We’re Just What We Seem….

Two Man Team

Brandon Lattimore
Ginny Martin

Table of Contents

ABSTRACT:

The functional purpose of this project was to build an autonomous, fire-fighting robot capable of competing in the Trinity Competition. The educational purpose was for us to learn to make decisions that we would have to adhere to and base the rest of our project on. We learned an incredible amount of what now seems like common sense. Most of all, this project gave us a sense of accomplishment. In January, we were positive that we would never be able to build a robot. Now we both feel that we could build one alone in just a couple of months. We also learned the value of teamwork. Every group in our JD room contributed to our success in one way or another. We held our composure in the most trying of circumstances and accepted it when OUR brilliant idea was discarded. Most of all, we learned to take a chance. If something didn't work, we twisted and re-wired and tried until it did. We developed unique solutions to late night problems and did our jobs with what we were given. This was our first feat in engineering.

Chassis

Our robot's body looked crude for a number of reasons. We initially planned to do a preliminary lay-out on round aluminum plates we got from the scrap bin so that we wouldn't have extra holes in our finished product.

Plexi-glass: Someone donated sheets of plexi-glass to us. We thought that would be good because it is lightweight and it just looks so darn cool! Then, with a bit of thought, we decided to go against plexi-glass because it is clear and therefore would not block out any ambient light and we thought our white line sensor probably needed all the ambient light blocking we could give it. We then thought that we could either spray paint the plexi-glass or cover it with tin foil or something. Too much work. Besides, we didn't know how to use machine tools to cut the plexi-glass into circles anyway.

Aluminum plates: We decided to stick with the circular aluminum plates because they were conveniently lying in the free bin. We first planned on using two plates, one for the base and another for an elevated level. This would prove to make working with the HC-12 difficult and add unneeded extra weight, so we just used one plate for the base.

Elevated plate: We did have an elevated plate that was only about one-fifth of a circle and we mounted the fan, relay, and fire sensors onto it. Mounting our components proved to be more work that we had anticipated. We put the elevated portion together with mismatched spacers and hardware from our homes. It was a trial and error process that we would rather not have undone. Once we were all mounted, secure, and running

properly we decided not to mess with what was already working. We mounted the elevated section with the fan at the front of the robot, and then elevated the HC-12 and mounted it towards the back of the robot. We elevated the HC-12 to allow for us to save space and mount the cellular phone battery underneath it.

Bottom plate: We mounted two distance sensors, one in the front and one on the left. On the right side, we mounted the fuse, the on-off switch and the power distribution switch. On the bottom of the robot, we started with the motors mounted basically in the middle of the robot. At the back, we mounted the H-bridge. Towards the front, we mounted the Lead-acid battery, caster wheel, and the white line sensor.

High Power System

Our high power system was designed to control our motors, fan, and H-bridge. Our battery went through the H-bridge to the motors and also went directly to the fan subsystem. We chose an 18V lead-acid battery that could supply up to 2 amps for 1 hour.

Low Power System

The low power system needed to supply the HC12, flame sensors, white line, and distance sensors. These were done in a series type configuration. All components could be run off of 5 volts so we chose to power this system with a 7.2 volt cell phone battery that was donated to us. The HC12 can be fried with very little current so it was very important that we separate the two systems in a way that the high power system had NO contact with the low power system.

Motors

We needed 2 motors, one for each wheel, to guide us through the maze.

Maxon: We were given 5 motor selections. In our preliminary calculations, we calculated that at worst case scenarios we need 4.25 Watts to power our robot. We chose the Maxon S 2322 6-watt motor because it consumes a low 1.4 A start-up current. This would be easy to find an H-bridge to supply this.

Wheels: We chose the 1.5-inch wheels because we had no intention of going over the ramps, and these wheels gave us no problems at all.

H-bridge

The H-bridge was used to power the motors and was easily used to change the direction of the motors.

