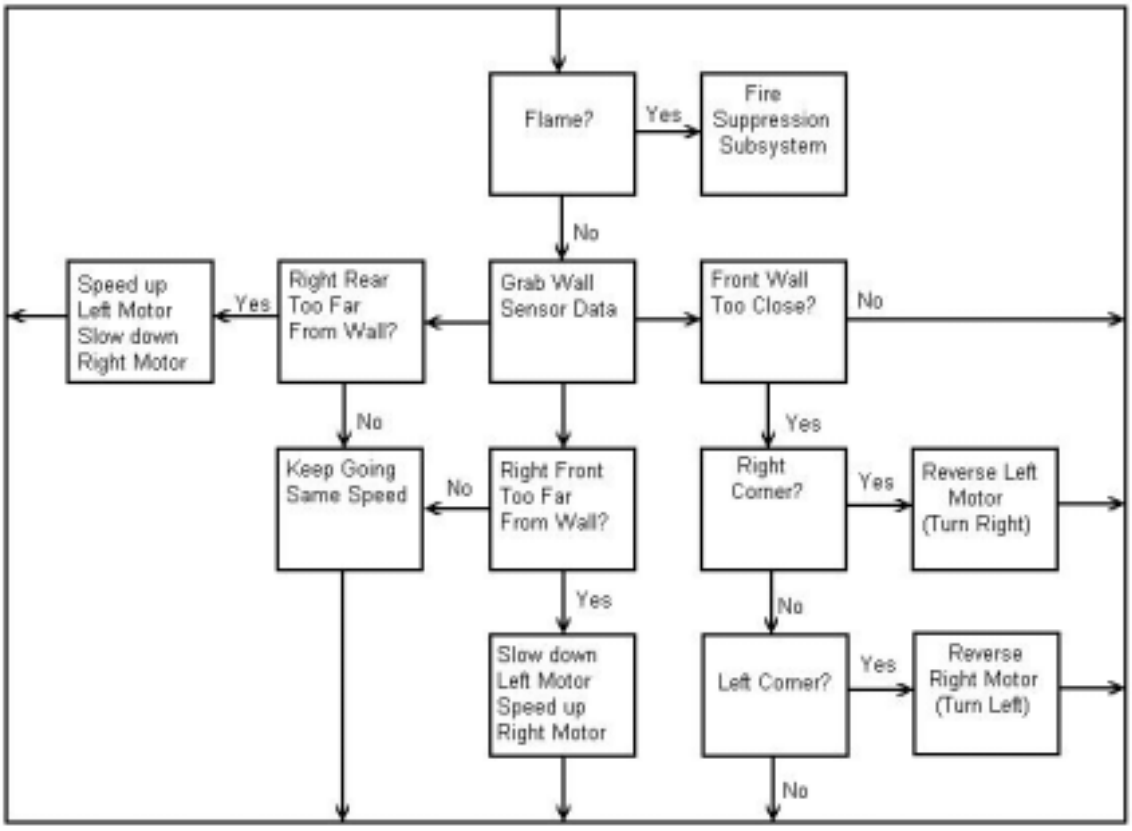


Figure 2.3.2: Wall Following Block Diagram

There are some features that are in the code that make debugging and addition of code easier. For modularity of code and for easier debugging, a large percentage of the

code is written as functions. A minimal amount of work is done within the interrupt routines in order for the HC11 to service other routines in a quick manner. The main loop does a minimal amount of work except when an interrupt flag is triggered. Most of the control code is heavily commented so other designers know what is going on within it. Overall, while the design of the control code is in its early stages, we are making sure the reliability of the code is extremely high from the start.

2.4 FIRE SUPPRESSION

Once the fire is located, the robot must take steps to extinguish the flame. Several options are available for accomplishing this task. The first of these options is a small fan, which blows out the flame. We elected to use a small stream of water to extinguish the candle flame. Each of these options has its advantages. However, we feel that the water stream is the most viable solution, being applicable to the widest range of possible fires.

The water stream method, in turn, requires several things:

1. A Water Tank
2. A Water Pump
3. Rubber Hosing and Nozzles
4. Sprinkler Head
5. Secondary Batteries
6. Interface Circuitry

A diagram of the entire fire suppression subsystem is shown in Figure 2.4.1 below.

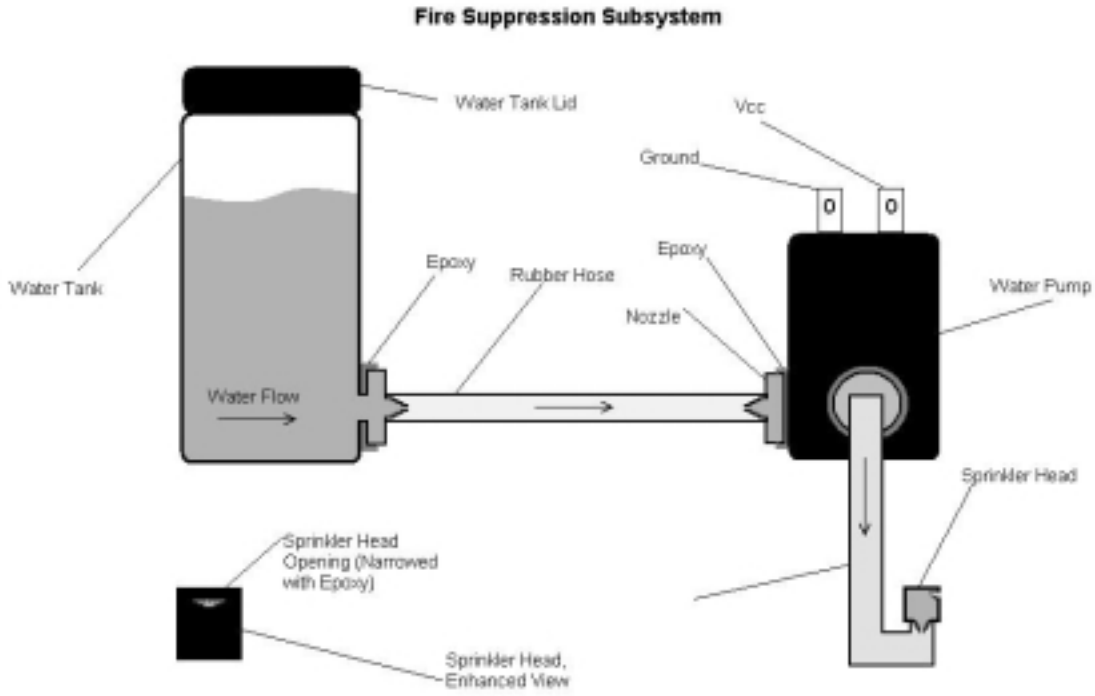


Figure 2.4.1: Fire Suppression Subsystem

2.4.1 Water Tank

The water tank was constructed from a plastic salt and pepper shaker set. Each, individually, was capable of holding 4 ounces of water. To increase the total capacity, the bottom was cut out of one of the canisters. The canisters were then sealed together using epoxy. Epoxy, however, proved vulnerable to the liquid mixture used in place of water (glass cleaner with ammonia and water). The epoxy seals were turned into a rubbery substance by the glass cleaner. Hot Glue was found to be a viable substitute.

2.4.2 Water Pump

The water pump was taken from a 1987 Dodge Colt windshield washer pump assembly. The motor, obtained for the cheap price of $5, was comparable to other pumps found on the Internet for $40 or more. The pump, made by Nippondenso Company, operated

at 12 Volts and drew 1 ampere of current. This was more than sufficient for our needs, and drained the tank in less than 20 seconds.

2.4.3 Rubber Hosing and Nozzles

Eight feet of rubber hosing was obtained from Ace Hardware. Assorted nylon nozzles were used to mate the rubber hosing to the water tank, pump, and sprinkler head. Epoxy was used to seal the nozzles to the water tank and pump, as the ammonia solution was not in contact with it. The nozzle used to mate the water pump to the hosing had to have its threading in order to fit.

2.4.4 Sprinkler Head

A simple sprinkler head, obtained from Ace Hardware, was used to direct and concentrate the stream of ammonia solution. The sprinkler head, normally used to widen a stream of water to soak a wide area, was altered using hot glue. The effect of this alteration was to narrow the liquid stream by sealing all but the center portion with hot glue.

2.4.5 Secondary Batteries

Secondary batteries were needed, due to the massive power consumption of the total fire suppression system. 12 Volts, at 1 Amp, even for the 20 seconds needed to completely drain the water tank, would drain the main battery of too much power. To fix this problem, secondary batteries were obtained. Meant to power answering machines, two batteries, each capable of 550 mA at 3.6 volts, were obtained for the water pump.

2.4.6 Interface Circuitry

Due to time constraints, the circuitry to interface the fire suppression system to the HC11 was not built. Had time constraints allowed its construction, figure 2.4.2 below shows what it would have looked like.

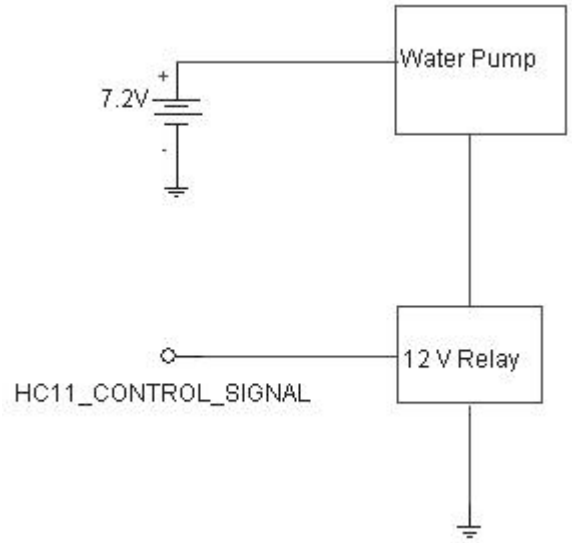


Figure 2.4.2: Interface Circuitry for Fire Suppression Subsystem

2.5 POWER BOARD IMPLEMENTATION

The implementation of the robot we have designed requires a variety of DC voltages in order to operate in an efficient manner. For instance, the HC11 and sensor system requires 5 volts, the motors require 12 volts, and the Altera PLD requires between 3.3 and 4.5 volts. Our current robot design uses one 12-volt battery as its primary source of power. Hence, it is desirable to use a system in order to step down the 12-volt battery source into 5-volt systems to assure proper operation of the sensors and digital devices

and to prevent a catastrophic failure of individual components. We decided to go along with two separate DC-to-DC converter implementations for the robot. The first converter outputs 5 volts for the sensor systems. The other converter is made adjustable for the varying requirements for the Altera device and the HC11.

