Table of Contents
and List of Figures

Abstract

This semester, we were assigned to develop a fully-functional firefighting robot to compete in the Trinity College Firefighting Home Robot Contest and the NM Tech local competition as well. The robot's main objective is to navigate through a maze without errors (i.e. touching a wall, etc.), locate the candle, put it out with a fan, and return home successfully. Ramps and furniture placings were optional. This year there was an option to perform an arbitrary start (where you can start from anywhere in the maze rather than the home circle). This robot is autonomous and NOT remote controlled, as well as possessing closed loop speed control, wall-following control (sense to be not dead-reckoning), Altera motor speed, and optical isolation between the high-power and low-power electronics.

Introduction

The first series of classes (roughly 2-3 weeks) were spent learning about various strategies in approaching the design. During this time, we thought of various ways of designing our robot. The few weeks afterwards, we started collecting and ordering the components for our robot. The very first task we had to accomplish was gathering as many components as possible and actually physically designing the robot. It all started with the actual mounts. Most of the robot was held together by screws, nuts, bolts, etc. We were able to mount everything with virtually little problem (including moving the motors in more to move the wheels under the robot). Once we finished that (with some additions and changes along the way), we had to develop the necessary circuit boards for the components that required assembly as well as testing each components' functionality

(H-bridge, sensors, etc.). During this whole time, we realized the HC12 immediately needed to be hooked up, that way we can start coding right away because that was the beef of the project (the brains). We needed to make sure it does not hit walls, detect a flame, put it out, go home, etc. All this we had to do efficiently and quickly as possible due to the time constraint (we had to have most of it functionally working by mid-term 2 months to accomplish it).

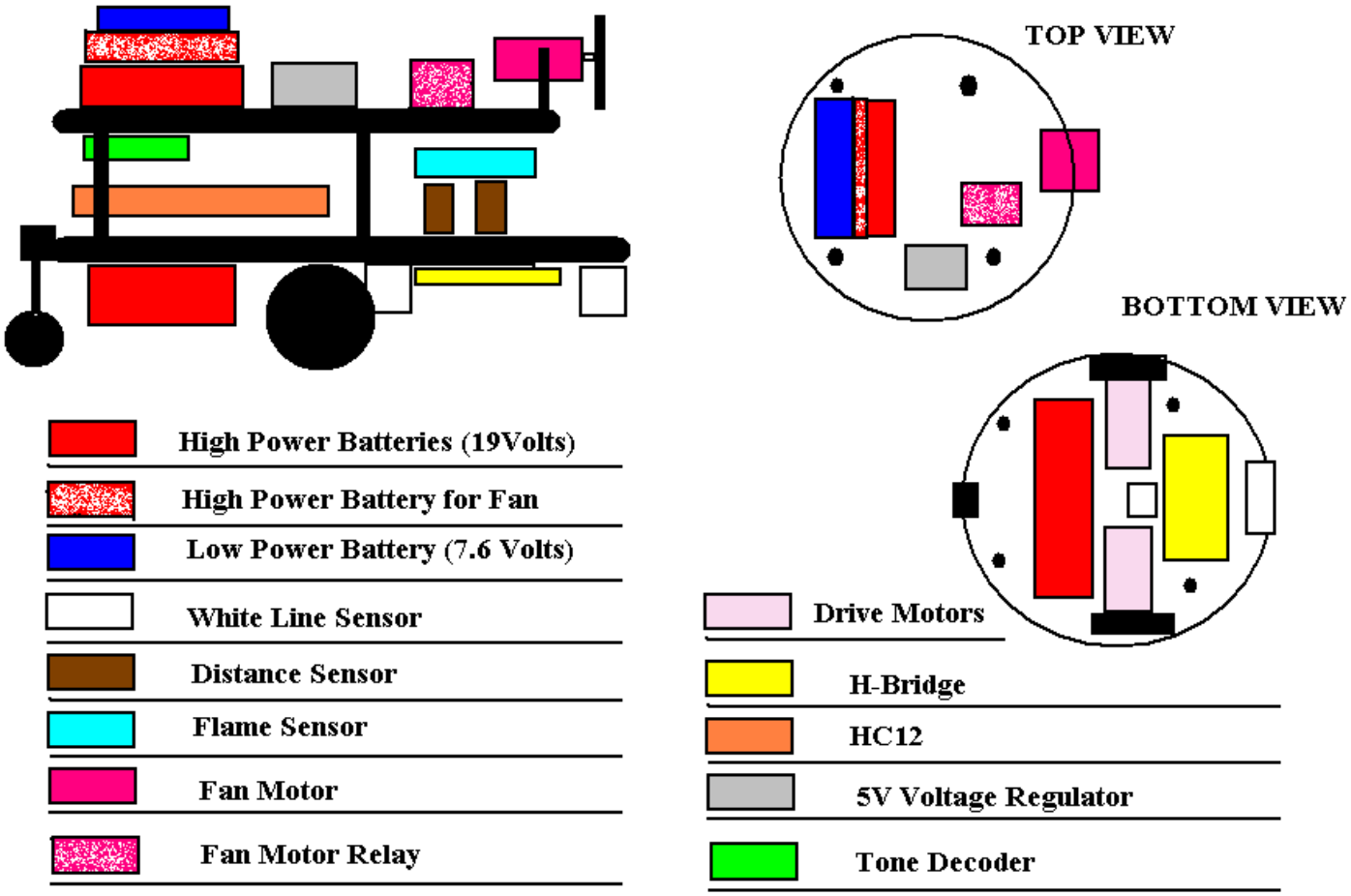

Figure 1

The System Layout

We used two 1/8" thick aluminum circular plates. The plates were placed as shown in Figure 1. We placed all the analog devices in the middle of the two plates to reduce noise form the drive motors among other distractions. All of the subsystems (Distance sensors, white line sensors, etc…) were connected with one-way connectors. We also used shielded cable for all high power wiring and analog controls.

The power for our robot came from three different sources. For high power we used three 6V lead acid batteries. For the low power we use an old 7V-cell phone battery. Finally we had a 9.6V NiCd battery used solely for our fan.

Subsystems

The Tone Decoder

Obviously the robot would not start without hearing a tone (the "house" is on fire - hence, similar to a fire alarm). This was actually optional, but it was worth bonus points and it would be a more realistic situation for a robot to start instead of the customary pressing of buttons to start it. Our tone decoder was like a handheld buzzer in a medicine bottle. Of course, there was a tone decoder circuit and we must employ the buzzer itself near the microphone (from the circuit) as close as possible in order for the robot to start. We did not worry about other outside tones affecting our robot, since our tone decoder triggered only when the tone had to be very close to the microphone itself. This tone decoder employed a 3.5 kHz tone in order to start the robot. This tone would produce a digital signal, sending it to the HC12 (the microcontroller used to control the robot), and thus letting the robot know it is time to go.