Allegro 2998: We used the allegro 2998 dual full-bridge motor driver because of its dual capabilities and its availability in the storeroom. This H-bridge will operate continuous output currents of 2 A per bridge or start-up currents to 3 A. It has internal PWM current control and is compatible with inputs of 3.3V or 5V. Our motors draw a startup current of 1.3 A so this H-bridge worked just fine

H-bridge board: For the H-bridge board, we chose wire wrapping because we needed it immediately and didn't yet have protel mastered. The H-bridge ran on 5V so we ran 12V

into a 7805 to get our 5V out. This worked well since we needed 5V for the opto-isolators. We used the opto-isolators they were in the storeroom because they were available and had been used effectively in the past. We ran the PWM system from the HC-12 into 2 of the inputs of the opto-isolators and then the direction pins into the other 2 inputs of the other two opto-isolators. We used 510-ohm resistors for the inputs, and 240-ohm for the output resistors. The output of each of the opto-isolators went to the H-Bridge, and from there, signals were sent to the motors. By sending a 0V or 5V to the H-Bridge, it would switch the polarity of the motors. This was easily done using port P of the HC-12.

White Line Sensor:

The white line sensor proved to be a much bigger hassle than it ever should have.

Initial design: We initially intended to go with a designed borrowed from a combination of the group 5 of spring 2000 and EE 212 lab 5. This design mounted a light bulb and a PN168 phototransistor on a board mounted to the underside of our board. The idea was for the bulb to continuously shine. When we crossed a white line, the reflection would be bright and the PN168 would pick it up. This set up worked off of visible light.

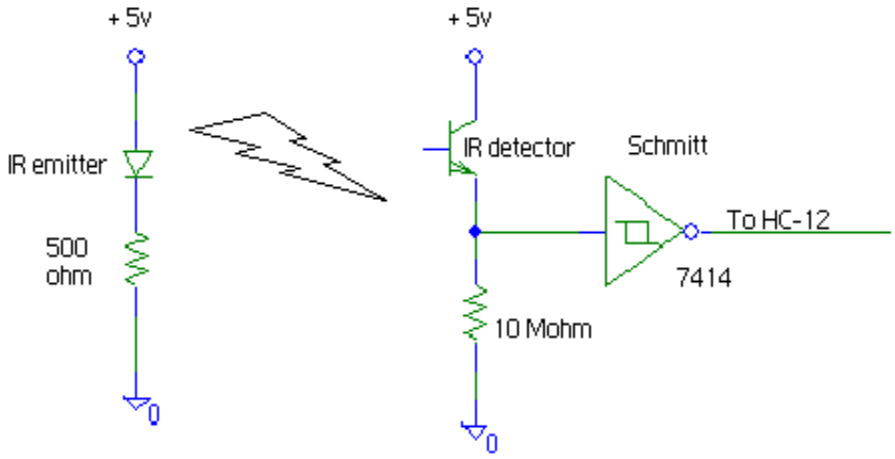
Packaged detector: On the way to the parts bin to gather my goodies, I found a neatly packaged infrared emitter/detector pair. This seemed like a good idea since it required minimal external circuitry. We initially tested it using our protoboard and weren't getting the response we wanted. I decided that the emitter was bad. I took apart the package and

it seemed that the diode had been burnt out, so I replaced it with a regular LED. This worked great! I immediately bought another package and replaced the IR diode with a visible light LED. We rigorously tested this and were getting around .5 to .7 volts from a black surface and 4.7 to 4.9 volts on a white line.

Trial and error: Then came time to mount the sensor under the robot and hook it to our HC12. Disaster!! We weren't getting any difference between black and white. We would take it off of the robot to test and it worked the same as our initial good readings. Back under the robot and no good. We decided that maybe we weren't getting different enough readings for the HC12 to read. So we put a Schmitt trigger between the output of the packaged pair and the HC12 input port. This would lock on to a high or low voltage and give a digital signal (high or low) to the HC12. It worked great in testing. It didn't work at all under the robot.

Looking to a higher power: At this point we had to enlist help from a higher power. As instructed, we went back to the original IR emitter in the "IR emitter/detector pair." Just to be safe (we've also been known to wear both a belt and suspenders) we put a Schmitt trigger on the output of the pair to give a definite high or low to our HC12 port. Wouldn't you know, we mounted it under the robot and it worked on the first try! We later learned that another group was having the same problems with a regular LED not giving good readings under the robot. After much testing, they discovered that the sensor was detecting white lines off of the robot just fine because it was picking up reflections from ambient light. The LED was doing nothing for the good of the robot. To remedy this

problem, they just mounted an extra light bulb under the robot, thus going back to the original plan of a light bulb and a phototransistor!



