Marshall

The Autonomous
Fire-Seeking and Extinguishing
Robot

By

Steven Logreira

EEL 5934
Final written report
9/15/97

Table of Contents

ABSTRACT

People choose careers as a way to support themselves and their families. Some careers are far more dangerous than others; one such career is being a fireman. The fireman is an unsung hero that puts his/her life on the line every day to help those in need. A fireman has to make quick and correct decisions every time that he/she is facing a dangerous situation. Bad judgment can lead to serious injury or even death.

"Life is too precious to jeopardize or lose."

What if instead of sending people to fight fires, we can send machines to fight fires. Machines never tire and can work 24 hours a day, seven days a week.

At the beginning of the semester, I saw a news bulletin about how a fireman tragically lost his life in a fire. That terrible news inspired me to develop on the idea of an autonomous fire-fighting robot.

The robot's main design would be to perform in certain situations much too dangerous for humans. The robot would be outfitted with special pyro-sensors that detect elevated heat levels. Once a fire has been detected, it is equipped with a water pump and a water-reservoir to give it fire-extinguishing capabilities. It will also possess some basic collision avoidance sensors to keep it from getting stranded

In essence, this autonomous, self-propelled robot will save lives and tax-payer's money if ever implemented on a commercial scale.

INTRODUCTION

Background

Fire-fighting has always been a very dangerous profession. Fire fighters spend most of their careers putting their life on the line. Current fire-fighting techniques require that firefighters take high risks. Why not send a robot to take these risks instead. Robots are not sensitive to smoke and can be made to withstand large amounts of heat. It is imperative to reduce the risk of losing precious life.

Scope and Objective

This project code-named Marshall is designed to implement many different functions such as fire detection, collision avoidance through Infrared (IR) sensors, bump sensing in case the IRs fail, and a water dispensing mechanism.

The objective of this project will be to built a robot using a Motorola 68HC11 microcontroller EVBU board with a Novasoft ME11 expansion board and various sensors.

The scope of this project is to understand how by implementing different sensing techniques we can achieve a product that can be eventually used in a real-life scenario.

Structure of the report

In the following sections, I will discuss in detail all the various components that make up Marshall. I will begin by discussing the platform that makes up Marshall's chassis. Then I will discuss the motors that propel him. followed by a detailed explanation of some of Marshall's sensors. Then I will discuss the lights, voice module and water spraying mechanism and finally it's expected behavior patterns.

INTEGRATED SYSTEM

Overview

Marshall is built on a 10-inch diameter aluminum platform designed by the MIL team at the University of Florida code-named "*Tolric*."

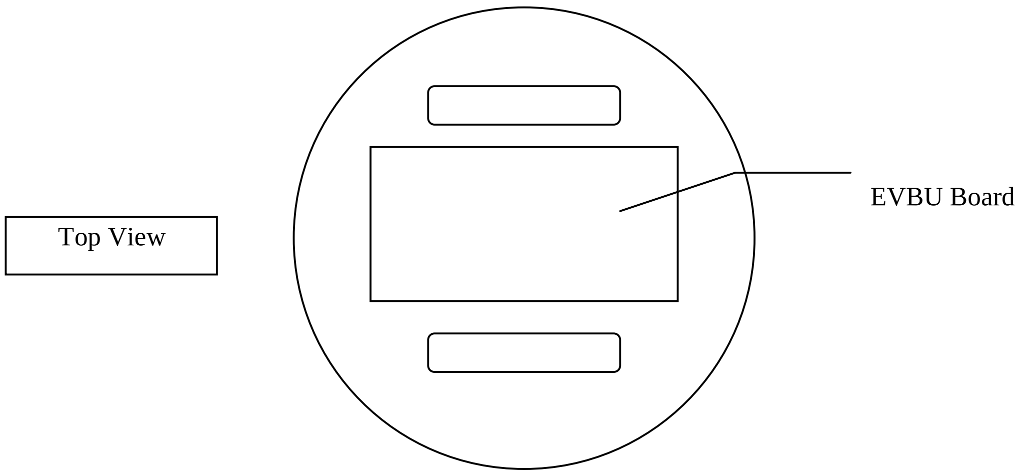
Mounted on top of the chassis, is a 68HC11 EVBU board and an ME11 32K expansion board. Other hardware include two hacked Futaba servo motors, Four hacked Sharp IR sensors with their respective IR LEDs, switches used as bumpers as a backup to the collision avoidance system, a pyro-sensor to detect fire, a lights and voice module warning circuit and finally a water spraying system consisting of a DC water pump and a water reservoir. Also note that mounted on top of the frame is the chassis of a toy fire truck to make the design aesthetically pleasing.