The Distance Sensors

The main purpose of the distance sensors was to detect walls for certain distances. Hence, we programmed the robot to avoid these walls via the distance sensors' readings. The sensors we decided to use were the Sharp GP2D120's. These sensors are applicable to the infrared spectrum section; hence other light sources would cause very little disruptions. These specific sensors had a good readable range from 4cm to 30cm. We were able to get these relatively short-range sensors to work for us.

The White Line Detector

There were white lines instituted in the entranceway of every room in the maze, as well as a white semicircle placed underneath the candle. We used this as some sort of detection that it was an entranceway to a room with the flame possibly being in there. The white line sensors were able to tell us whether we entered or exited a room. Furthermore, it would inform us of how close we are to the candle. So we decided to construct our white line detector subsystem with a red and white LED on the proper circuit board and a good protective cover to block out the extra outside light. In the early stages while on a black surface, the white line detector read .5-. 6V and while on a white surface, it shot up to 4.5V. We wanted a more precise reading (0V and 5V), so we ran the signal coming from the LED's through a Schmidt trigger to give us a close-to-zero output as possible (final reading was only in the millivolts) and close-to-five output as possible (5.01V being the closest). The perf board also consisted of 1k resistors that gives us a black/white reading difference of roughly .5-1 in. In addition, we added 2

miniature light bulbs with a 1k pot to adjust the intensity. This is so that in case of a dim or a darkened room, we need ambient lighting to help the white line detector subsystem detect the white lines. This board with the lights was mounted on the bottom of our bottom-half of the robot. This was done so it would catch a white line with the greatest of probability.

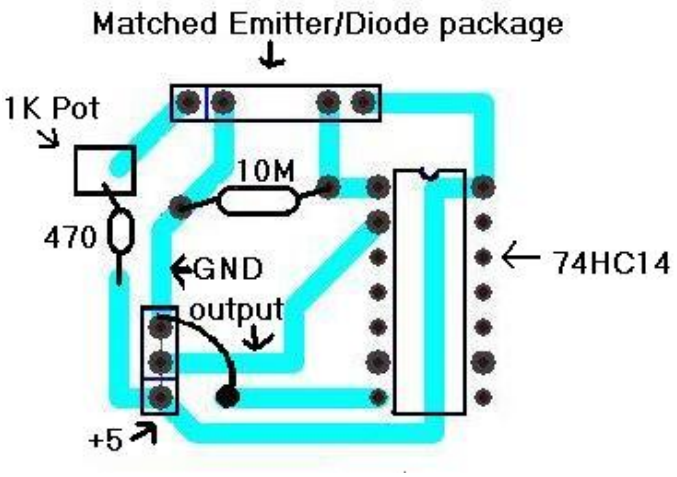

Figure 2: White line Sensor

The Fire Sensor

Since our main goal is to extinguish a flame, the fire sensor was one of the more crucial elements to our robot. For the possible detection of a flame, we opted for two PN168 phototransistors. We knew that outside light would prove to be a detriment to our mission of detecting the flame. So we constructed a pie shape shield, for our two phototransistors, out of a black acrylic sheet. We mounted the two phototransistors side-by-side with a blinder between them to give us binocular vision. Each phototransistor (depending on which one was closer to the flame) would result in a 5V (4.85-4.97)

reading when pointed directly at the flame and completely to 0V or a few millivolts when pointing completely away from the flame. About half the power (~2V) is detected about 35 degrees away from binocular vision. A perf-board was made to connect the phototransistors to the HC12. The perf-board has two 100k resistors connected to ground, a 5V input, and 2 output wires. The goal was for our flame sensor to enter a room and do a 180 degree turn looking for a significant voltage spike. This signal ran to the A/D ports on the HC12.

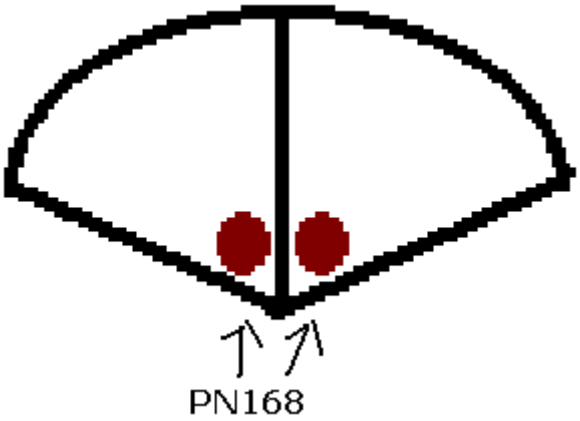

Figure 3: Flame Sensor

The Fan Relay

In order for our HC12 to be safely separated from all of our high power systems, we used a relay to control the fan. We used an Antex Electronics 0DC-01 relay. When the HC12 sends a 5V signal to the relay, it turns on the fan. When no signal is sent to the relay, the fan is off.

H-Bridge

In order to for our robot to move with any controlled direction we needed to construct an H-Bridge control system. We decided to use the National Semiconductor

LMD18200 H-Bridge chip. We chose this h-bridge because it has great current-handling ability, which was needed for our 11W drive motors. The only disadvantage of this H-Bridge is that they need two external bootstrap capacitors and we need two of these expensive H-Bridges for our design. The H-Bridges were fairly simple to interface to the HC12 because they accept the 5V PWM from the HC12 and needed only a single logic line to control their direction. We also used optoisolators with this design to help separate the HC12 and the powerful H-Bridges.

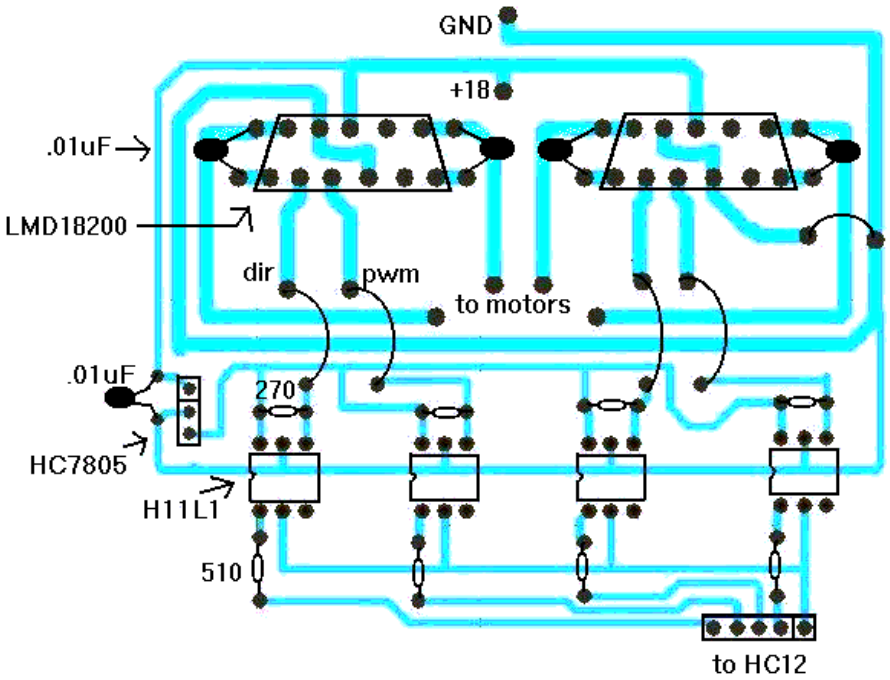

Figure 4: H-bridge

Drive Motors

When we started this project, we all agreed that speed would win in competition. This brought us to the decision to pick a motor with high speed and torque. The motors we finally decided to use for our robot were the Maxon 11W motors. We chose these motors because of the high torque (18 mNm) and speed they can deliver. The 10:1

gearhead help the motors give us the speed we desire. The disadvantage the motors possess are a starting current of 3.2A. Other than the high starting current, the motors are nice and reliable.