Distance Sensor

We chose the SHARP GP2D120 distance sensor to tell how far we were from walls so our robot didn't run into them. We originally had one on each side of the robot and one in the front. We planned to use A to D ports for everything and were limited to 6. Thus, we found a way to navigate the maze using only 2 such sensors: a left and front sensor. We put a 33uF capacitor between power and ground as per the spec sheet instructions.

Sensor positioning. We mounted the front sensor in the center of the robot and hooked it to an A/D port. We tried to keep the robot at least 10cm away from front walls and 7cm away from left walls. We angled the left sensor to roughly 45 degrees so that we would shoot the signal to the wall and then not pass the point of reflection before the return

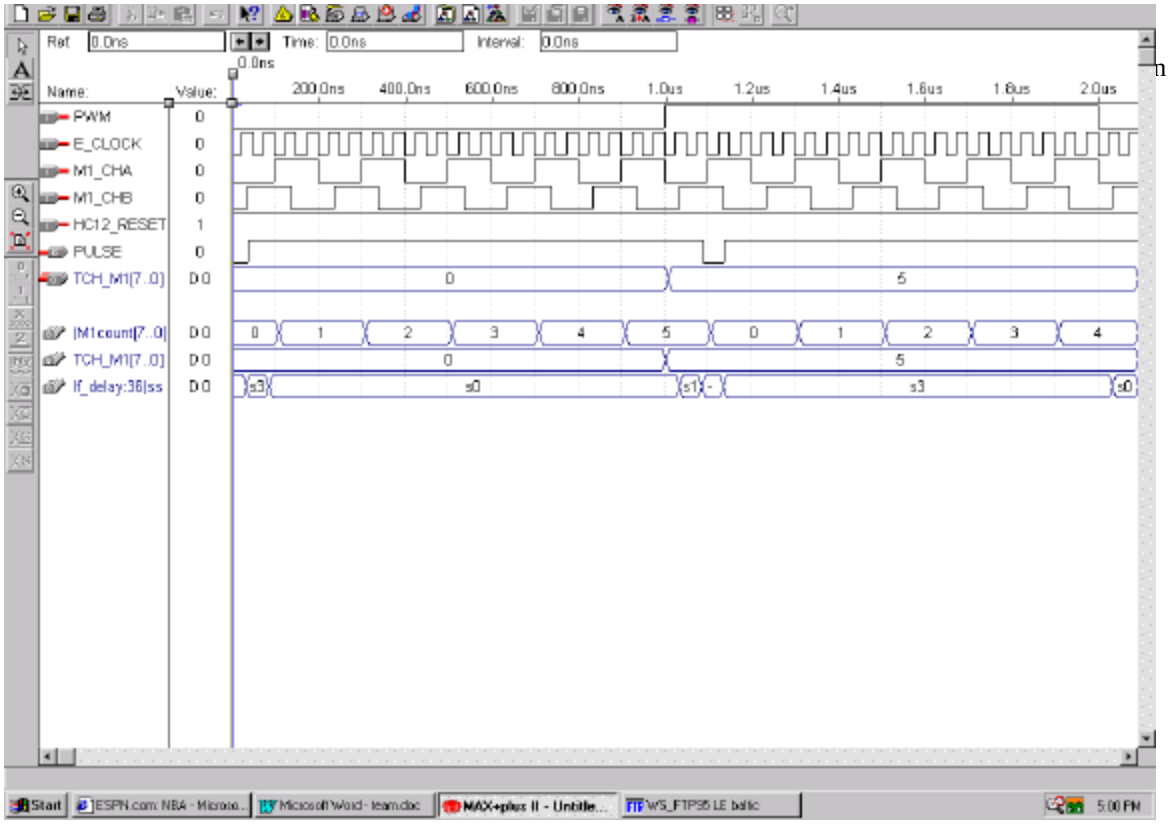
signal could be picked up, as would happen if we were at a 90 degree angle to the wall. Trial and error allowed us to find the right angle for the sensor.

Closed Loop Control

One of the requirements of the design project was that it have closed loop control. We elected to do this through Altera using motor encoders. We never got this implemented, but we did have a pretty good idea of how we wanted to do it.