The schematic for the 5-volt regulator is shown in Figure 2.5.1 below.

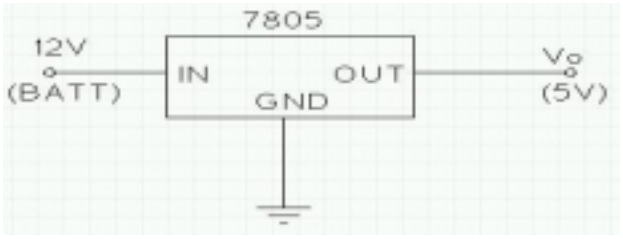


Figure 2.5.1: Schematic for a 5-Volt Converter From a 12-V Battery

This circuit is relatively trivial. We used a 7805 voltage regulator wired as above in order to step down the battery to a 5-volt source for the sensors. The maximum current output of the 7805 used is approximately one ampere. This is more than enough current to source all of the sensors with their required power.

The second voltage converter schematic is shown in Figure 2.5.2 below.

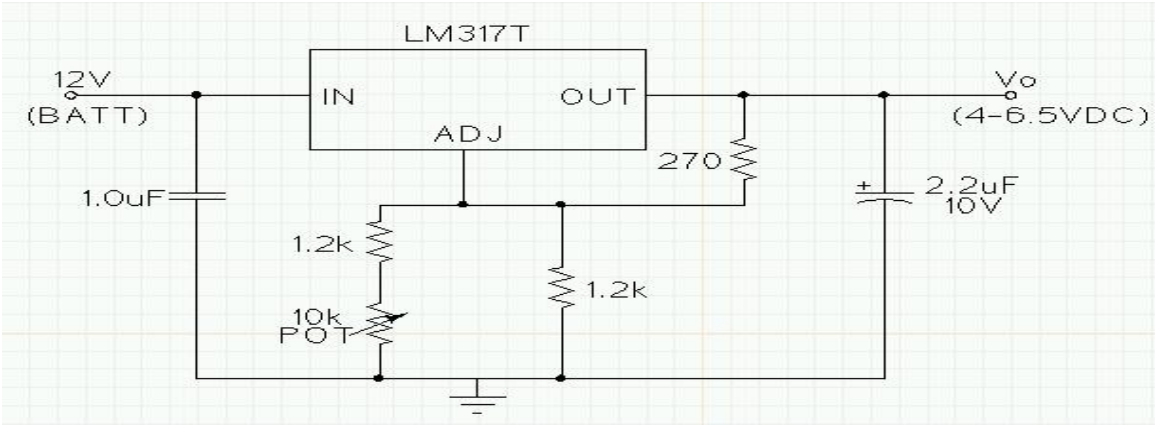


Figure 2.5.2: Schematic for an Adjustable Voltage Converter and Regulator

The basic concept of the schematic shown above was taken from a circuit idea in National Semiconductor's Power IC Data book. The LM317 is an adjustable voltage regulator that

outputs between 1.5 and 28 volts dependent on the design of the additional circuitry and the input voltage to the regulator. The capacitors are included in the design to filter out noise and to make the output more stable. For reasons unknown at this time, the resistor connecting the output terminal to the adjustment terminal on the LM317 should be near 240 ohms. The resistor combination connecting the adjustment terminal of the LM317 to ground is the method of voltage adjustment. The resistor-potentiometer combination provides an equivalent resistance for the LM317 so that it can produce an output voltage defined by

$$Vout = 1.25 * (1 + R2/R1)$$

Where R1 is the feedback resistor between the output and adjustment pins of the LM317 and R2 is the resistance seen between the adjustment terminal of the LM317 and ground. The output of the voltage regulator is as follows. When the potentiometer is adjusted to 0 ohms, the equivalent resistance of R2 is seen to be approximately 600 ohms; hence the output voltage is calculated at about 4.02 volts. When the potentiometer is adjusted to be 10 kilohms, the equivalent resistance of R2 is approximately 1084 ohms, and the output voltage should be about 6.27 volts. However, due to tolerances with the resistors, some slight variation will result with the maximum and minimum output voltages. Each of the LM317 regulators can source a maximum current of one ampere, so the power requirements of the HC11 and the Altera device are satisfied with ease.

We used two of these adjustable regulators in the robot. One of these regulators is for the exclusive use of the Altera PLD chip, as it requires operation at a slightly lower voltage than 5 volts. The second regulator is for the HC11 microprocessor, as it has a tendency to reset itself if the source drops below a voltage around 4.5 volts. On the recommendation of Dr. Stephen Bruder, we are running the HC11 at a supply voltage of

approximately 5.2 volts in order to prevent accidental resets while the robot is moving around the maze. There are also additional connection terminals if the 7805 is not able to source enough current to adequately power all of the power system.

To make sure the voltage regulators were able to supply the maximum current possible, heat sinks were installed and heat-sinking compound was applied to the voltage regulators in order for them to dissipate the required heat. The power board is fully operational and supplies the voltages required by other subsystems. The minimum and maximum voltages from the adjustable voltage regulators are very close to the calculated voltages using the formula above.

2.6 CHASSIS DESIGN

Chassis design began with a disc-shaped aluminum plate with two motors attached by aluminum "L"-brackets in a differential drive configuration that was centered on the disc. Moving the motors forward by the width of the mounting brackets and the addition of a rear caster wheel altered this basic design. The new configuration, while providing a more stable base, still allowed the robot to maintain the turn-in-place capability that is necessary for maneuvering in the tight confines of the maze. To add further stability and to save space, the battery was attached, with Velcro, to the bottom of the base between the motors and the caster wheel. By placing the single heaviest component on the bottom of the robot, the center of gravity was relocated significantly lower on the robot. After completing these alterations to the basic design of the base several other modifications were required to attach necessary and vital equipment.

First, a number of different sized holes were drilled in the base. These holes were drilled to allow for wiring, the attachment of the power board, the H-bridges, which were

attached using one inch hex standoffs, and white line sensors, which were attached using hot glue. They were also drilled for the placement of three-eighths inch all-thread, which was chosen for the structural strength it added to the robot chassis. This all-thread was secured to the base with hex nuts and lock washers on the top and bottom of the base. See Figure 2.6.1 below:

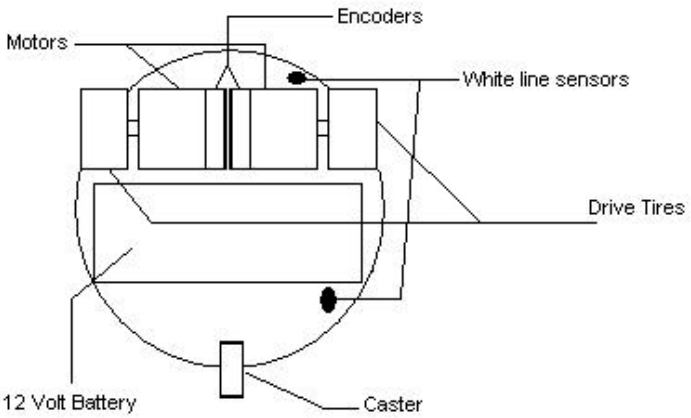


Figure 2.6.1: Bottom View of Base

The water pump and tank were attached to the bottom level with Hot Glue. Next, to ensure stability when filled, the tank was also braced with nylon cable ties that were then attached to the all-thread. Finally, the second and third levels of the robot were connected to the base via the all-thread by drilling slotted holes for the all-thread to pass through. The additional levels were then secured by hex nuts resting on one and one-half inch flat washers to prevent the nuts from cutting into the masonite. See Figure 2.6.2 on the next page.

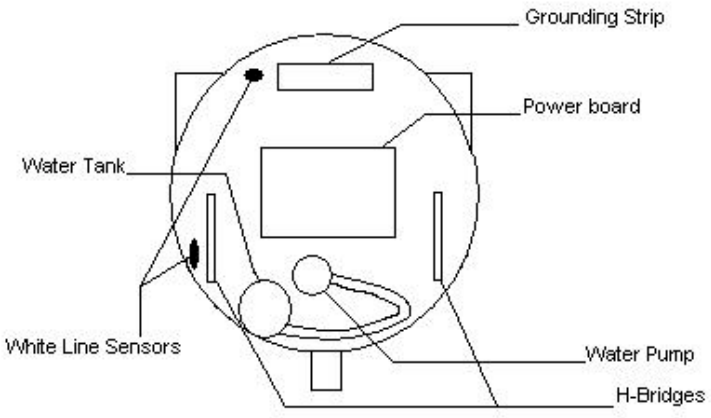


Figure 2.6.2: Top View of Base

The second and third levels were cut from a sheet of masonite; a composite material resembling compressed cardboard, which was chosen for a number of reasons. The main reasons for choosing masonite were weight, rigidity, price, and availability. Masonite is a very lightweight material that is extremely rigid in small sections. It is also inexpensive, a four-foot by eight-foot sheet of one quarter-inch masonite costs about twelve dollars, that is approximately one quarter the cost of sheet aluminum, and can be found at most lumber yards and many hardware stores.

The second level had to have holes drilled into it so we could mount the wall sensors and wall sensor board. The wall sensor board was mounted using one-inch hex aluminum standoffs. Then a larger hole was drilled into the center of the second level to pass wires through. It was then discovered that the water tank was too tall to fit between the base and second level. This required that a large slot be cut in the second level, which allowed the water tank to pass through the masonite and to rise above the second level.