5V Regulator

The voltage regulator we chose to use was the low dropout LM2940 voltage regulator. We choose this voltage regulator because it has internal short circuit protection and good low voltage behavior. The disadvantage is that the regulator needs a 22uF or bigger capacitor between ground and output. Overall, this low dropout voltage regulator worked great.

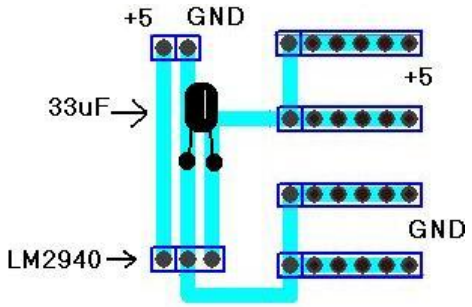

Figure 5: Voltage Regulator

The Circuit Layout

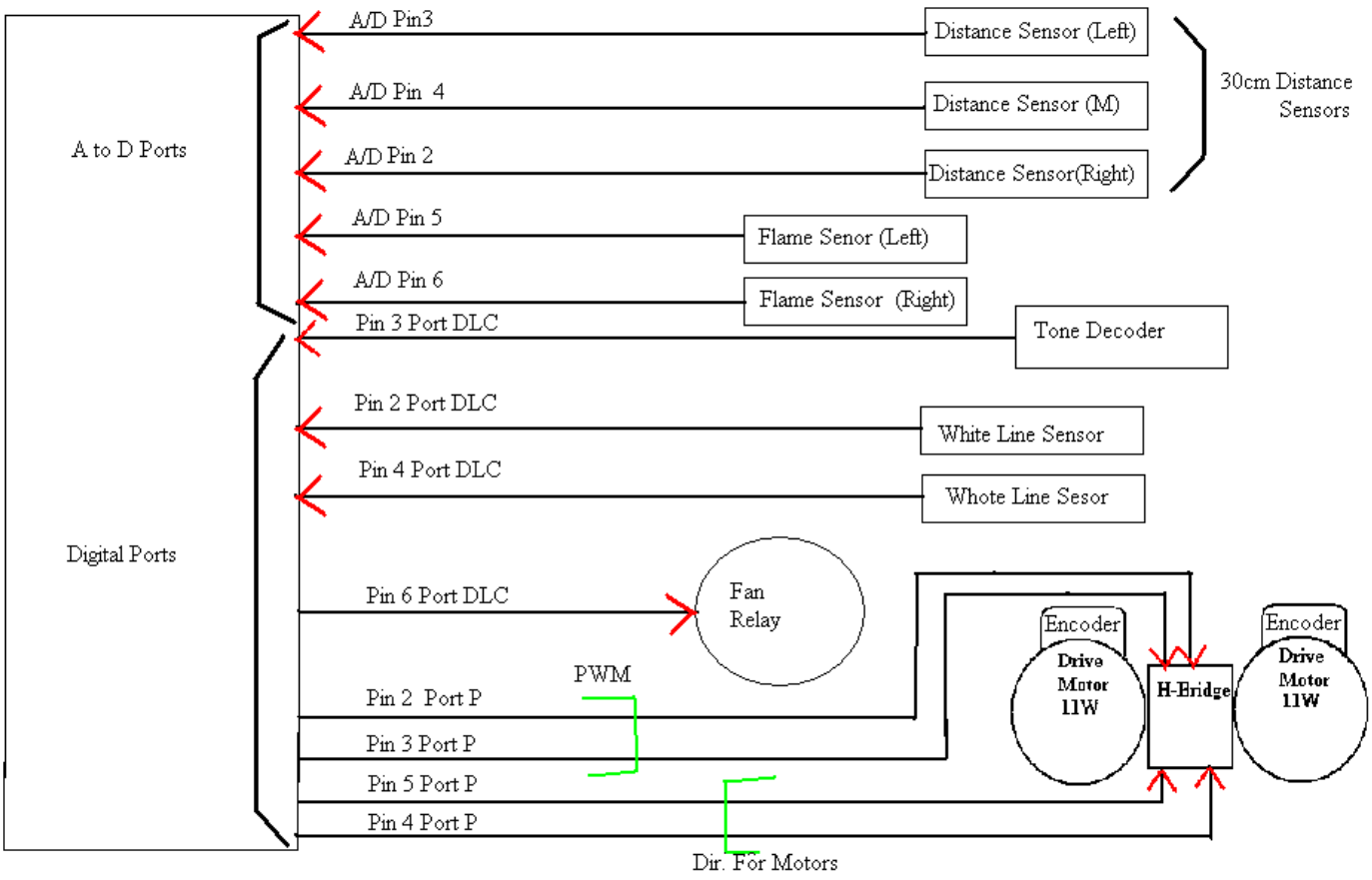

Figure 6

The HC12/The Software

The Motorola 68HC12 microcontroller (HC12 for short) is what one would call the brain of the robot. It was virtually the CPU of the robot. Due to its limited memory, we had to apply a memory expansion board for it (constructed by us from a previous EE 308 class). Once we had our robot mostly physically built, we spent the rest of the time (and pretty much the majority) programming it to execute its functions well. No doubt, the software is considered the toughest aspect to master and it took up well over half of the class' duration.

We decided to keep it simple and apply a left wall follow routine for searching the rooms. The left wall follow routine is used to navigate the maze. If just using the left wall follow routine, the robot will circle around the maze going into each room, except room 4. In the final code, the robot left wall follows until it sees a white line, and then the white line function is called. The wall follow routine uses a proportional controller based on the distance sensors to make sure the robot does not collide or touch the wall in any way (time adage).

Right wall follow is basically the same thing as left wall follow except the right wall sensor is used instead of the left and a few constants were adjusted. Right wall follow is used to enter room 4 and also to return home from room 1. To enter the island room, room 4, we had to switch to right wall follow, but we could not do that until we were close to room 4. What we did was wait until our front sensor picked up which happened right before the robot arrived at room 4.

Integrated into both left and right wall follow is a center sensor function to avoid

running straight into the wall. If the center sensor detects a wall, the robot will do a turn in place until the robot is facing to the right if left wall following or to the left if right wall following.