The first step was to determine motor speed and direction. To do this we used counters and a state machine. The counters told if we were going forward or backward based on the direction of the count. The states of the state machine acted as a delay so we could latch our counted values. The encoders were internal on the motors. They consisted of two lights/sensors, A and B, at 90 degrees to each other. There were tick marks on a disk connected directly to the motor shaft. As the shaft turned, the tick mark moved, blocking either A or B. Eventually both. Which was blocked first determined motor direction. How frequently they were blocked determined motor speed.

We simulated what would happen if we did have our motors hooked up to this code.



Code Development

Our main was written in C. We weren't very proficient programmers so we didn't utilize functions for anything other than the basics. We did our entire program using for, if, and while statements. We had each of our sensors hooked to an A/D port so all we needed to worry about was high or low.

Wall-following: Our base was wall following. We used our GP2D120s to stay 6cm from the left wall and to make a right turn when the front sensor saw a wall less than 10cm away. If we got too close or too far from a left wall, our right motor speeded up or slowed down accordingly.

In a room: We were counting white lines to tell when we entered and left a room. When we crossed a white line, we stopped briefly, and then scanned the room left 90degrees, then right 90degrees, then back to our original position. If a flame was detected, we entered the room and left wall followed until we saw anther white line. Then we stopped and turned our fan on. This is not a very practical way of finding a fire and it was not our first idea, but it proved to be by far the easiest to implement. If no flame was detected in the room, we rolled straight back for a specified count value, then turned right for another specified count value. These values were not always perfect due to variations in battery charge, but they worked fine for our application. Then we continued to left wall follow to the next room.

Island room: Finding the island room required a bit of strategic thinking because we had no right wall sensor. How we found this room was to count white lines. We first looked for white line number 3, then proceeded to turn right and dead-reckon forward until the front wall sensor saw the island room. We then wall-followed around to the entrance of the room. The problem with this is that the robot had to start in the correct position to properly see the island room. So we abandoned that and tried a different method. If we reached white line 4, that meant that we had returned to the home circle. If this were the case, we backed straight up a specified amount, turned for another specified count (hopefully that amounted to roughly 180 degrees) and then left wall followed to the next white line which was the door to the island room. We then went back into our scan for fire and left wall follow to candle circle routine. This worked beautifully and every time we attempted it.

Flame Sensor

Our choices for the fire sensor were the Hamamatsu and the PN168. We chose the PN168 mostly because of cost as the Hamamatsu was a lot more expensive and the PN168s were given to us.

Binocular vision: Our first idea was to use binocular vision. We were going to use 2 PN168s to zero in on the flame by trying to match the voltages received by each. We mounted the sensors in cardboard tubes and put a 10K pot on the input. This allowed us to calibrate our sensors to put out the same values on each sensor at the same distance. At best, we could detect a flame at 40 cm. We then needed to calculate the half power point of each so that we would know where our vision for each overlapped. One half power point was 35 degrees and the other was 36 degrees. We intended to do most of the dirty work in software. We wanted to scan the room, taking a reading every 1/10 second or so, and save the largest value we encountered. For the reverse scan, we would drop the last to bits of our highest number (in case of a flicker), and back track until we reached that number again. Then we would move forward from that point using binocular vision to zero in on our flame and then turn on the fan when we hit the candle white line. This was ineffective since we couldn't see the flame from more than 15 inches away.

Stronger flame sensor: We then decided to add another fire sensor with its only purpose being to see if there is a flame in the room, and then use the other two as binoculars to get

us close to the flame. To do so, we definitely needed a stronger sensor, so we tried working with the resistor values. Instead of 100k, we tried 50k as suggested by another team. This proved very unsuccessful, as the reading from our voltmeter just got lower. We then upped the resistor value to 1M and we saw some difference, so we then tried a 3.3M resistor and got excellent readings. We could now sense the flame from over a meter away. The only problem was that it was very sensitive to ambient light so we would need to really filter this sensor, as we had not done it with the other two. We wanted a wider range of sight rather than just a pen cap, so we used a credit card. We folded the credit card into a box and painted it black. We still weren't blocking out enough light so we decided to filter its view. We tore up a floppy disk and took the IR filter out of it. We covered the box entirely and it worked too well at keeping all the ambient light that it couldn't see a flame from more than a foot. We then put strips of black tape and strips of the floppy disk across it. This proved to work beautifully. We could now see the flame from about a meter away, which was about the distance we needed from any doorway. We tested this in the maze and could in fact see the flame from every doorway and we didn't have any problems with reflection, as it wasn't too powerful. This single flame sensor worked well enough for us that we abandoned the binocular vision and just kept the one sensor. We used it to get us into the room with the flame, and we then wall-followed to the white circle, stopped and extinguished the flame.