Finally, a three-eighths inch hole was drilled through the second level for the placement of the water pumps pressure hose and the spray nozzle. The nozzle was then secured into position with more hot glue. See Figure 2.6.3 below:

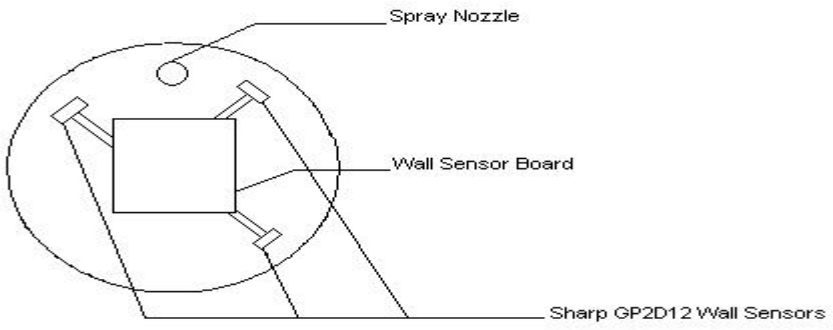


Figure 2.6.3: Second Level of Chassis

The third level was drilled to accept the Altera motor control board, the HC11 board, the fire sensor board, which were all mounted using standoffs, and the fire sensor, which was mounted using two one quarter inch nylon cable retainers placed in opposing directions. The monopulse radar fire detection system was then attached to the third level using self-adhering Velcro strips.

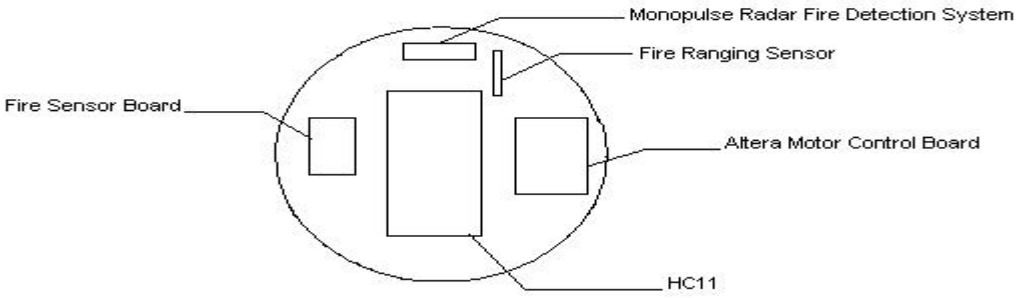


Figure 2.6.4: Third Level of Chassis

The primary reason for the distribution of the high power vs. low power systems is noise. With all of the high power systems on the base, there is a high level of noise that can cause signal bounce. To ensure that the HC11 and the Altera chip experienced the smallest amount of noise possible, it was necessary to move them as far away from the high power circuitry as possible. With the noise sensitive equipment on the third level, there were three levels of protection from noise. This design ensured that noise would not be an issue.

2.7 BUDGET ANALYSIS

For this project, our group was given a budget of $100 along with the starter kit provided by the EE department. This starter kit includes:

- 2 – Pittman GM9434 DC Servo Motors with Encoders and wheels
- 2 – LMD18200 H-bridge Motor Drivers
- 1 – 12” circular aluminum sheet for the chassis
- 1 – 12V Camcorder battery
- Connectors, chip puller, wire strippers, logic probe

Items that were available to us, through check out or by other means include:

- Threadlock and all-thread
- Velcro
- Shielded 4-conducter cable
- Copper circuit board
- Phone connector crimper

Using our budget of $100, our group acquired many more necessary parts for our robot. Using the parts acquisition sheet, found on the EE 382 web page, we bought a variety of

sensors, chips, and other parts necessary to build a completely autonomous system. An itemized list of the parts we bought can be found in Appendix B.

We used a variety of different companies to acquire the parts we needed. These companies include National Semiconductor, Digikey, Wyle, Burr-Brown, QT Optoelectronics, and Texas Instruments. These items include the voltage regulators, our EPM7128ALC64-12 Altera PLD, and a majority of the sensors that we needed for the robot.

The above companies as well as the EE Department donated a substantial portion of the items we used. These items will be denoted as donated in Appendix B as well as the pie chart below.

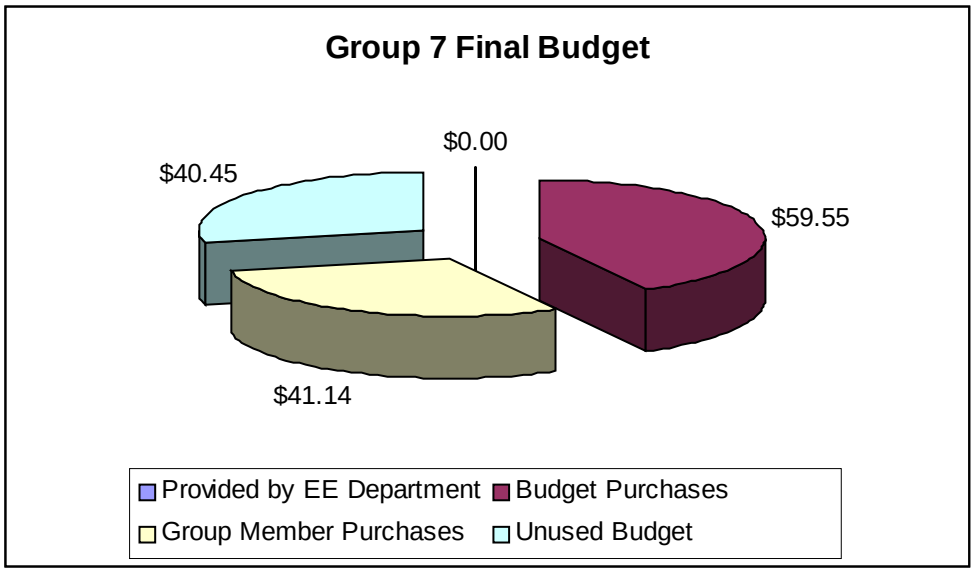


Figure 2.7.1: Pie Chart of Group 7 Budget Usage

Based on this chart, our group was well within the $100 budget given to us by the EE department. Since we were concerned about going over budget, we decided to purchase some of the "unnecessary" items ourselves. These items, including a larger diameter threadlock, fall under the "Group Member Purchases" section of the pie chart. If we had put all of our

individual purchases on the $100 budget, we would have been $0.69 over budget. When working under a budget ceiling, it is a good idea not to use more money than we have available. Doing so can cause the customer to wonder what their money is being used for. Thus, by not going over budget, the customer will be satisfied that his money is being used carefully.

3 CONCLUSION

We have determined that complete motor control from within Altera chips is not feasible for a single semester course. Two semesters might make it possible, but this group harbors serious doubts about whether it can even be done. Had we not spent so much time attempting to make Altera chips hold the substance of motor control, we feel we would have completed the project.

Sensors are fully functional on their own. PWM is functional from within an Altera chip, but still needs to be integrated with the other motor control subsystems. The H-bridges work as constructed, but also need to be tested in concert with other motor control subsystems. Encoder signal processing from within an Altera chip simulates correctly, but has not yet been physically tested. Fire suppression lacks the interface circuitry to work with the HC11.

We estimate that another month would be needed to fully integrate and test all systems to make Pokey move. The course, however, is over, and our time is up.

BIBLIOGRAPHY

Altera Corporation. (1994). *MAX+PLUS II Programmable Logic Development System AHDL*. San Hose, CA: Altera Corporation.

Burr-Brown Corporation. "Burr-Brown Products." Burr-Brown Web Site. http://www.burr-brown.com/Products/Products.html (May 10, 1999).

Digi-Key Corporation. "Digi-Key Part Search" Digi-Key Web Site. http://www.digikey.com/DigiKeySearch.html (May 10, 1999)

Jameco Electronics. "Jameco Electronics: On-Line Catalog" http://www.jameco.com/Catalog/ (May 10, 1999)

Jones, J.L., Sieger, B. A., and Flynn, A. M. (1999). *Mobile Robots: Inspiration to Implementation.* Natick, MA: A. K. Peters.

Kamen, E. and Heck, B. (1997). *Fundamentals of Signals and Systems Using Matlab.* Upper Saddle River, NJ: Prentice Hall.

National Semiconductor. "The Design Engineer Resource." National Semiconductor Web Site. http://www.national.com/design/index.html (May 10, 1999)

Tandy Corporation. "Radio Shack – You've Got Questions. We've got answers." http://www.radioshack.com/ (May 10, 1999).

Wedeward, K. "EE 382.01 Resources." New Mexico Tech Electrical Engineering Department Homepage. http://www.ee.nmt.edu/~wedeward/EE382/SP99/resources.html (Jan. 19, 1999).

Wyle Electronics. "Wyle Web Site" Wyle Web Site. http://www.wyle.com (May 10,1999).