If a white line is detected, the robot will enter a white line function. The white line function basically does a room scan. It first turns in place to the right and checks to see if the flame sensors are above a set threshold. If the sensors pick up a value higher than the threshold then the value is stored and the counter is stored as well. If another value is seen that is higher than the previous value then it is stored and the counter is stored as well. When the scan has finished, the robot will turn to the right and it will return to the highest value it picked up using the counter. This should be the candle. If nothing is detected in the right turn, the robot will turn to the left by twice as long as the right so that it returns to the center and then goes on to the left by the same distance and was traversed on the right. The same sensor scan will occur, with the highest values being stored. If there isn't a candle in the room, the white line function will turn the robot around and exit the room and then resume right wall following.

If there is a candle in the room the code enters a function called flame follow. In the flame follow function, the robot is turned to the location of the highest sensor reading. The code then enters a flame follow loop. The flame follow loop uses the two flame sensors to orient the flame and point directly at the candle and proceed in that direction. It is a basic proportional controller based on the two flame sensors. If the robot gets too close to a wall either left, right, or center, then the robot will enter either left or right wall follow. This is to keep the robot from hitting walls as it approaches the candle.

Inside the flame follow function the robot will again check for a white line. If it senses another white line, it knows that it is near the candle and it will enter the extinguish function. The extinguish function basically turns on the fan and goes a little turn to the right and a turn to the left to make sure the candle is extinguished. After the candle is put out, the extinguish function will direct the robot on which return home routine function to enter. This is based on the room count, if the room count is one then it knows to go into the room one return home function, if the room count is two go to return home for room two and so on and so forth for all four rooms.

Returning home from room one is simple, just right wall follow until you see a home circle. The only complications we ran into were the quarter circles they were using at Trinity, we had to adjust our code for that since our robot thought it was on the home circle as soon as it put out the candle. This is due to the fact that we are using two white line sensors to detect the home circle, one in front and one in back. The robot knows it is on the home circle when both sensors are high. To fix the problem we just had the robot only wall follow for a set amount of time, then start checking for the home circle. This worked out very well.

To return home from room 2 we had to do some strange coding. If you just left wall follow, you will enter room 3 and thus lose the return trip points. If you just right wall follow, you will enter room 1 and also lose your points. So what we did was have the robot basically follow whichever wall was present. If there is a left wall, follow it, otherwise right wall follow. This seemed to work fairly well, but we had problems with the robot getting out of the room. Because of our problem getting the robot out of the

room we never got the return home function from room 2 completely functional. It would work about 50% of the time.

Returning home from room 3 was very simple. All you have to do is left wall follow and you are there. We had a delay on checking for the home circle and after that it would pick up the home circle and that was it. It worked almost flawlessly.

Returning home from room 4 lead to a few complications. We had trouble getting the robot to pick up the home circle when it right wall followed out of room 4. What we did was have the robot switch to left wall following after it left room 4. That fixed the problem.

Conclusion

We achieved our goal of developing a quick, fully-functional firefighting autonomous robot. We were able to navigate through the maze with ramps as obstacles (with the motor's immense power, we were able to easily go over it without noticing it). We were not able to add furniture to the maze. This is due to the fact that we did not want to take the chance of using this feature, only to have the furniture directly in front of the flame, preventing any light for our flame sensor to detect. We were able to return home with the greatest of ease as well (with the exception of room 2). It is a relatively simple, compact, and cheap design (we were not allowed to spend over $300 and we barely went over the $200 mark) as we decided to keep it simple and not too complicated of a design. With our robot, achievement was evident as we captured 6th place in the Trinity College Firefighting Robot Contest out of 62 robots. Overall, we were happy with our results and benefited the value of teamwork. During the semester, we lost a

teammate for specific reasons and it threw us off-track for a few weeks. If anything, more time would have been more beneficial. Just a few more weeks can give us the needed time to fine-tune the program. It is a debate about whether or not more or less people in a group would benefit. The more the people, the more we can learn from each other; but, at the same time it could also cause laziness to a few. The fewer the people in a group, the more that motivates each member to work extra harder; however, it can also cause burnout. Whatever the case, it was an overall joyous experience.