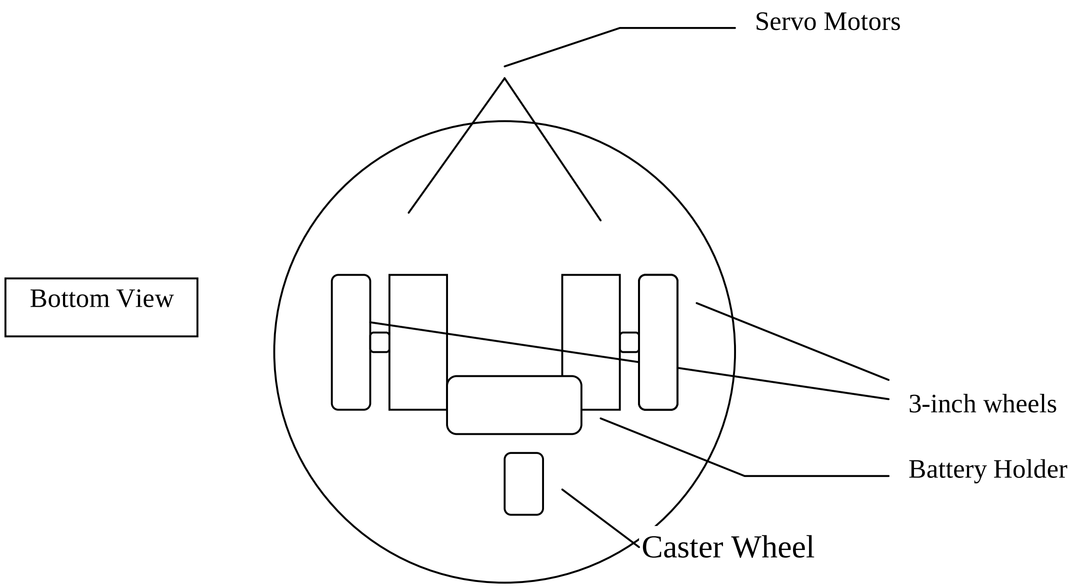
Aluminum Platform and toy chassis

The 10-inch circular aluminum platform is designed to hold two servo motors with their respective wheels, the Processor boards, the system sensors and all other mechanisms.

I proceeded to paint the aluminum chassis red; symbolic of a fire truck. Mounted on the chassis is the frame of a toy Chevy Blazer emergency vehicle.. The toy frame will also hold the water pump and its' reservoir as well as the lights, voice module and the Pyro-sensor.

I chose the round shape for the frame because it is more functional than a rectangular frame. A rectangular body will have certain blind spots on it's bump sensors that will partially inhibit its bump-sensing capabilities. Because of the round shape, the robot will have a much broader sensing scope. A top, front and side view of the robot can be seen on Figure I.