Appendix A: Data Files and C Code for Pokey

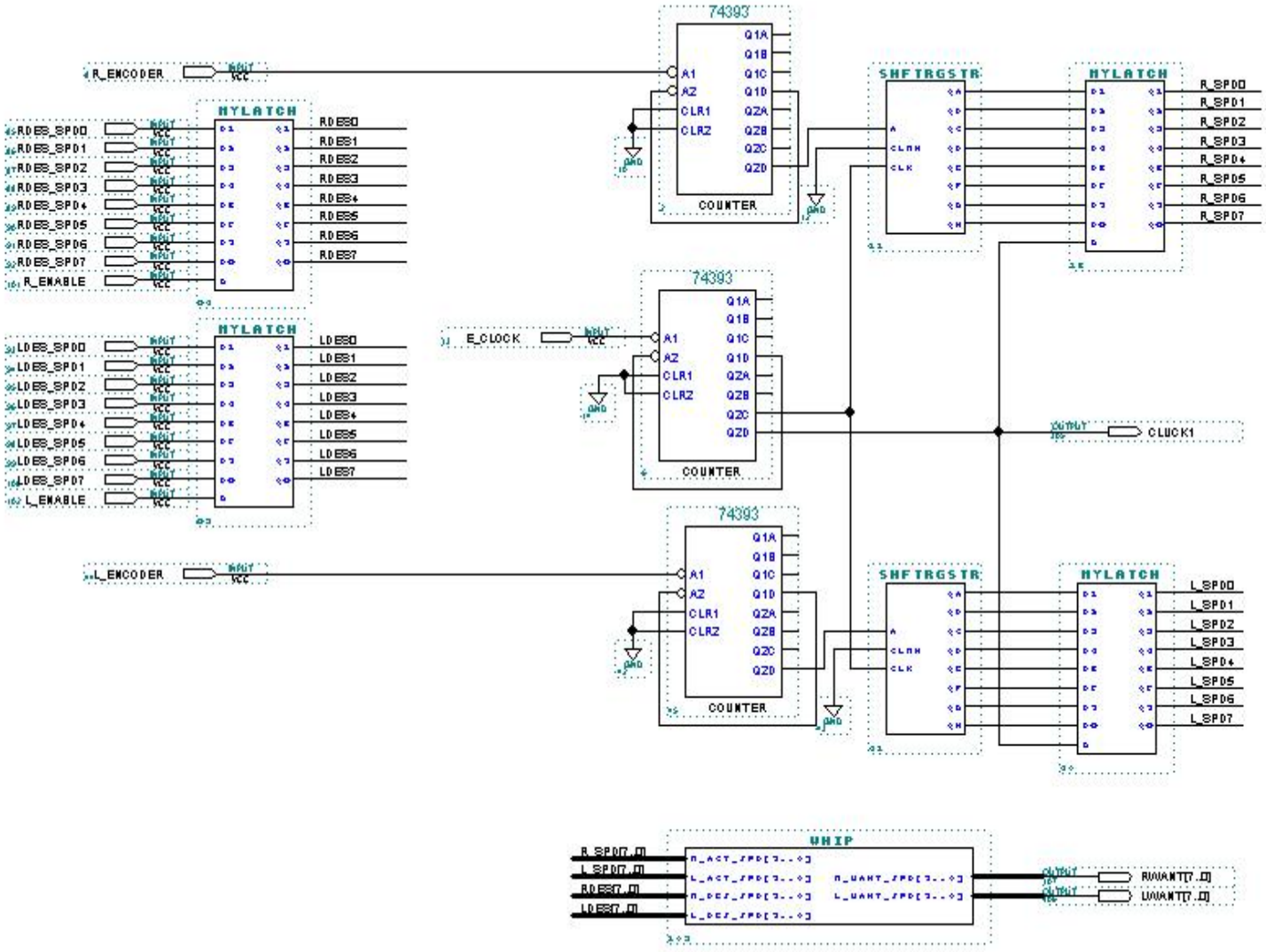


Figure 1: First Attempt at Motor Control in Altera (Part 1)

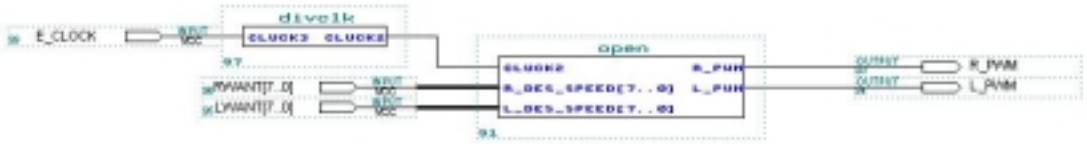


Figure 2: First Attempt at Motor Control in Altera (Part2)

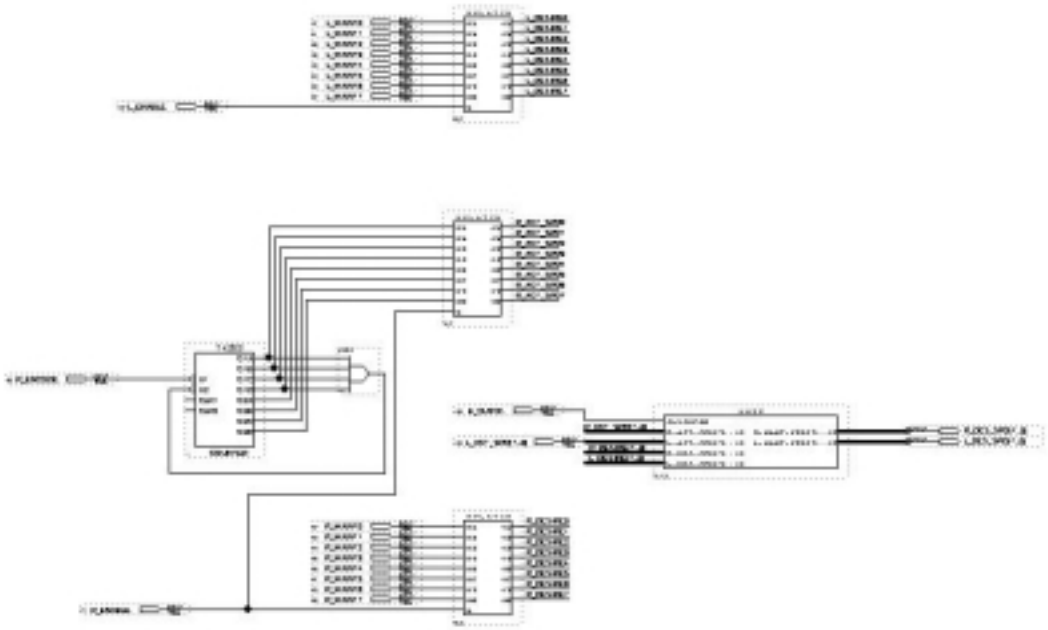


Figure 3: Second Attempt at Motor Control in Altera (Part 1)

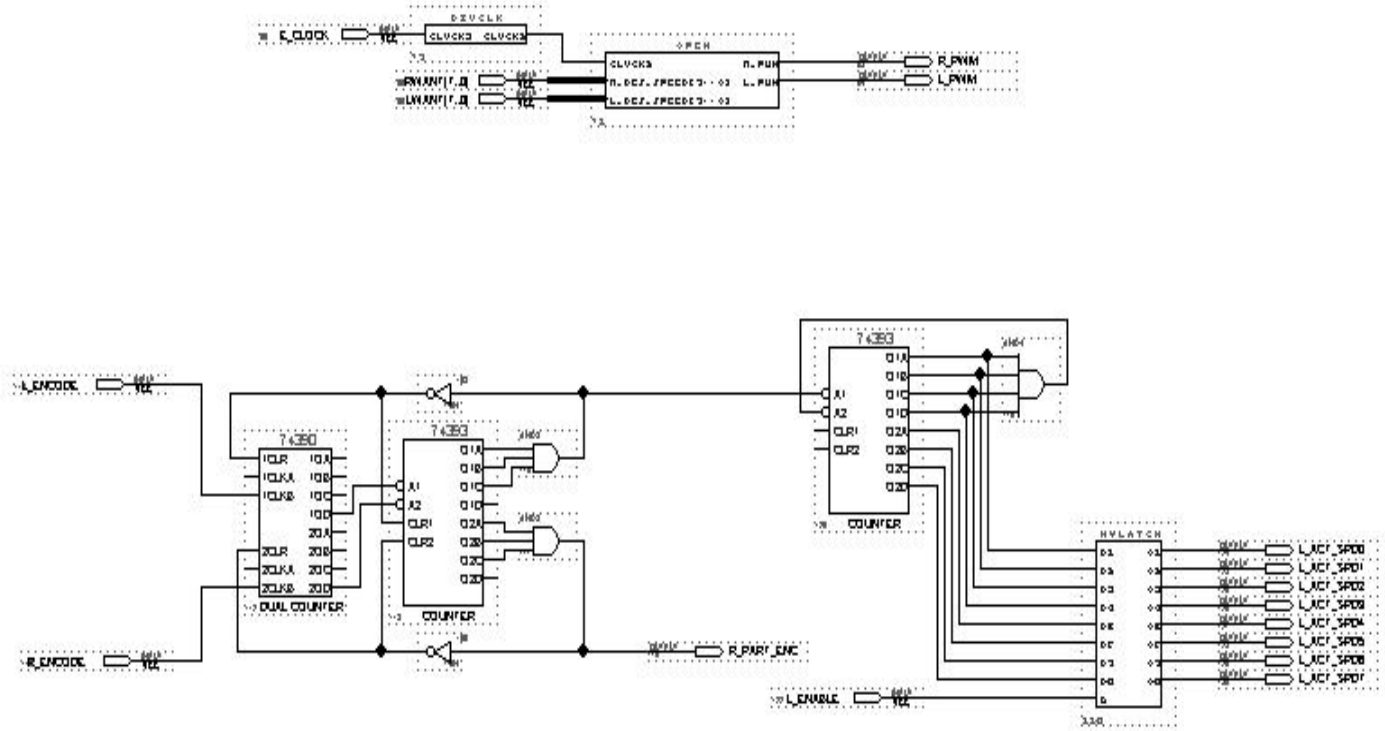


Figure 4: Second Attempt at Motor Control in Altera (Part 2)