Appendix A

The Code

```
/* Group 2 */

#include "stdio.h"
#include "math.h"
#include "hc12.h"
#include "DBug12.h"

unsigned char rduty;       /*  Duty Cycle For Right Motor */
unsigned char lduty;       /*  Duty Cycle For Left Motor  */
volatile int prduty;       /*  % Duty Cycle For R Motor   */
volatile int plduty;       /* % Duty Cycle For L Motor    */
int s_des_r;       /*  Right Motor Desired Speed  */
int s_des_l;       /*  Left Motor Desired Speed   */
volatile int s_mon_r;      /*  Right Motor Monitored Speed  */
volatile int s_mon_l;      /*  Left Motor Monitored Speed   */
volatile int dir_r;        /*  Right Motor Desired Direction  */
volatile int dir_l;        /*  Left Motor Desired Direction  */
volatile int k_drive;              /*  Drive Constant          */
volatile int k_wall;             /* Wall Constant */
int k_wall_r;                    /* Right Wall Constant */
volatile int dd;          /* Distance desired */
int s_des;                       /* Speed Desired */
volatile int dsl;          /*  Left Distance Sensor   */
volatile int dsr;          /*  Right Distance Sensor  */
volatile int dsc;          /*  Center Distance Sensor   */
volatile int line;
volatile int k_center;           /*  Constant for Center Sensor */
int tone;                        /*  Tone Flag */
int flame = 0;                   /*  Flame Flag  */
int locr = 0;                    /* Location right sensor sensed flame  */
int locl;                        /* Location left sensor sensed flame  */
int stored_l;                    /* Stored counter value for left side */
int stored_r;                    /* Stored counter value for right side */
int i = 0;                       /* i counter flag */
int k= 60;                       /* K constant  */
int roomc = 0;                   /* room count flag */
int rwf = 0;                     /* right wall follow flag */
int turnspeed;                   /* Speed at which to turn in loops */
int init = 0;                    /* initialization flag */
int inff = 0;                    /* initialization flag for flame following */
int fsrr = 0;                    /* fire sensor right adjusted value */
int h = 0;                       /* h counter */
```

```
int rightside = 0;                  /* rightside flame sensor flag to tell which side the flame is
on */
int diffs= 0;                       /* difference of flame sensors for flame following */
int home = 0;                       /* home flag to say when we are at the home circle */
int linecnt = 0;            /* line Count Flag  */
int getout = 0;                     /* Flag to tell when out of room 2 */
int line2 = 0;                      /* Second Line Sensor Flag */

void main()
{
   setup();                         /* Runs Setup Function to set up HC12 */
   while ( tone == 0) {             /* wait for Tone and do nothing until it is recieved */
         if ( PORTDLC & 0x08)
                  tone = 0;
         else
                  tone = 1;
         RPWM = 250;
         LPWM = 250;
         DBug12FNP->printf("wait for tone /n ");
         }
         i = 0;
         while ( i < 8 ) {
                  gofwd();
                  i = i + 1;
         }
         i = 0;

   while (1){
         /* White line sensor Setup */
         if (PORTDLC & 0x04)          /* check for white line and set flag if present */
                  line = 0;
         else
                  line = 1;

         /* Check for white line */
         if  (line){
                  if (flame == 1)
                           extinguish();  /* if there is a line & flame is set jump to extiguish */
                  else
                  whiteline();  /* else jump to the white line routine to scan room. */
         }
         if ( (dsl > 145) && (dsr > 145))        /* code to slow the robot down on shrp turns
*/
                  s_des = 175;
         else
                  s_des = 125;
```

```
        if ( inff == 1){
                flamef();               /* if the flame follow routine is intitialized flame
follow */
        }
        else{
        if ( rwf == 1){                 /* if rwf flag is set run right wall following code */

                if (init == 0){
                while ( dsr > 125 ){    /* while there is a front wall turn to right as if
leftwall following */
                        leftwall();
                        init = 1;
                }
                i = 0;
                while ( i < 1){
                        leftwall();             /* leftwall follow for two cycles so that the
robot doesn't turn and go down the maze */
                        i = i + 1;
                }
                gofwd();
                i = 0;
                }

                rightwall();                    /* right wall follow into room four */
        }
        else
                leftwall();                     /* as long as rwf flag isn't set leftwall follow
around the maze */

        }
        }

}

//*********************************************************************
******//
//************************   SETUP FOR HC12
*********************************//
//*********************************************************************
******//
void setup()
{
   /* PORT DLC STUFF FOR WHITELINE SENSOR AND TONE DECODER */
        DDRDLC = 0x40;
        PORTDLC = 0x00;
```

```
  DDRP = 0xff;        /* PortP output  */
  PORTP = 0x00;       /* Set port P to 0 for strt */
  dd = 114;           /* distance desired */
  s_des = 125;        /* speed desired */
  k_wall = 47/16;     /* K constant for left wall */
  k_wall_r = 47/16;   /* K constant for right wall */
  k_center = 28;      /* K constant for center */
  k_drive = 1/8;      /*  Set Drive Constant */
  turnspeed = 150;    /* speed for turn in place functions */

  /* A/D Setup For Distance Sensors */
  ATDCTL2 |= 0x80;            /* A/D Power Up */
  ATDCTL3 = 0x00;         /* Continue Conversions In Active Background Mode */
  ATDCTL4 = 0x01;         /* 8Mhz max P clock, 2Mhz min */
  ATDCTL5 = 0x70;         /* Convert All channels 8 times continuosly  */

  /*  Choose 8-bit mode  */
  PWCLK = PWCLK & ~0xc0;
  /*  Choose left-aligned */
  PWCTL = PWCTL & ~0x08;
  /*  Polarity, High at begining, Low when duty count is reached  */
  PWPOL = PWPOL | 0x0f;
  /*  And Clock mode S0 for Channels 1,0 and S1 for Channels 2,3 */
  PWPOL = PWPOL | 0xf0;

  /*  Set up for M and N */
  PWCLK = 0x07;
  PWSCAL1 = 0x02;

  /*  Select period of 250 for Channels 1 and 0 */

  PWPER1 = 249;         /*  PWPER1 + 1 */
  PWPER0 = 249;         /*  PWPER0 + 1 */
  PWPER2 = 99;            /* PWPER2 + 1 */
  /*  Enable PWM on Channels 2, 1 and 0 */

  PWEN = 0x07;
  DBug12FNP->printf("PWM Enabled");
    /*  Set duty cycle to zero to start off */
    RPWM = 0x00;  /* right duty cycle on Channel 1 */
    LPWM = 0x00;  /* right duty cycle on Channel 0 */
       /* Set dyty cycle for PWM2 to %50 */
       PWDTY2 = 50;
       tone = 0;
       flame = 0;
}
```

```
//*************************************************************************
*************//
//************************  Left Wall Following
************************************//
//*************************************************************************
*************//

void leftwall()
{

        /* A/D Converter Control Loop */
        dsr  = 150 - ADR2H;          /* Use PAD0 for Left Sensor */
        dsl  = 150 - ADR3H;          /* Use PAD1 for Right Sensor */
        dsc  = ADR4H;            /* Use PAD2 for Center Sensor */

        s_des_r = s_des - (dd - dsl) * k_wall ;  /* right speed for left wall following */
        s_des_l = s_des + (dd - dsl) * k_wall ;  /* left speed for lefr wall following */

        if (s_des_r > 250)               /* check for overflows */
                s_des_r = 250;
        if (s_des_l > 250)
                s_des_l = 250;

        if (PORTDLC & 0x04)                  /* check for whitelines */
                line = 0;
        else
                line = 1;


    /* PWM Closed Loop Control */
        prduty = s_des_r + (s_des_r - RMS)* k_drive;
        plduty = s_des_l + (s_des_l - LMS) * k_drive;
        if ( dsc <= k_center)   /* if no front wall do this */
        {
      RPWM = prduty;                    /* RPWM is set in HC12.h so that I didn't have to
remember which port was for right wall */
      LPWM = plduty;
        PORTP = 0x00;
        }
        else                        /* else turn in place to get away from wall */
        // Check for a front wall and turn in place if there is one present */

        {
                while ( dsc >= k_center ){      /* while there is a front wall turn right */
                        PORTP = 0x2F;
```

```
                        RPWM = 175;
                        LPWM = 175;
                        dsc  = ADR4H;
                        if (roomc == 3)                 /* room count must equal 3 (must
have checked three rooms )*/
                                rwf = 1;        /* set RWF flag since we see the wall beside
room 4 */
                }

        }
}
//*************************************************************************
***************//
//****************************** White Line Function
*************************************//
//*************************************************************************
***************//



void whiteline (){
        gofwd();        /* go forward a couple of times to get into the room */
        gofwd();
        i = 0;
        roomc = roomc + 1;    /* increment the room counter */
        if (roomc == 4)
        special();              /* if it is room 4 go into the special function */

        while (i < k){          /* while loop that scans for fire  turning right*/
                PORTP = 0x2F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i +1;
                rightside = 1;
                if (fsr > stored_r){      /* if there is a greater value from the Right flame
sensor */
                        stored_r = fsr; /* then store that value and store the value of the i
flag so you can return */
                        locr = i;
                        }
                else if (fsl > stored_l){       /* same a right snesor above except it is for
left side */
                        stored_l = fsl;
                        locl = i;
                        }
        }
```

```
        if ( stored_l > 64){
                flame = 1;              /* if the stored value is above a threshold then there
is a flame present */
                line = 0;
                flamef();               /* go into flame follow mode */
        }
        else if ( stored_r > 22){
                flame = 1;
                line = 0;
                flamef();
        }

        i = 0;
        while ( i < 2*k) {              /* while loop that acans for fire turning left twice as
long as right */
                PORTP = 0x1F;                   /* same basic loop as above checks to see if
above threshold */
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i +1;
                rightside = 0;
                if (fsr > stored_r){
                        stored_r = fsr;
                        locr = i;
                        }
                else if (fsl > stored_l){
                        stored_l = fsl;
                        locl = i;
                        }
        }
        if ( stored_l > 64){

                flame = 1;
                line = 0;
                flamef();
        }
        else if ( stored_r > 22){
                flame = 1;
                line = 0;
                flamef();
        }

        else{           /* if no flame detected then turn around and go out of the room */
        turnr();
        turn180();
        gofwd();
```

```
        gofwd();
        gofwd();

        }
        line = 0;
}


//*****************************************************************************
*****************//
//****************************** Flame Follow Function
*************************************//
//*****************************************************************************
*****************//


void flamef()
{
int l;
        inff = 1;
        /* A/D Converter Control Loop */
        dsr  = 150 - ADR2H;          /* Use PAD0 for Left Sensor */
        dsl  = 150 - ADR3H;          /* Use PAD1 for Right Sensor */
        dsc  = ADR4H;            /* Use PAD2 for Center Sensor */