Top View
EVBU Board


Servo Motors
Bottom View
3-inch wheels
Battery Holder
Caster Wheel

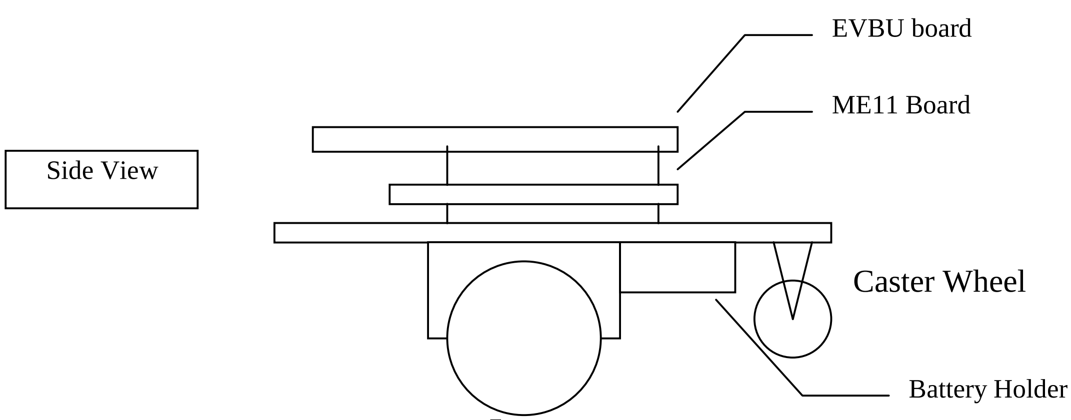

EVBU board
ME11 Board
Side View
Caster Wheel
Battery Holder

Figure I - top, bottom and side view of Marshall

Fire Engine Body

Marshall's body was adapted from a toy emergency vehicle. It rests on top of the Aluminum frame as can be seen in figure 1.a. The water pump, the water tank, the pyro sensor, the lights, the spray nozzle, bump sensors and IR transmitters and receivers have been incorporated into the toy car chassis.