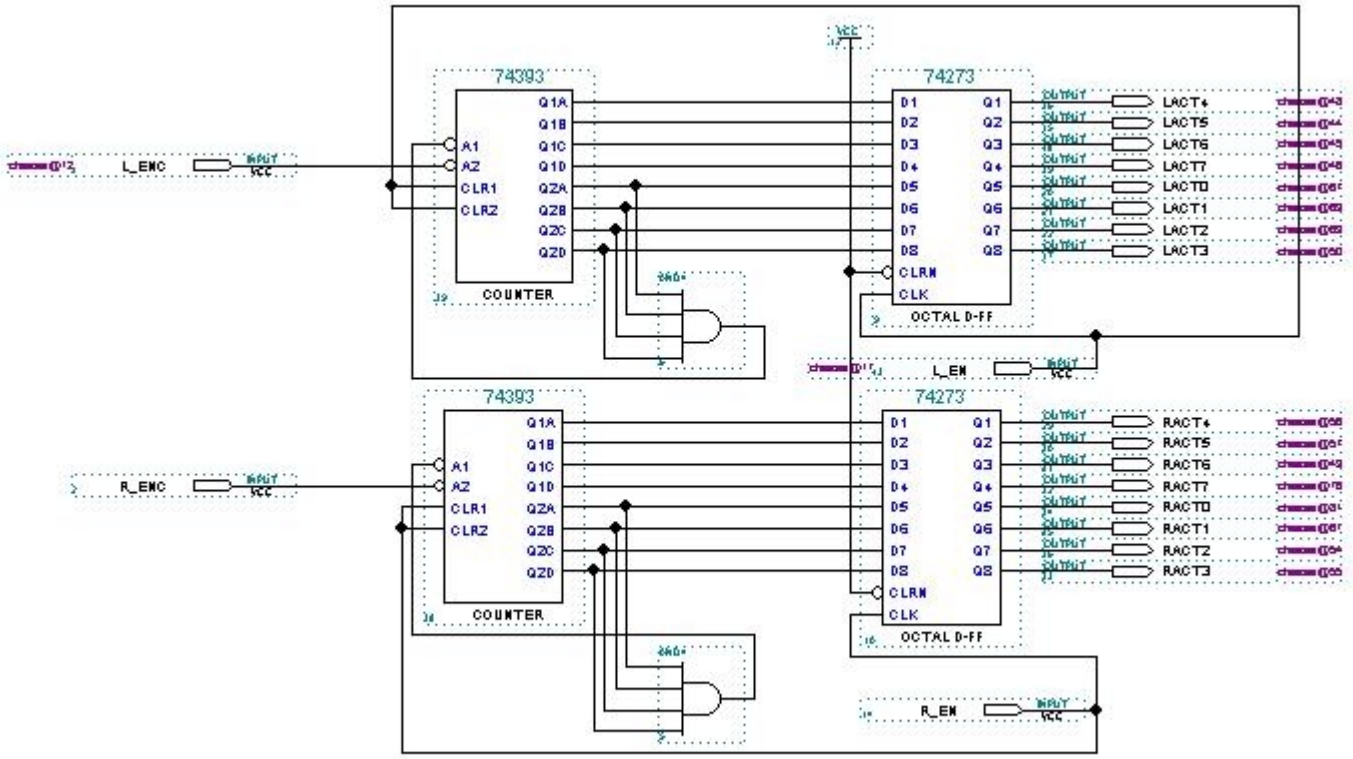
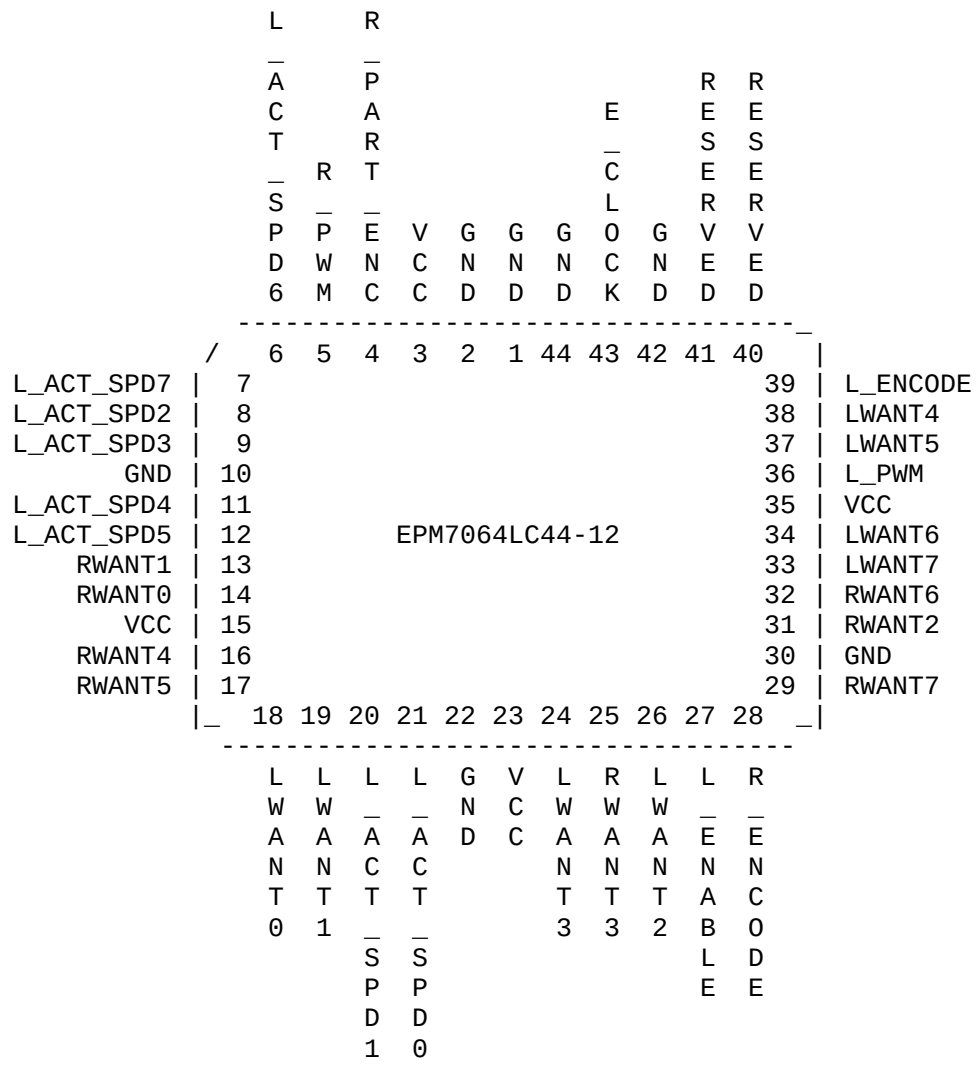
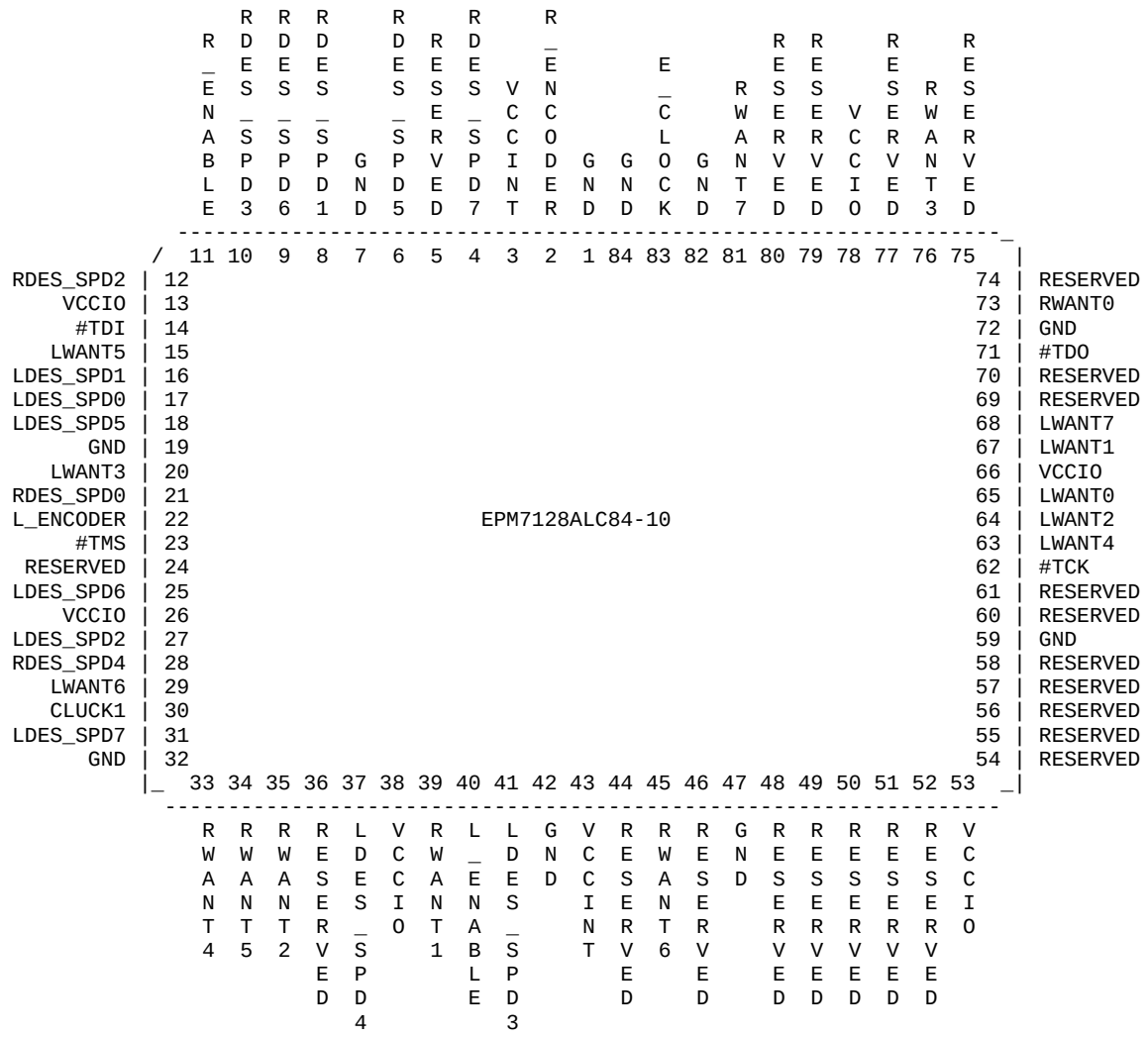


Figure 5: Encoder Signal Processing in Altera (Final Attempt)



N.C. = Not Connected.
VCC = Dedicated power pin, which MUST be connected to VCC.
GND = Dedicated ground pin or unused dedicated input, which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.