        l = 0; /* initialization stuff to turn the robot back to the highest senor value */
        if (h == 0){
        if ( rightside == 1){
                        if ( stored_r > stored_l){
                        while (l < locr){
                                PORTP = 0x2F;
                                RPWM = turnspeed;
                                LPWM = turnspeed;
                                l = l + 1;
                                DBug12FNP->printf("loop2");
                        }
                        l = 0;
                }
                else {
                        while (l < locl){
                                PORTP = 0x2F;
                                RPWM = turnspeed;
                                LPWM = turnspeed;
                                l = l + 1;
```

```
                        DBug12FNP->printf("loop3");
                }
                l = 0;
        }
    }
    else {
        if ( stored_r > stored_l){
            while (l < locr){
                PORTP = 0x1F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                l = l + 1;
                DBug12FNP->printf("loop4");

            }
            l = 0;
        }


        else{
            while (l < locl){
                PORTP = 0x1F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                l = l + 1;
                DBug12FNP->printf("loop5");

            }
            l = 0;
        }
    }
    h = 1;
    PORTP = 0x00;

    if ( dsl < 108)
        leftwall();     /* left wall follow if to close to left wall */
    else if ( dsr < 112)
        rightwall();    /* right wall follow if to close to right wall */
    else if ( dsc > k_center){
        while ( dsc >= k_center ){      /* turn in place to right if there is a front
wall */
            PORTP = 0x2F;
            RPWM = 175;
            LPWM = 175;
            dsc  = ADR4H;
        }
```

```
        }
        else{           /* otherwise flame follow */
        fsrr = 45 + fsr;
        diffs = fsrr - fsl;         /* difference between the two flame sensors */
        if (diffs >199)
                diffs = 199;    /* check for overflow */
        if (diffs < -199)
                diffs = -199;
        s_des_r = 150 + (diffs * 1/2);  /* right speed for left wall following */
        s_des_l = 150 - (diffs * 1/2);  /* left speed for lefr wall following */
        LPWM = s_des_r;
        RPWM = s_des_l;

        }
}

//*****************************************************************************
****************//
//****************************** Extiguish Function
*************************************//
//*****************************************************************************
****************//


void extinguish (){
        PORTDLC = 0x40;             /* turn on the fan */
        sturnr();                   /* turn in place to the right then left */
        sturnl();
        sturnl();
        sturnr();
        sturnr();
        sturnl();
        PORTDLC = 0x00;             /* turn off the fan */
        line = 0;
        DBug12FNP->printf("loop5");
                if (roomc == 1)
                        roomuno();      /* if room count is 1 go to return from room one
routine */
                else if (roomc == 2)
                        roomdose();     /* if room count is 2 go to return from room two
routine */
                else if (roomc == 3)
                        roomtrece();    /* if room count is 3 go to return from room three
routine */
                else if (roomc == 4)
                        roomquatro();  /* if room count is 4 go to return from room four
```

```
routine */



        while(1){       /* otherwise if room count is over 4 just stop */
        RPWM = 250;
        LPWM = 250;
        }
}


void sturnr()
{
        i = 0;
        while ( i < k/2 ){              /* short right turn in place */
                PORTP = 0x2F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i + 1;
        }
        i = 0;

}

void sturnl()
{
        i = 0;
        while ( i < k/2 ){              /* short left turn in place */
                PORTP = 0x1F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i + 1;
        }
        i = 0;

}


//*****************************************************************************
**********//
//*********************************  Turn Right Function
*********************************//
//*****************************************************************************
**********//
```

```
void turnr()
{
        i = 0;
        while ( i < k ){ /* Right turn in place */
                PORTP = 0x2F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i + 1;
        }
        i = 0;

}

//*************************************************************************
***************//
//********************************** Turn Left
*****************************************//
//*************************************************************************
***************//


void turnl()
{
        i = 0;
        while ( i < k ){/* Left turn in place */
                PORTP = 0x1F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i + 1;
        }
        i = 0;
}

//*************************************************************************
***************//
//********************************** Turn 180
*****************************************//
//*************************************************************************
***************//


void turn180()
{
        i =0;
        while ( i < k - 40 ){     /* Turn 180 degrees in place */
                PORTP = 0x2F;
```

```
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i + 1;
        }
        i = 0;
}

//*************************************************************************
**************//
//********************************** Go Forward
*************************************//
//*************************************************************************
**************//


void gofwd()
{
        i = 0;
        while ( i < (k/2 - 25 )){  /* go forward */
                PORTP = 0x0F;
                RPWM = turnspeed;
                LPWM = turnspeed;
                i = i + 1;
        }

        i = 0;
}


//*************************************************************************
**************//
//********************************** Right Wall
*************************************//
//*************************************************************************
**************//

void rightwall()
{

        /* A/D Converter Control Loop */
        dsr  = 150 - ADR2H;          /* Use PAD0 for Left Sensor */
        dsl  = 150 - ADR3H;          /* Use PAD1 for Right Sensor */
        dsc  = ADR4H;            /* Use PAD2 for Center Sensor */

        s_des_r = s_des + ((dd + 3) - dsr) * k_wall_r ;  /* right speed for left wall
following */
```

```
        s_des_l = s_des - ((dd + 3) - dsr) * k_wall_r ;  /* left speed for lefr wall following
*/

        if (s_des_r > 250)
                s_des_r = 250;
        if (s_des_l > 250)
                s_des_l = 250;



    /* PWM Closed Loop Control */
        prduty = s_des_r + (s_des_r - RMS)* k_drive;
        plduty = s_des_l + (s_des_l - LMS) * k_drive;
        if ( dsc <= k_center)
        {
      RPWM = prduty;
      LPWM = plduty;
        PORTP = 0x00;
        }
        else
        // Check for a front wall and turn in place if there is one present */

        {
                while ( dsc >= k_center ){
                        PORTP = 0x1F;
                        RPWM = 175;
                        LPWM = 175;
                        dsc  = ADR4H;
                        DBug12FNP->printf("FRONT  = %d\r\n", dsc);
                }

        }
        DBug12FNP->printf("RIGHT L Speed  = %d\t ,R Speed  = %d\r\n"
,plduty,prduty);

//      DBug12FNP->printf("L SPEED  = %d\t ,R Speed  = %d\t, distance = %d\r\n"
,s_des_l,s_des_r,dsr);
}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////// Room One ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

void roomuno()
{
```

```
	i = 0;
	while ( i < 100 ){	/* right wall follow to get off of circle around candle */
		rightwall();
		i = i + 1;
	}
	i = 0;
	while ( home == 0){

	/* White line sensor Setup */
	if (PORTDLC & 0x10)	/* check for white lines on both white line sensors
when both are high we are home */
		line2 = 0;
	else
		line2 = 1;

	rightwall();
	if (PORTDLC & 0x04)
		line = 0;
	else {
		line = 1;
		linecnt = linecnt + 1;
	}
	if ( line && line2)
		home = 1;
	if (line == 0)
		linecnt = 0;


	}
		while (1){ /* when at home stop */
		RPWM = 250;
		LPWM = 250;
	}

}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////// Room Two ////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

void roomdose()
{
	s_des = 175;	/* slow down the robot */
	i = 0;
	while ( i < 200){	/* leftwall follow for a while to get off the circle around
```

```
candle */
                leftwall();
                i = i + 1;
        }
        i = 0;
        while ( home == 0) {  /* while not on home circle switch between walls that are
available */
        if (PORTDLC & 0x10)         /* if no leftwall right wall follow */
                line2 = 0;
        else
                line2 = 1;

        if (PORTDLC & 0x04)
                line = 0;
        else {
                line = 1;
                linecnt = linecnt + 1;
        }
        if ( (linecnt > 2 ) && line2)
                home = 1;
        if (line == 0)
                linecnt = 0;
        if ( dsc > k_center){
                while ( dsc >= k_center ){
                        PORTP = 0x2F;
                        RPWM = 175;
                        LPWM = 175;
                        dsc  = ADR4H;
                        DBug12FNP->printf("FRONT  = %d\r\n", dsc);
                }
        }
        if ( dsr < 140) {
                rightwall();
        }
        else
        leftwall();
        }
        while (1){                  /* stop when at home */
                RPWM = 250;
                LPWM = 250;
        }

}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////// Room Three //////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
```

```
void roomtrece()
{
        i = 0;
        while ( i < 100){
                leftwall();       /* left wall follow to get off the circle around the candle */
                i = i + 1;
        }
        i = 0;
        while ( home == 0){   /* while not at home left wall follow */
        leftwall();
        if (PORTDLC & 0x10)
                line2 = 0;
        else
                line2 = 1;

        if (PORTDLC & 0x04)
                line = 0;
        else {
                line = 1;
                linecnt = linecnt + 1;
        }
        if ( linecnt && line2)
                home = 1;
        if (line == 0)
                linecnt = 0;


        }
        while (1){                /* stop at home */
                RPWM = 250;
                LPWM = 250;
        }
}


////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////// Room Four ///////////////////////////////////
////////////////////////////////////////////////////////////////////////////////

void roomquatro()
{
        i = 0;
        while ( i < 60){                  /* right wall follow to get off of the white circle and
out of the room */
                rightwall();
```

```
            i = i + 1;
        }
        i = 0;
        while ( home == 0){
        leftwall();                     /* Once out of the room left wall follow home */
        if (PORTDLC & 0x10)
            line2 = 0;
        else
            line2 = 1;
        if (PORTDLC & 0x04)
            line = 0;
        else {
            line = 1;
            linecnt = linecnt + 1;
        }
        if ( line && line2)
            home = 1;
        if (line == 0)
            linecnt = 0;
        }
        while (1){      /* when home stop */
            RPWM = 250;
            LPWM = 250;
        }


}

void special()  /* function to go into room four with out a scan since we have already
checked the other rooms */
{
        gofwd();
        gofwd();
        gofwd();
        gofwd();
        gofwd();
        line = 0;
        while (line == 0){
            rightwall();
        if (PORTDLC & 0x04)
            line = 0;
        else
            line = 1;
        }
        extinguish();   /* when there is a white line start up the extinguishing */
}
```