Fire Extinguisher

Danger: To extinguish the fire, we went with the mother of all fans. We used a motor and propeller from an old remote controlled airplane. We ran 12 volts to it thinking that it would only be on for a total of like 5 seconds, so it would be fine. The propeller was about 8-inches long and that proved to be dangerous. Not only did we almost lose a limb every time the robot thought it saw fire, but it would almost start to try and lift the robot off the ground. So for testing purposes, we used replaced the fan blade with a piece of black electrical tape. This itself could put out the flame. The day before the school competition, we were given a fan propeller of smaller and less dangerous materials. It fit on our motor and put out the fire consistently.

Starting the fan: To implement the fan, we used a fan relay from the storeroom. We hooked the ground from the HC-12 to one pin and an output pin from the HC-12 to another. We ran 12 volts to one end of the motor and the high power ground to one of the pins on the relay. The last pin on the relay was tied to the negative side of the motor. We programmed the robot to stop at the white line when it has entered a room and advanced to the flame. It would then set the pin to 5volts and turn the fan on.

APPENDIX A: Budget

DESCRIPTION	PRICE	QTY	TOTAL
H-Bridge (Allegro)UDN2998W	$7.00	1	$7.00
Perf-board 1.5”X1.75”:	$1.00	3	$3.00
Schmitt trigger optoisolators	$1.50	4	$6.00
5V regulators: 7805	$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	1	$1.25
Terminal housing (with pins) 5-pin (pair):	$1.75	1	$1.75
Terminal housing (with pins) 6-pin (pair):	$2.50	1	$2.50
3/32” shrink tubing (price per inch)	$0.10	6	$0.60
IR distance sensors:0-80cm: GP2D12:	$10.00	3	$30.00
Maxon 6-Watt 22mm motors (green body):	$50.00	2	$100.00
12V 2.0Ahr lead battery:	$30.00	1	$30.00
12V lead-acid chargers:	$9.00	1	$9.00
9V Alkaline batteries:	$2.00	2	$4.00
Fuse holder (GMA fuses):	$1.00	1	$1.00
GMA fuses(1,2,4,6A):	$0.20	1	$0.20
0.5A sub-miniature fuses:	$1.00	1	$1.00
Solder-tail DIP socket2x3-pin:	$0.20	4	$0.80
4-pin audio locking conn. male-straight:	$0.50	4	$2.00
4-pin audio locking conn. female:	$1.50	4	$6.00
Polarized single-row housing:6-pin housing:	$0.80	1	$0.80
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	3	$4.50
1k Ohm 10-turn pot:	$0.50	1	$0.50
Misc. capacitors:	$0.10	3	$0.30
Misc. LEDS	$0.10	2	$0.20
1.5” rubber wheels:	$2.50	2	$5.00
Caster wheel assembly	$10.00	2	$20.00
misc			2.81
GRAND TOTAL			**$253.46**