Figure 5: Pinout, First Attempt at Motor Control in Altera (Part 1)

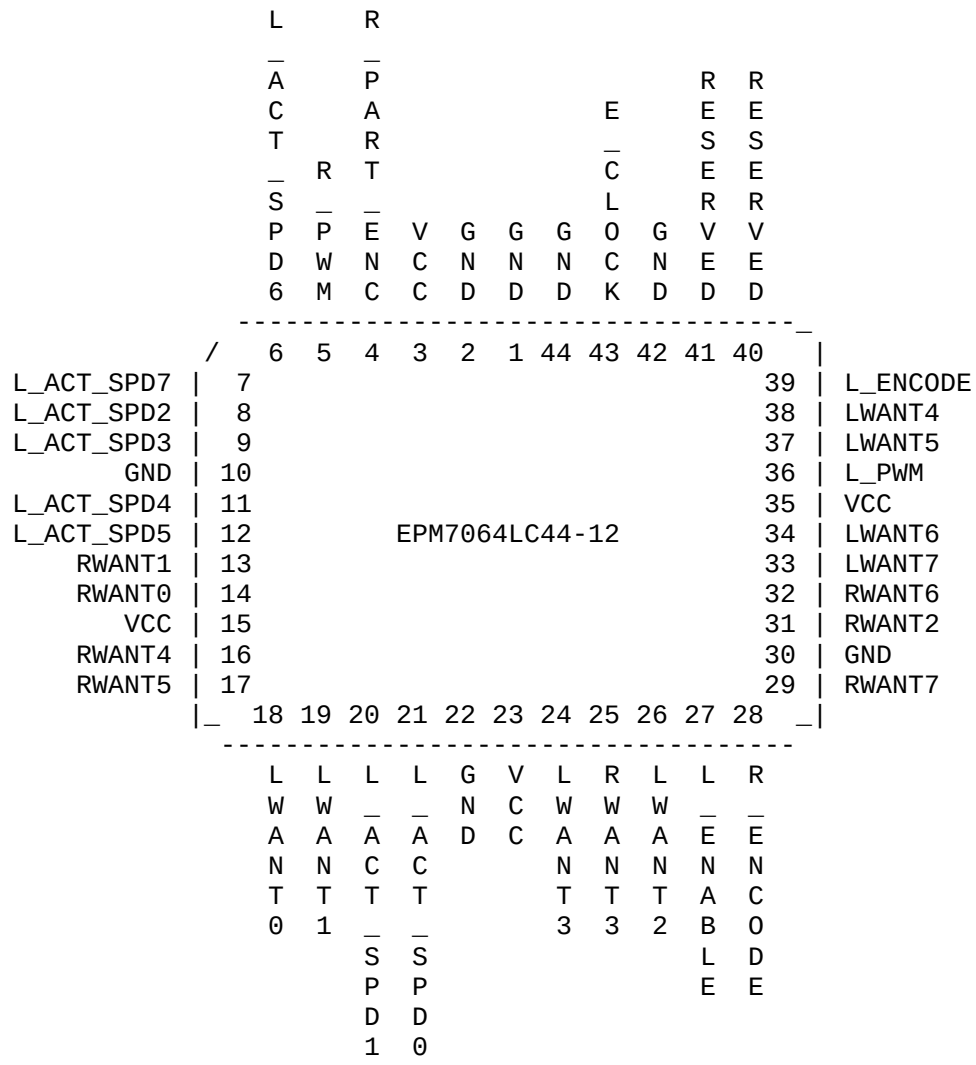


N.C. = Not Connected.
VCCINT = Dedicated power pin, which MUST be connected to VCC (3.3 volts).
VCCIO = Dedicated power pin, which MUST be connected to VCC (3.3 volts).
GND = Dedicated ground pin or unused dedicated input, which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.

Figure 6: Pinout, First Attempt at Motor Control in Altera (Part 2)

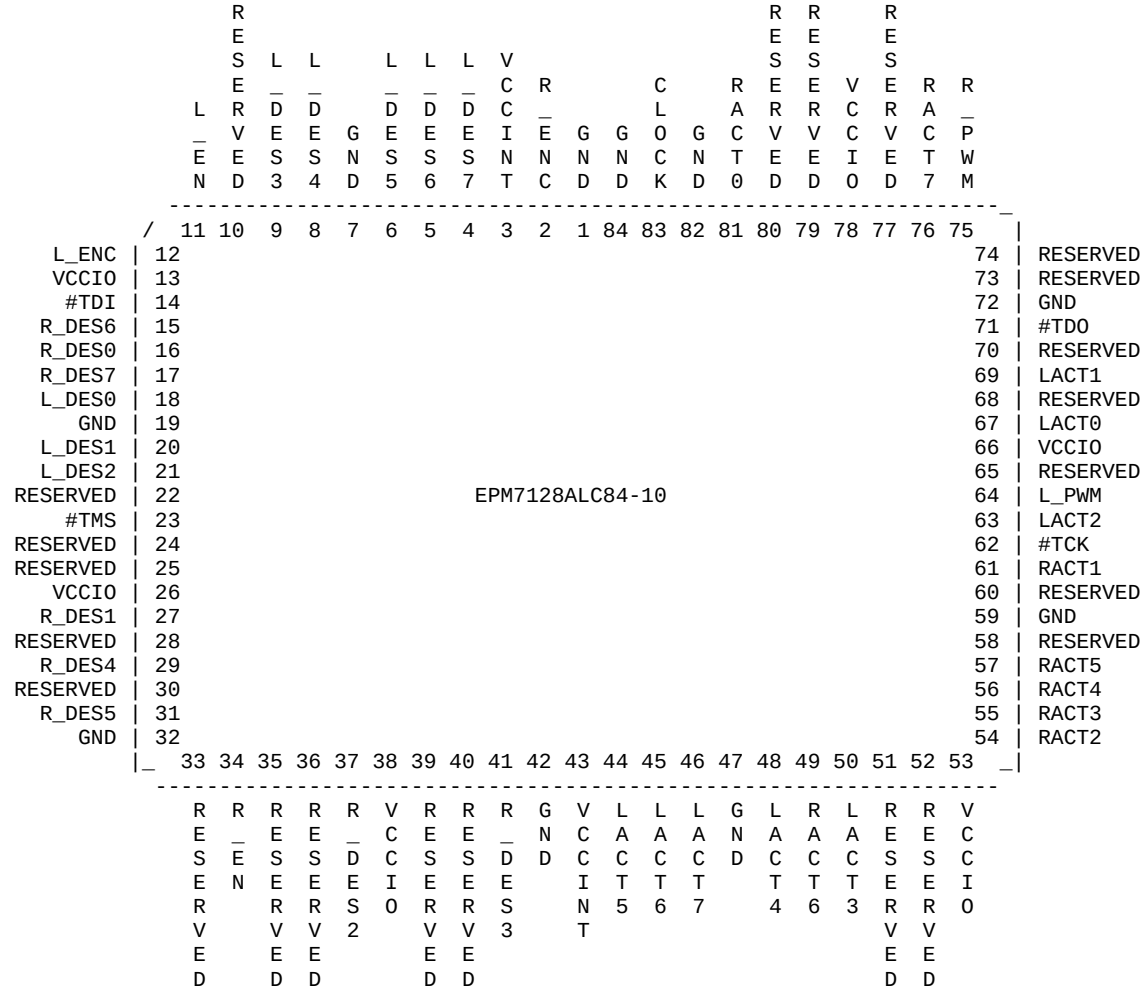
```
                   L     L     L                                            L
                   _     _     _                                            _
                R  A  R  A     A                    R     R  R  R     L  R  D
                E  C  E  C     C  R  L              _     E  E  E     _  E  E
                S  T  S  T     T  _  _  V           E     S  S  S     E  S  S
                E  _  E  _     _  W  W  C           N     E  E  E  V  N  E  _
                R  S  R  S     S  A  A  C           C     R  R  R  C  A  R  S
                V  P  V  P  G  P  N  N  I  G  G  G  O  G  V  V  V  C  B  V  P
                E  D  E  D  N  D  T  T  N  N  N  N  D  N  E  E  E  I  L  E  D
                D  7  D  1  D  6  5  2  T  D  D  D  E  D  D  D  D  O  E  D  6
              -----------------------------------------------------------------_
            /  11 10  9  8  7  6  5  4  3  2  1 84 83 82 81 80 79 78 77 76 75   |
   E_CLOCK | 12                                                              74 | L_DES_SPD4
     VCCIO | 13                                                              73 | RESERVED
      #TDI | 14                                                              72 | GND
   R_WANT0 | 15                                                              71 | #TDO
   L_WANT3 | 16                                                              70 | R_DES_SPD7
   L_WANT7 | 17                                                              69 | L_DES_SPD7
L_DES_SPD0 | 18                                                              68 | L_DES_SPD5
       GND | 19                                                              67 | RESERVED
L_DES_SPD1 | 20                                                              66 | VCCIO
L_DES_SPD2 | 21                                                              65 | L_ACT_SPD4
L_DES_SPD3 | 22                       EPM7128ALC84-10                        64 | RESERVED
      #TMS | 23                                                              63 | R_ENABLE
   L_WANT4 | 24                                                              62 | #TCK
L_ACT_SPD3 | 25                                                              61 | RESERVED
     VCCIO | 26                                                              60 | RESERVED
   R_WANT3 | 27                                                              59 | GND
L_ACT_SPD2 | 28                                                              58 | R_DES_SPD6
   R_WANT2 | 29                                                              57 | RESERVED
  RESERVED | 30                                                              56 | RESERVED
   R_WANT1 | 31                                                              55 | R_DES_SPD4
       GND | 32                                                              54 | L_ACT_SPD0
           |_  33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53  _|
             ------------------------------------------------------------------
                R  R  L  R  R  V  L  R  R  G  V  R  R  R  G  L  L  R  L  R  V
                _  _  _  _  E  C  _  _  _  N  C  E  _  _  N  _  _  _  _  E  C
                W  W  W  D  S  C  W  D  W  D  C  S  D  D  D  W  W  D  A  S  C
                A  A  A  E  E  I  A  E  A     I  E  E  E     A  A  E  C  E  I
                N  N  N  S  R  O  N  S  N     N  R  S  S     N  N  S  T  R  O
                T  T  T  _  V     T  _  T     T  V  _  _     T  T  _  _  V
                7  6  6  S  E     5  S  4        E  S  S     1  0  S  S  E
                         P  D        P           D  P  P           P  P  D
                         D           D              D  D           D  D
                         1           2              3  5           0  5


N.C. = Not Connected.
VCCINT = Dedicated power pin, which MUST be connected to VCC (3.3 volts).
VCCIO = Dedicated power pin, which MUST be connected to VCC (3.3 volts).
GND = Dedicated ground pin or unused dedicated input, which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.
```

Figure 7: Pinout, Second Attempt at Motor Control in Altera (Part 1)



N.C. = Not Connected.
VCC = Dedicated power pin, which MUST be connected to VCC.
GND = Dedicated ground pin or unused dedicated input, which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.