Appendix B

The Final Budget

Item Description	price/ea	quantity	cost	free
H-BridgeLMD 18200T	$14.00		$0.00	1
H-Bridge (National)LMD 18201T	$12.00		$0.00	
H-Bridge (Thompson)L298N	$8.00		$0.00	
H-Bridge (Allegro)UDN2998W	$7.00	2	$14.00	
H-Bridge (Thompson) L293D	$6.00		$0.00	
			$0.00	
Battery Holder: 8-AA holder	$2.00		$0.00	
Battery Holder: G6007	$0.50		$0.75	
Project enclosure: Large	$3.00		$0.00	
Project enclosure: Medium	$2.50		$0.00	
Project enclosure: Small	$2.00		$0.00	
			$0.00	
Perf-board 3.2”X4.2”:	$3.00	1	$3.00	
Perf-board 2.5”X3.2”:	$2.00		$0.00	
Perf-board 2.25”X1.8”:	$1.25		$0.00	
Perf-board 1.5”X1.75”:	$1.00	1	$1.00	
			$0.00	
Freq-voltage converters: LM2907 or 17	$2.50		$0.00	
Tone decoder: NE567	$1.00		$0.00	1
Lock 2-pin connector	$0.50		$0.00	
DC to DC convertor	$17.00		$0.00	1
Schmitt trigger optoisolators	$1.50	4	$6.00	2
5V regulators: 7805	$0.75	1	$7.00	
			$0.00	
Terminal housing (with pins) 2-pin (pair):	$0.75	3	$2.25	
Terminal housing (with pins) 3-pin (pair):	$1.00	2	$2.00	
Terminal housing (with pins) 4-pin (pair):	$1.25	3	$3.75	
Terminal housing (with pins) 5-pin (pair):	$1.75	3	$5.25	
Terminal housing (with pins) 6-pin (pair):	$2.50	2	$5.00	
			$0.00	
3.4kHz 9V buzzer:	$1.50		$0.00	
3/16” shrink tubing (price per inch)	$0.10	10	$1.00	
3/32” shrink tubing (price per inch)	$0.10	10	$1.00	
4-wire 26agw multi-strand wire (price/ft)	$1.00		$0.00	
IR distance sensors:0-30cm: GP2D120:	$8.00	6	$48.00	
IR distance sensors:0-80cm: GP2D12:	$10.00		$0.00	
10” diam. 1/16” (thick) Al. disks:	$7.00		$0.00	
12”x12” diam. 1/16” (thick) Al. plate:	$7.00		$0.00	
			$0.00	
Pittman 28-Watt Motor:	$40.00		$0.00	
Maxon 11-Watt motor:	$50.00	2	$100.00	
Maxon 6-Watt 22mm motors (green body):	$50.00		$0.00	
Maxon 5-Watt motor:	$35.00		$0.00	
Maxon 6-Watt 22mm motors (grey body):	$60.00		$0.00	
			$0.00	