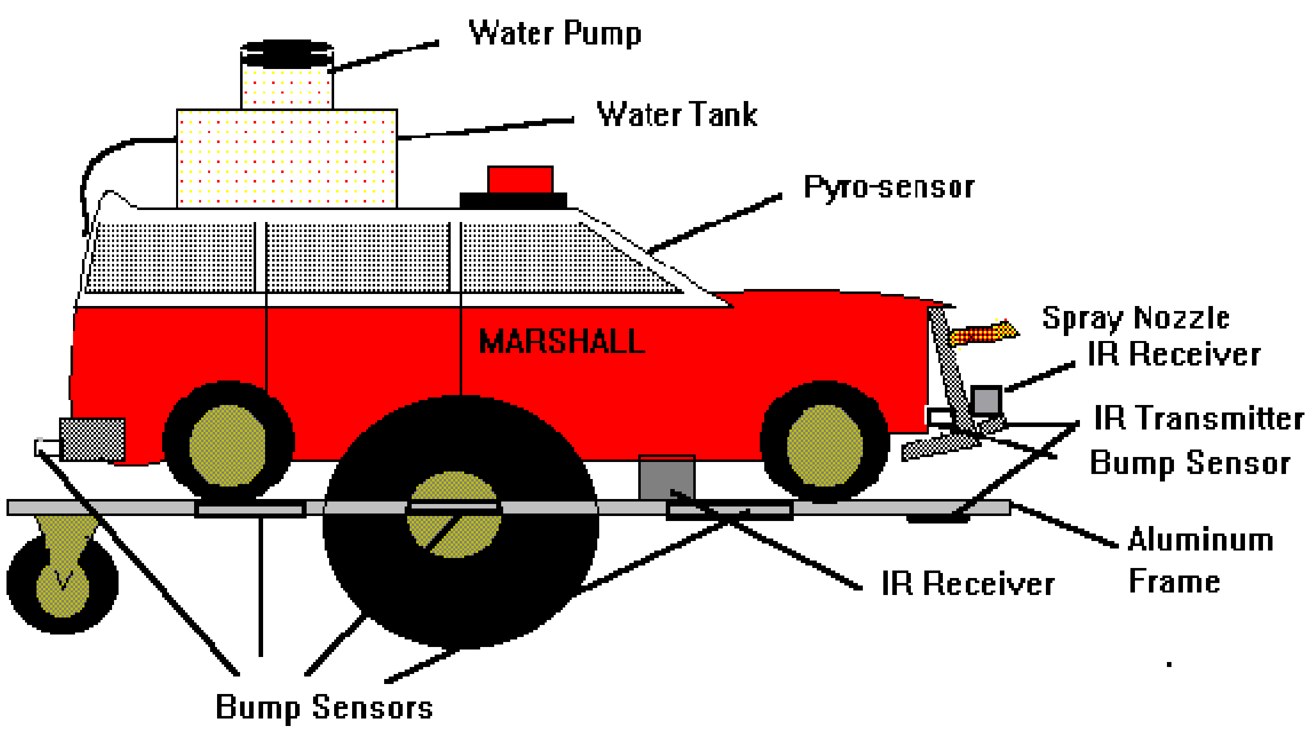


Figure 1.a - Toy car body mounted on chassis

Servo Motors

The motors on Marshall are Hacked Futaba servos. I hacked them using a technique developed by the MIL lab staff. Basically I clipped off a small tab on the gear shaft, I removed the electronics board inside the motor casing and soldered the wires straight to the motors. Then I attached three-inch wheels to the servos by forcing a screw through the center shaft of the wheel and of the servo motor. The specifications for the motor can be seen on table I:

Power Supply:	4.8 to 6.0 Volts
Power Consumption:	6.0 Volts @ 8mA(when idle)
Output Torque:	42 oz/in
Operation Speed:	1.32 rev/sec
Weight:	1.5 Oz
Dimensions:	1.59 X .77 X 1.4 in.

Table I- Specifications for Servo Motors

The motors are controlled by a L293 motor driver chip located on the ME11 expansion board. The direction of the motors is controlled by the PD4 and PD5 pins (**PORT D)** of the Processor and the speed is controlled by pins PA5 and PA6 (**Output Capture**) using pulse-width modulation to control the left and right enable pins.

SENSORS

IR Sensors and IR LEDs

The IR sensors receive a signal from the IR LEDs that tells the processor if an object has been detected. This gives the robot a neural net thus giving the robot object avoidance capabilities. It is imperative that the sensors are correctly placed on the robot to maximize its sensing capabilities. Figure 2 shows an effective way in which the sensors can be arranged and figure 3 shows the control circuitry for the Sensors. The four IR sensors are Sharp GP1U52X complemented by four IR LEDs. Two of the IRs cross in the front (note that the sensors have been mounted on the front grill to give it the effect of headlights. Picture just depicts desired configuration) and there are also sensors on either side. The sensors are connected to a MC14051 MUX outputting the signal to PE0. The MUX control lines are supplied by the 74HC53 Output Latch.