Figure 8: Pinout, Second Attempt at Motor Control in Altera (Part 2)



N.C. = Not Connected.
VCCINT = Dedicated power pin, which MUST be connected to VCC (3.3 volts).
VCCIO = Dedicated power pin, which MUST be connected to VCC (3.3 volts).
GND = Dedicated ground pin or unused dedicated input, which MUST be connected to GND.
RESERVED = Unused I/O pin, which MUST be left unconnected.

Figure 9: Pinout, Encoder Signal Processing and PWM in Altera

```
SUBDESIGN whip
(
      R_ACT_SPD[7..0]         : INPUT;
      L_ACT_SPD[7..0]         : INPUT;
      R_DES_SPD[7..0]         : INPUT;
      L_DES_SPD[7..0]         : INPUT;

      R_WANT_SPD[7..0]        : OUTPUT;
      L_WANT_SPD[7..0]        : OUTPUT;
)

BEGIN

      IF R_ACT_SPD[7..0] == R_DES_SPD[7..0] THEN
            R_WANT_SPD[7..0] = R_ACT_SPD[7..0];
      ELSIF R_ACT_SPD[7..0] < R_DES_SPD[7..0] THEN
            R_WANT_SPD[7..0] = R_ACT_SPD[7..0] + 10;
      ELSIF R_ACT_SPD[7..0] > R_DES_SPD[7..0] THEN
            R_WANT_SPD[7..0] = R_ACT_SPD[7..0] - 10;
      END IF;
      IF L_ACT_SPD[7..0] == L_DES_SPD[7..0] THEN
            L_WANT_SPD[7..0] = L_ACT_SPD[7..0] + 10;
      ELSIF L_ACT_SPD[7..0] < L_DES_SPD[7..0] THEN
            L_WANT_SPD[7..0] = L_ACT_SPD[7..0] + 10;
      ELSIF L_ACT_SPD[7..0] > L_DES_SPD[7..0] THEN
            L_WANT_SPD[7..0] = L_ACT_SPD[7..0] - 10;
      END IF;

END;
```