APPENDIX B: Final Code

```
/*THIS is the final program that puts out the flame in any room*/

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

#define TRUE  1
#define FALSE 0
#define DD  71    /*desired distance from wall*/
#define DCleft    PWDTY0
#define DCright   PWDTY1
#define DM  ADR3H /*actual (measured) distance from wall*/
#define DS  ADR4H /*front sensor*/
#define DW  ADR2H /*white line*/
#define DFS ADR5H /*side fire sensor*/
#define DFF ADR6H /*front fire sensor*/
#define go  ADR7H /*big fire sensor*/

int ERROR, FD, ADD, dif;
int i, LINE;
int leftspeed, rightspeed;

main()
{
      ATDCTL2 |= 0x80;  /* Power up A/D */
        ATDCTL4 = 0x01;       /* 9 us/conversion */
        ATDCTL5 = 0x70;       /* 0 1 1 1 0 x x x 0 */

      PWCLK = PWCLK & ~0xC0;  /* Choose 8-bit mode */
      PWCTL = PWCTL & ~0x08;  /* Choose left-aligned */
      PWPOL = PWPOL | 0x0f;   /* Choose high polarity */
      PWPOL = PWPOL | 0x30;   /* Select clock mode 1 for Channels 1 and
0 */

      /* Select N = 0 and M = 2 for Channels 1 and 0 */
      PWCLK = PWCLK & ~0x38;  /* N = 0 */
      PWSCAL0 = 0x02;         /* M = 2 */


      /* Select period of 256 for Channels 1 and 0 */
      PWPER1 = 255;           /* Period 1 = PWPER1 + 1 */
      PWPER0 = 255;           /* Period 0 = PWPER0 + 1 */
      PWEN = PWEN | 0x03;     /* Enable PWM on Channels 1 and 0 */

      PWDTY1 = 80;
      PWDTY0 = 80;            /* initializes speeds */
      DDRP = 0xA0;            /*bit 7 and 6 are output, bits 5 through
input*/
      DDRT = 0xA0;            /*direction pin */
      PORTP = 0x00;           /*direction pin right motor*/
      PORTT = 0x80;           /*direction pin left motor*/

      LINE = 0;         /*initialize white line count at 0*/
      DCleft = 110;           /*initialize speed*/
```

```
      DCright=80;       /*initialize speed*/

while(TRUE)       /*always do this*/
{
      PORTP=0x00;
      ERROR = ( DD - DM ) ;
      leftspeed = 110 - ERROR;
      rightspeed = 80 + ERROR;
      if( DM <60)       /*if left measured is < 60, move in toward
wall*/
      {
            leftspeed = 110 - ( 2 * ERROR );
            rightspeed = 80 + ( ERROR * 2 );
      }
      if( DM >90)       /*if  left measured is >90, move away from
wall*/
      {
            leftspeed = 110 - ERROR ;
            rightspeed = 80 + ( ERROR * 2) ;
      }
      if(rightspeed<0)  /*changes R direction rather than overflowing*/
      {
            PORTP=PORTP | 0x80;
            DCright = -rightspeed;
      }
      else              /*maintain R direction*/
      {
            PORTP=PORTP & ~0x80;
            DCright = rightspeed;
      }
      if(leftspeed<0)         /*changes L  direction rather than
overflowing*/
      {
            PORTT = 0x00;
            DCleft = -leftspeed;
      }
      else              /*maintain L direction*/
      {
            PORTT = 0x80;
            DCleft = leftspeed;
      }
      PORTT = 0x80;

      if(DS>45)         /*front sensor notes wall, turns */
      {
            while(DS>60)      /*front sensor stops turning when wall
value is < 60 */
            {
                  PORTP=0x80;
                  PORTT=0x80;
                  DCleft=200;
                  DCright=200;
                  DBug12FNP->printf("im still here  %d\n",ADR4H);
            }
      PORTP=0x00; /*sets R motor to go forward*/
      }
```

```
      DBug12FNP->printf("front sensor:    %d;left sensor:   %d  fire
%d\n\r", ADR4H, ADR3H, ADR7H);

      if(DW==0)         /*if white line detected*/
      {     LINE = LINE + 1;  /*counts number of lines*/
            if (LINE != 4 )
            {
                  for(i=0;i<33;i++)
                  {
                        PORTT=0x80;
                        PORTP=0x00;
                        DCright=80;
                        DCleft=80;
                        DBug12FNP->printf("i is %d\n",i);
                  }
                  for(i=0;i<100;i++)
                  {
                        PORTP=0x80;
                        PORTT=0x80;
                        DCright = 80;
                        DCleft = 80;
                        DBug12FNP->printf("i is %d\n",i);
            if(go>9)          /*fire is in room*/

                  {
                        while (1)
                        {
                              PORTP=0x00;
                              ERROR = ( DD - DM ) ;
                              leftspeed = 110 - ERROR;
                              rightspeed = 80 + ERROR;
                              if( DM <60)       /*if left measured is <
60, move in toward wall*/
                              {
                                    leftspeed = 110 - ( 2 * ERROR );
                                    rightspeed = 80 + ( ERROR * 2 );
                              }
                              if( DM >90)       /*if  left measured is
>90, move away from wall*/
                              {
                                    leftspeed = 110 - ERROR ;
                                    rightspeed = 80 + ( ERROR * 2) ;
                              }
                              if(rightspeed<0)  /*changes R direction
rather than overflowing*/
                              {
                                    PORTP=PORTP | 0x80;
                                    DCright = -rightspeed;
                              }
                              else              /*maintain R
direction*/
                              {
                                    PORTP=PORTP & ~0x80;
                                    DCright = rightspeed;
                              }
```

```
                              if(leftspeed<0)         /*changes L
direction rather than overflowing*/
                              {
                                    PORTT = 0x00;
                                    DCleft = -leftspeed;
                              }
                              else              /*maintain L
direction*/
                              {