Rechargable 7.2V NiCad pack:	$10.00		$0.00
12V 2.0Ahr lead battery:	$30.00		$0.00
12V lead-acid chargers:	$9.00		$0.00
Rechargeable Alkaline batteries: Rayovac	$1.00		$0.00
9V Alkaline batteries:	$2.00		$0.00
9-Volt battery connector	$0.50		$0.00
			$0.00
4-wire 26AGW multi-strand wire unshielded:/ft	$1.00		$0.00
4-wire 26AGW multi-strand wire shielded:/ft	$1.25		$0.00
			$0.00
Fuse holder (GMA fuses):	$1.00	1	$1.00
GMA fuses(1,2,4,6A):	$0.20	2	$0.40
0.5A sub-miniature fuses:	$1.00		$0.00
			$0.00
Solder-tail DIP socket2x3-pin:	$0.20		$0.00
Solder-tail DIP socket2x4-pin:	$0.25		$0.00
Solder-tail DIP socket2x7-pin:	$0.30		$0.00
Solder-tail DIP socket2x8-pin:	$0.30		$0.00
Wire-wrap DIP socket2x4-pin:	$1.00		$0.00
Wire-wrap DIP socket 2x7-pin:	$1.25		$0.00
Wire-wrap DIP socket2x8-pin:	$1.50		$0.00
30-pin SIP socket	$1.50		$0.00
10-pin male header:	$0.20		$0.00
20-pin male header	$0.50		$0.00
IDC connectors (female)2X5	$0.70		$0.00
IDC connectors (female)2X8	$0.70		$0.00
IDC connectors (female)2X10	$1.00		$0.00
Boxed headers2X5-pin (male)	$1.00		$0.00
Boxed headers2X8-pin (male)	$1.50		$0.00
4-pin audio locking conn. male-straight:	$0.50		$0.00
4-pin audio locking conn. male-right-ang.:	$1.00		$0.00
4-pin audio locking conn. female:	$1.50		$0.00
2x10 (female) header housing:	$2.00		$0.00
crimp contacts: (i.e. pins for above)	$0.10		$0.00
Electret mic (no-leads):	$1.00		$0.00
Electret mic (with-leads):	$1.50		$0.00
Polarized single-row housing:3-pin housing:	$0.50		$0.00
Polarized single-row housing:4-pin housing:	$0.60		$0.00
Polarized single-row housing:5-pin housing:	$0.70		$0.00
Polarized single-row housing:6-pin housing:	$0.80		$0.00
			$0.00
100uF cap 6.3V:	$0.25		$0.00
Ferrite noise-filters:	$1.00		$0.00
Female 2.1mm charger plug	$2.00	1	$2.00
Male 2.1mm charger plug	$2.00	1	$2.00
Antex solid-state opto-isolated2.5A DC relays:	$5.00	1	$5.00
Reflective IR emitter/detector pair:	$1.50	2	$3.42
			$0.00
1k Ohm 10-turn pot:	$0.50		$0.00
10k Ohm multi-turn pot:	$1.50		$0.00
1-turn pots500 Ohm	$0.50		$0.00

1-turn pots1k Ohm	$0.50	3	$1.50	
1-turn pots 5k Ohm	$0.50		$0.00	
1-turn pots10k Ohm	$0.50		$0.00	
1-turn pots 75k Ohm	$0.50		$0.00	
1-turn pots100k Ohm	$0.50		$0.00	
			$0.00	
Misc. capacitors:	$0.10	5	$0.50	
Misc. LEDS	$0.10		$0.00	2
Misc. switches:	$0.50	1	$0.50	
			$0.00	
Misc. spacers/standoffs 4-40 spacers:	$0.10		$0.00	
Misc. spacers/standoffs 6-32 spacers:	$0.10		$0.00	
Misc. spacers/standoffs 8-32 spacers:	$0.10		$0.00	
			$0.00	
1.5” rubber wheels:	$2.50		$0.00	2
Caster wheel assembly	$10.00		$0.00	1
			$0.00	
velcro C15S3J	$1.20	3 ft.	$3.60	
metal spacer: GS111	$0.10	4	$0.40	
cable strap C15S4B	$2.00	1 ft.	$2.00	
16-Amp connectors G7017	$0.12	8	$0.96	
broken drill bit	$0.50	2	$1.00	
small connectors (3-pin) C4753H	$0.30	4	$1.20	
MICA4 connectors			$0.00	2
HC12			$0.00	3
Duracell 3V Li battery			$0.00	3
fan			$0.00	2
6V battery			$0.00	3
misc. resistors			$0.00	12
white line sensor HoA708-1 8847			$0.00	2
battery holder			$0.00	1
acrylic sheet			$0.00	1
light bulb			$0.00	2
GRAND TOTAL			224.73	

Fire Sensor:	$0
White Line Detector:	$3.42
Wall-Following Platform:	$48.00
Motor Control:	$100.00
5V Regulator:	$7.00
H-Bridge:	$20.00
Fan Relay:	$5.00
Miscellaneous Parts:	$41.31
GRAND TOTAL:	**$224.73**

The budget was not supposed to exceed $300. So we realized in the beginning to gather used, donated, free parts as much as possible. Virtually every subsystem

mentioned above possessed an etched circuit board with various components. Those costs fell under the miscellaneous parts. In addition, these costs also include the parts that we accidentally “fried” at one point (although we didn’t burn too many parts). We thought our final budget was around $208, but we failed to take in consideration all the parts that malfunctioned along the way, adding an estimated extra $16. The fire sensor subsystem consisted mainly of the 2 phototransistors and the acrylic sheet. It was basically donated to us (hence the $0). The white line detector subsystem consisted mainly of 2 reflective IR emitters in the beginning, but later changed to 2 LED's. We incorporated 2 Schmidt triggers and 2 light bulbs with 2 1k pots totaling up to $3.42. The wall-following platform is basically 3 IR distance sensors with the applicable capacitors totaling up to almost $50.00. Evidently, the motor control provided over half of our budget. The two Maxon 11W motors alone costed $100. Along with the 2 H-Bridges and 4 optoisolators among other miniscule parts, it ran up to $120 (motors plus the H-Bridge and optoisolators). The 5V regulator costed $7.00. The fan relay, which consisted mainly of the fan itself and the relay motor, totaled $5.00. The miscellaneous parts (excluding what was already mentioned earlier in this paragraph) includes various fuses, switches, nuts, bolts, screws, wheels, batteries, velcro, crimps, heat-shrink wire, etc. which ran a little over $40.00. Overall, we accomplished our mission of not going over the $300 budget.