Figure 10: Closed Loop Motor Control in Altera Attempt, Program “whip.tdf”

```
/* EE 382 Group 7 Code
 * Michael Schumacher
 * Shawn McVay
 * Gerald Weed
 * Jack Landes
 *
 * Initial creation date: 7 April 1999
 * Modification date: 3 May 1999
 *
 * Purpose: To see if the HC11 can drive the robot.
 *
 * Notes:
 * This code is not complete as of yet. This code should essentially do wall
 * following. Some sensor code is written, but the fire stuff right now is
 * pseudocoded as of now.
 *
 * Order of code:
 *
 * 1. Init. variables and interrupts.
 *    a. RTI interrupt
 *    b. TOF interrupt.
 *    c. PIA expansion Ports A and B.
 *    d. A/D converter.
 *    e. TIC1 & 2 interrupt.
 * 2. Sensor code.
 * 3. Motor control code.
 * 4. Other stuff.
 */

/* Lovely #include and #define statements. */
/* Need definitions for Altera addresses and other fun things. */
#include <hc11.c>
#include "pia.h"
#define TRUE 1
#define FALSE 0
#define MAXSPD 3000
#define STRAIGHT 0.6
#define STOP 0.0
#define SLOWER 0.4
#define FASTER 0.8
#define FULLBORE 1.0
#define DELAY 300
/* These are set dependent on sensor calibration. */
#define WALLHUG 0xCA
#define WALLFAR 0x65
#define CLOSEFRONT 0xCA
#define DET_FIRE 0x30
/* These are for the Altera EPM 7128 chip. */
#define LEFTENC (*(unsigned char *) 0xB100)
#define RIGHTENC (*(unsigned char *) 0xB200)
#define LEFTPWM (*(unsigned char *) 0xB300)
#define RIGHTPWM (*(unsigned char *) 0xB400)

/* Function declarations. */
void rti_isr(void);
void tof_isr(void);
```

```
void tic1_isr(void);
void tic2_isr(void);
char smokey(void);
char kill_nine(void);
void out2bytes(unsigned int n);
void out1byte(unsigned char n);
void send_altera(void);
void recv_altera(void);
void poll_fire(void);
void poll_wall(void);
char extinguish(void);

/* Variable declarations. */
char fire_out = FALSE; /* Is the fire out? */
unsigned char rf, rr, lf; /* Sensor definitions. */
unsigned char front; /* Distance to FRONT WALL. */
char in_room = FALSE; /* Are we there yet? In a room, that is. */
char is_fire = FALSE; /* Have we found the fire yet? */
unsigned char mono_left, mono_rt, ranger; /* Fire sensor stuff. */
float l_des, r_des; /* Desired motor speed variables. */
float l_act, r_act; /* Actual motor speed variables. */
unsigned char l_enc, r_enc; /* What's coming off the encoders. */
unsigned char lencprev, rencprev; /* Last value of the encoders. */
unsigned char l_pwm, r_pwm; /* What we're sending off to the PWM. */
float Kp_left, Kp_right; /* Constant of proportionality for P motor control. */
float m; /* Conversion factor for motor speed to 8 bit Encoder speed. */
float lspdfact, rspdfact; /* Speed factor (between 0 and 1) for desired spd. */
char calc_dc_flag = FALSE; /* Do we have to check the motors? */
char check_wall = FALSE; /* Do we have to check for the walls? */
char l_rev = FALSE; /* Right and left reverse switches for the HC11. */
char r_rev = FALSE;

/* At last! Real main function! */
void main(void) {

  /* Priority stuff first, like stuff that needs to be set in the
     first 64 E-clock cycles. */
  OPTION |= 0x80; /* Fire up A/D converter. */

  /* Initialize your constants here. Primarily for motor control. */
  Kp_right = 1.0/25000.0;
  Kp_left = Kp_right;
  m = 3000.0/255.0; /* For Full speed, 0xFF -> 3000 RPM or thereabouts. */
  l_enc = 0; /* Assume the motors are going zero speed initially... */
  r_enc = 0;
  lencprev = 0; /* Safe to assume motors weren't moving beforehand either. */
  rencprev = 0;

  /* For now, also assume that for straight line going, one wants 60% of full
   * speed capacity. Modify in subroutines and functions for turns and wall
   * following algorithms.
   */
  lspdfact = STRAIGHT;
  rspdfact = STRAIGHT;

  /* Start RTI interrupt. */
```

```
RTI_JMP = JMP_OP_CODE;
RTI_VEC = rti_isr;
PACTL &= ~0x01; /* Set RTI to go every 8 ms. */
TMSK2 |= 0x40;
TFLG2 = 0x40;

/* Start TOF interrupt. */
TOF_JMP = JMP_OP_CODE;
TOF_VEC = tof_isr;
TMSK2 |= 0x80;
TFLG2 = 0x80;

/* Set up timer input captures for line sensors.
 * Set up TIC1 and TIC2 for capture on rising edge.
 * TIC1 : Front line sensor.
 * TIC2 : rear line sensor.
 */
TIC1_JMP = JMP_OP_CODE;
TIC1_VEC = tic1_isr;
TIC2_JMP = JMP_OP_CODE;
TIC2_VEC = tic2_isr;
TCTL2 = 0x18;
TMSK1 |= 0x06;
TFLG1 = 0x06;

/* Enable PIA ports A and B.
 * PIA_A and B are (for now) defined as input.
 *
 * I'm thinking right now that we can use these ports as a way to
 * trigger the interrupts for the line sensors.
 */
PIA_CRA &= ~0x04;
PIA_DDRA = 0x00;
PIA_CRA |= 0x04;

PIA_CRB &= ~0x04;
PIA_DDRB = 0x00;
PIA_CRB |= 0x04;

enable(); /* Enable interrupts and other fun things. */

/* Actual meat of the program!
 * Need to concern ourselves with the following (probably in order):
 * - Distance calculations to walls.
 * - Motor control calculations.
 *   > Adjustment for wall following.
 *   > Right or left turns? Where? How fast?
 *   > Gunning it through the maze vs. in a room.
 *   > Speeding up to catch air via the ramp (optional)
 * - GPS subsystem (optional, scores extra points with Dr. Bruder)
 * - Marshmallow subsystem. (optional)
 */
while(!fire_out){

  /* I'm going to start by looking at the sensors for wall following.
   * (Why? Because I coded it in that way)
```

```
*/
if(check_wall){

  /* Grab the latest data from the sensors. */
  poll_wall();

  /* Calculate the dist. to front wall. */
  front = (lf >> 1) + (rf >> 1);

  /* Since we are right wall following, if we get too close to the wall,
        * we need to yell at the motors to do something about it.
        * Same if we get too far away to the right wall, for fear of tagging
        * the opposite wall.
        */
  if(rr > WALLHUG) { /* Robot's heading into the left wall */
         lspdfact = FASTER;
         rspdfact = SLOWER;
         l_rev = FALSE;
         r_rev = FALSE;
  }
  else if (rf > WALLHUG) { /* Robot's heading into the right wall */
         lspdfact = SLOWER;
         rspdfact = FASTER;
         l_rev = FALSE;
         r_rev = FALSE;
  }
  else { /* Don't bother it. */
         lspdfact = STRAIGHT;
         rspdfact = STRAIGHT;
         r_rev = FALSE;
         l_rev = FALSE;
  }
  /* What happens if we get close to a front wall? Turn! */
  if (front > CLOSEFRONT) {
         /* Dependent on the direction of what's to come. If it's obvious,
          * do the required turn.
          * The comparison is a serving suggestion.
          */
         if((lf << 1) < rf) { /* Hang a left. */
           lspdfact = FULLBORE;
           l_rev = TRUE;
           rspdfact = FULLBORE;
           r_rev = FALSE;
         }
         else if ((rf << 1) < lf) { /* Hang a right. */
           lspdfact = FULLBORE;
           l_rev = FALSE;
           rspdfact = FULLBORE;
           r_rev = TRUE;
         }
         else /* Call an all stop and exit -- this is a test condition. */ {
           lspdfact = STOP;
           rspdfact = STOP;
           exit(0);
         }
  }
```

```
      /* Tell the HC11 not to do this for a while. */
            check_wall = FALSE;
    }

    /* Fun motor control stuff.
     * This is the calculation part of the program for the motor
control.
     * Things that need to be done:
     * 1. Get and calc. the actual speed from the decoders.
     * 2. Calculate the desired speed for the motors.
     * 3. Send the motor speed to the data.
     */
    if(calc_dc_flag){
      /* Grab the current speed off of the Altera chip. */
      recv_altera();

      /* Calculate actual speed. */
      l_act = m * (float) l_enc;
      r_act = m * (float) r_enc;

      /* Calculate desired speed. */
      r_des = MAXSPD * rspdfact;
      l_des = MAXSPD * lspdfact;

      /* Figure out what 8-bit value needs to be sent to the motor.
       * This needs to be figured out -- ran out of time.
       * Of course, the following two statements below are wrong.
            */
      r_des = r_des + 1;
      l_des = l_des + 1;

      /* Last thing to do is send it the new data. */
      send_altera();
      calc_dc_flag = FALSE;
    }

    /* Sound of white (line) noise. */
    if(in_room) { is_fire = smokey(); }

    /* Do we have FIRE?!?!? */
    if(is_fire) { fire_out = kill_nine(); }
  }
}

/* Fun interrupt declarations and statements. */

/* TIC1 interrupt */
#pragma interrupt_handler tic1_isr
/* TIC1_servicing:
 * 1. Tell the HC11 it's in a room.
 * 2. Exit out.
 */
void tic1_isr(void)
{
  in_room = TRUE;
  /* Clear interrupts */
```

```
  TFLG1 = 0x04;
}

/* TIC2 interrupt */
#pragma interrupt_handler tic2_isr
/* TIC1_servicing:
 * Pretty much the same right now as tic1_isr, but will be tweaked later.
 * 1. Tell the HC11 it's in a room.
 * 2. Exit out.
 */
void tic2_isr(void)
{
  in_room = TRUE;
  /* Clear interrupts */
  TFLG1 = 0x02;
}
/* TOF interrupt. */
#pragma interrupt_handler tof_isr
/* TOF servicing:
 * 1. Tell the HC11 to look at the wall sensors.
 * 2. Exit out of the interrupt.
 */
void tof_isr(void)
{
  check_wall = TRUE;
  /* Exit nicely out of the interrupt. */
  TMSK2 |=0x80;
}

/* RTI interrupt. */
#pragma interrupt_handler rti_isr
/* Function of RTI interrupt:
 * Yell at the HC11 to do motor control calculations.
 */
void rti_isr(void)
{
  calc_dc_flag = TRUE;
  /* Exit nicely out of the interrupt. */
  TFLG2 = 0x40;
}

/* Yet more function declarations. */

/* poll_wall:
 * This function polls the wall sensors.
 */
void poll_wall(void)
{
/* Set up ADCTL in order to start conversion.
   * Poll pins AD0-3 once and STOP. Thus CC = 0.
   */
  ADCTL = 0x10;

  /* Wait for the nice conversions to stop.
   * Poll CCF continuously to see if it's done.*/
  while(!(ADCTL & 0x80)) ;
```

```
  /* Send values on their way. Remember right wall following.
   * ADR1 is Right Rear sensor.
   * ADR2 is Right Front sensor.
   * ADR3 is Left Front sensor.
   * ADR4 is currently not in use.
   */
  rr = ADR1;
  rf = ADR2;
  lf = ADR3;
}

/* poll_fire:
 * This function polls the fire sensors. *
 */
void poll_fire(void)
{
 /* Poll upper three A/D pins. These are for fire detection.
   * Scan four at once and STOP, *for now*. CC = 1 for fire sensors. */
  ADCTL = 0x14;
  while (!(ADCTL & 0x80)) ; /* Wait for those A/D to finish. */

  /* Fire sensors are located as follows:
   * ADR1 = monopulse left
   * ADR2 = monopulse right
   * ADR3 = ranger
   * ADR4 is empty.
   */
  mono_left = ADR1;
  mono_rt = ADR2;
  ranger = ADR3;
}

/* send_altera:
 * What's goin down:
 * Fire off the motor controller speeds to the Altera chip.
 * We're going to need the addresses of the Altera chip to do any more with
 * this.
 */
void send_altera(void)
{
  LEFT_MOTOR = l_pwm;
  RT_MOTOR = r_pwm;
}

/* recv_altera:
 * First store the last values off of the encoder.
 * Then grab the latest motor control figures off of Altera.
 */
void recv_altera(void)
{
  lencprev = l_enc;
  rencprev = r_enc;

  l_enc = LEFT_ENCODER;
  r_enc = RT_ENCODER;
```

```
}

/* smokey:
 * Like Smokey the Bear, only you can prevent fires.
 * Here is the code to detect if a fire is in a room.
 * It returns TRUE or FALSE if it found it.
 */
char smokey(void)
{
  char fire = FALSE; /* Is there a fire? */
  int i = 0; /* Loop crap. */

  /* Call all_stop on the motors. */
  lspdfact = STOP;
  rspdfact = STOP;

  /* Enable some sort of scanning procedure here.
   * This could either be making the bot go 360 or enabling a
   * small motor to rotate the sensors.
   */

  /* DISABLE THE TOF INTERRUPT HERE so it doesn't freak out. */
  TMSK2 &=~0x80;

  /* Find anything? There will be some set fire levels present. */
  if((mono_left > DET_FIRE) || (mono_rt > DET_FIRE) ||  (ranger > DET_FIRE))
    fire = TRUE;

  /* AT this point, reenable the TOF interrupt */
  if(!fire) TMSK2 |= 0x80;

  /* Return fire status. */
  return(fire);
}


/* kill_nine:
 * Just like UNIX, this code directs the bot to the fire and kills it.
 * Outline:
 * 1a. Direct the robot towards the fire.
 * 1b. Move robot to fire.
 * 2. STOP when the robot gets inside the ring.
 * 3. Extinguish the fire if you really, really find it.
 *    Otherwise, go full bore out of the room.
 */
char kill_nine(void)
{
  char is_out = FALSE;

  while(is_fire) {
    /* Direct the robot to the fire. Use Monopulse system. */
    if(mono_left > mono_rt) {
      lspdfact = 0.4;
      rspdfact = STRAIGHT;
    }
    else if(mono_rt > mono_left){
```

```
      rspdfact = 0.4;
      lspdfact = STRAIGHT;
    }
    else {
      lspdfact = STRAIGHT;
      rspdfact = STRAIGHT;
    }

    /* Is the fire detected? */
    if(ranger > 0xCA){
      while(!is_out){
            lspdfact = STOP;
            rspdfact = STOP;
            is_out = extinguish();
      }
  else {
    is_fire = FALSE;
  }
 }
}
}

char extinguish(void)
{
  char out = FALSE;
  char trigger;

  while (!out){
    trigger = TRUE; /* trigger extinguisher. */
    poll_fire();
    if(ranger < DET_FIRE){
      out = TRUE;
      trigger = FALSE;
    }
  }
  return(out);
}

/* Enter your debugging functions here. */

/* out2bytes: print an integer to the terminal. */
void out2bytes(unsigned int n)
{
  asm("  jsr 0xffc1");
  asm("  jsr 0xffc4");
}

/* out1byte: prints a single byte character to the terminal. */
void out1byte(unsigned char n)
{
  asm("  jsr 0xffbb");
}
```

Figure 11: HC11 Control Code (Final Revision)

Appendix B: Itemized Budget for Group 7

Description	Cost	Donated? (Y/N)
Starter Kit (Provided by EE Dept.)	$0.00	Y
Available Items (Provided by EE Dept.)	$0.00	Y
4 - IR Emitters	$7.52	N
84 Pin PLCC Socket	$1.15	N
EPM7128ALC84-12	$16.00	N
EPM7032LC44-12	$12.60	N
84 Pin Wire Wrap Adapter	$9.86	N
2 - GP2D12 Sensors	$11.78	N
March Stockroom	$0.44	N
April Stockroom	$0.20	N
Mounting Hardware	$3.68	Y
Masonite, Allthread, and Hardware	$16.64	Y
Perforated Board	$2.79	Y
Water Pump	$5.00	Y
Small Diameter Tubing (4ft)	$0.68	Y
Large Diameter Hose (4ft)	$0.80	Y
Hose Adapters (2)	$1.30	Y
Sprinkler Head	$0.89	Y
Auxiliary Batteries (2)	$9.36	Y
Total	$100.69	
Total (Undonated)	$59.55	
Total Budget Available	$100.00	
Total Budget Unused	$40.45	