                                    PORTT = 0x80;
                                    DCleft = leftspeed;
                              }
                              PORTT = 0x80;
                              if(DS>45)         /*front sensor notes
wall, turns */
                              {
                                    while(DS>60)      /*front sensor
stops turning when wall value is < 60 */
                                    {
                                          PORTP=0x80;
                                          PORTT=0x80;
                                          DCleft=200;
                                          DCright=200;
                                          DBug12FNP->printf("im still
here  %d\n",ADR4H);
                                    }
                              PORTP=0x00; /*sets R motor to go
forward*/
                              }
                              if (DW==0)
                              {
                                    while (DW==0)
                                    {
                                          DCleft = 0;
                                          DCright = 0;
                                          PORTT = 0x20;
                                    }

                              }
                        }
                  DCleft=0;
                  DCright=0;
                  }
                  }
                  for(i=0;i<150;i++)
                  {
                        PORTP=0x00;
                        PORTT=0x00;
                        DCright = 80;
                        DCleft = 80;
                        DBug12FNP->printf("i is %d\n",i);
      if(go>9)          /*fire is in room*/

                  {
                        while (1)
                        {
                              PORTP=0x00;
```

```
                              ERROR = ( DD - DM ) ;
                              leftspeed = 110 - ERROR;
                              rightspeed = 80 + ERROR;
                              if( DM <60)       /*if left measured is <
60, move in toward wall*/
                              {
                                    leftspeed = 110 - ( 2 * ERROR );
                                    rightspeed = 80 + ( ERROR * 2 );
                              }
                              if( DM >90)       /*if  left measured is
>90, move away from wall*/
                              {
                                    leftspeed = 110 - ERROR ;
                                    rightspeed = 80 + ( ERROR * 2) ;
                              }
                              if(rightspeed<0)  /*changes R direction
rather than overflowing*/
                              {
                                    PORTP=PORTP | 0x80;
                                    DCright = -rightspeed;
                              }
                              else              /*maintain R
direction*/
                              {
                                    PORTP=PORTP & ~0x80;
                                    DCright = rightspeed;
                              }
                              if(leftspeed<0)         /*changes L
direction rather than overflowing*/
                              {
                                    PORTT = 0x00;
                                    DCleft = -leftspeed;
                              }
                              else              /*maintain L
direction*/
                              {
                                    PORTT = 0x80;
                                    DCleft = leftspeed;
                              }
                              PORTT = 0x80;
                              if(DS>45)         /*front sensor notes
wall, turns */
                              {
                                    while(DS>60)      /*front sensor
stops turning when wall value is < 60 */
                                    {
                                          PORTP=0x80;
                                          PORTT=0x80;
                                          DCleft=200;
                                          DCright=200;
                                          DBug12FNP->printf("im still
here  %d\n",ADR4H);
                                    }
                              PORTP=0x00; /*sets R motor to go
forward*/
                              }
                              if (DW==0)
```

```
                              {
                                    while (DW==0)
                                    {
                                          DCleft = 0;
                                          DCright = 0;
                                          PORTT = 0x20;
                                    }

                              }
                        }
                  DCleft=0;
                  DCright=0;
                  }

                  }
                  for(i=0;i<50;i++)
                  {
                        PORTP=0x80;
                        PORTT=0x80;
                        DCright = 80;
                        DCleft = 80;
                        DBug12FNP->printf("i is %d\n",i);
                  }

                  DBug12FNP->printf("right fire is %d left fire is
%d\n\r", DFS, DFF);
                  DBug12FNP->printf("DCleft is  %d    DCright is
%d\n\r", DCright, DCleft);
                  if(go>16)         /*fire is in room*/

                  {
                        while (1)
                        {
                              PORTP=0x00;
                              ERROR = ( DD - DM ) ;
                              leftspeed = 110 - ERROR;
                              rightspeed = 80 + ERROR;
                              if( DM <60)       /*if left measured is <
60, move in toward wall*/
                              {
                                    leftspeed = 110 - ( 2 * ERROR );
                                    rightspeed = 80 + ( ERROR * 2 );
                              }
                              if( DM >90)       /*if  left measured is
>90, move away from wall*/
                              {
                                    leftspeed = 110 - ERROR ;
                                    rightspeed = 80 + ( ERROR * 2) ;
                              }
                              if(rightspeed<0)  /*changes R direction
rather than overflowing*/
                              {
                                    PORTP=PORTP | 0x80;
                                    DCright = -rightspeed;
                              }
                              else              /*maintain R
direction*/
```

```
                                {
                                      PORTP=PORTP & ~0x80;
                                      DCright = rightspeed;
                                }
                                if(leftspeed<0)         /*changes L
direction rather than overflowing*/
                                {
                                      PORTT = 0x00;
                                      DCleft = -leftspeed;
                                }
                                else              /*maintain L
direction*/
                                {
                                      PORTT = 0x80;
                                      DCleft = leftspeed;
                                }
                                PORTT = 0x80;
                                if(DS>45)         /*front sensor notes
wall, turns */
                                {
                                      while(DS>60)      /*front sensor
stops turning when wall value is < 60 */
                                      {
                                            PORTP=0x80;
                                            PORTT=0x80;
                                            DCleft=200;
                                            DCright=200;
                                            DBug12FNP->printf("im still
here  %d\n",ADR4H);
                                      }
                                PORTP=0x00; /*sets R motor to go
forward*/
                                }
                                if (DW==0)
                                {
                                      while (DW==0)
                                      {
                                            DCleft = 0;
                                            DCright = 0;
                                            PORTT = 0x20;
                                      }

                                }
                          }
                    DCleft=0;
                    DCright=0;
                    }
                    else              /*fire is not in room*/
                    {
                          for(i=0;i<40;i++) /*counts*/
                          {
                                PORTP=0x80; /*R backward*/
                                PORTT=0x00; /*L backward*/
                                DCleft = 110;
                                DCright = 80;     /*goes back in straight
line*/
```

```
                              DBug12FNP->printf("value of i: %d\n\r",
i);
                        }
                  }
                  {
                        for(i=0;i<22;i++) /*counts*/
                        {
                              DCleft=200; /*turns right*/
                              PORTP=0x80; /*R backward*/
                              DCright=200;
                              PORTT=0x80; /*L forward*/
                              DBug12FNP->printf("value of i: %d\n\r ",
i);
                        }
                  }
                  {
                        for(i=0;i<33;i++) /*counts*/
                        {
                              PORTT=0x80; /*L forward*/
                              PORTP=0x00; /*R forward*/
                              DCright=100;
                              DCleft=120; /*goes forward to pick up
left wall again*/
                              DBug12FNP->printf("value of i: %d\n\r ",
i);
                              if(LINE == 4)
                              {
                                    for(i=0;i<10;i++) /*dead-reckon
backward count*/
                                    {
                                          PORTP = 0x80;     /*R
backward*/
                                          PORTT = 0x00;     /*L
backward*/
                                          DCleft = 110;
                                          DCright = 80;
                                          DBug12FNP->printf("value in
LINE loop: %d\n\r", i);
                                    }
                                    for(i=0;i<42;i++)
                                    {
                                          PORTP=0x80;
                                          PORTT=0x80;
                                          DCleft=200;
                                          DCright=200;
                                    }
                              PORTT = 0x00;
                              }
                        }
                  }
            }

            else  /*line is 4*/
            {
                  for(i=0;i<20;i++) /*dead-reckon backward count*/
                  {
                        PORTP = 0x80;     /*R backward*/
```

```
                        PORTT = 0x00;     /*L backward*/
                        DCleft = 110;
                        DCright = 80;
                        DBug12FNP->printf("value in LINE loop: %d\n\r",
i);
                  }
                  for(i=0;i<37;i++) /*turn around*/
                  {
                        PORTP = 0x80;     /*R backward*/
                        PORTT = 0x80;     /*L forward*/
                        DCleft = 170;
                        DCright = 150;
                        DBug12FNP->printf("value in LINE loop: %d\n\r",
i);
                  }
                  LINE = LINE + 1;
            }

}


}     /*ends while*/
}     /*ends main*/
```