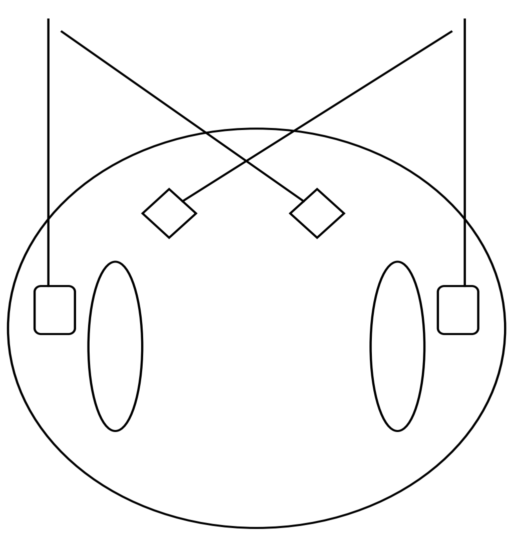


Figure 2 - IR sensor placement

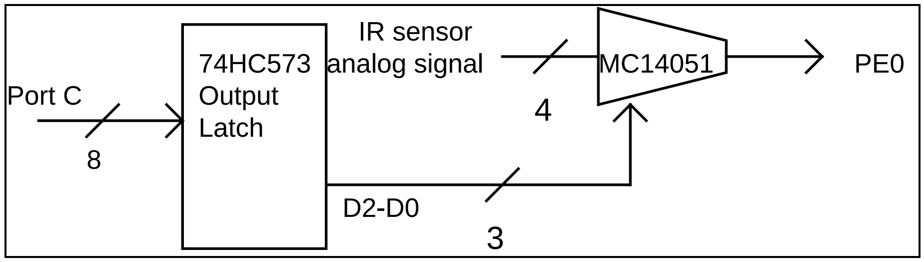


Figure 3 - IR sensor controller

Along with the 4 IR receivers also are four IR transmitters that continuously pulse an IR beam at 40 Khz.

Bump Sensors

The bump sensor mechanism is comprised of a series of contact switches connected in parallel that trigger certain behaviors of the robot. I will talk discuss more about these behaviors on the "behaviors" segment of this document. The circuit schematics can be seen in Appendix B

Pyro-Electric sensor

A Pyro-Electric-sensor is a one that detects changes in heat radiation with respect to movement and distance. It is composed of Lithium Tantalite crystals that induce a charge when heated. This charge is then amplified. These sensors are typically used to detect motion. When a heat source passes in front of the sensor, it emits an analog positive or negative voltage swing; the voltage depends on which way the subject is moving (right or left). This voltage level change can be used to trigger a particular behavior. These type of sensors are very popular in outdoor motion detectors. Figure 4 shows a picture of the pyro-sensor.

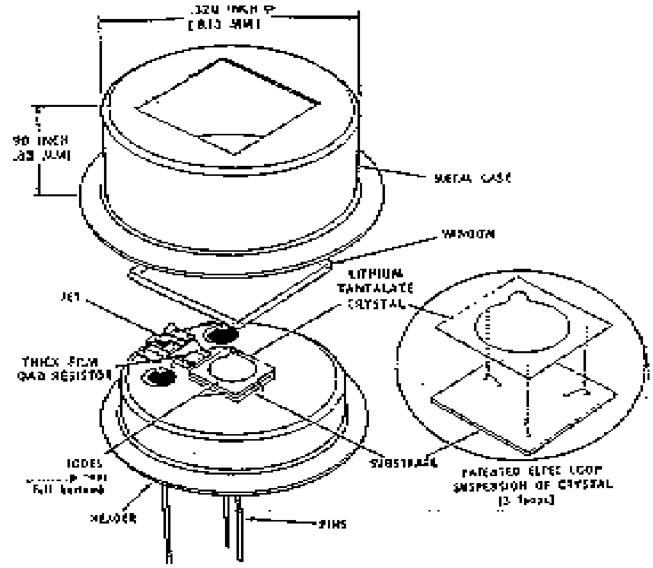


Figure 4 - Pyro-Electric Sensor components.

Motivation for using a pyro-sensor

The main behavior of my robot is to detect a fire or an intense heat source; therefore, a Pyro-Electric sensor is best suited for that task. I can take the readings from the sensor into an A/D port of the 68HC11 Microcontroller and according to certain thresholds trigger a behavior. In my case, the behavior will be to turn on a siren and flashing lights and then direct my robot towards the heat source. Once the robot is close to the heat source, It will pump water from a reservoir and attempt to extinguish the fire.

Hacks to adapt sensor

The sensor I am using came from an outdoor motion detector. Since the sensitivity of those sensors is high, I had to reduce the sensitivity so I could apply it to my design. The motion detector has two parts to it. One is the 120V supply circuitry which steps the voltage down to an acceptable 5V and provides switching circuitry to trigger the flashing lights on, and the other is the circuitry for the sensor. This circuitry includes the sensor itself, voltage regulation and an LM324 op-amp to amplify the signal.

I removed the 120V circuitry and kept the sensor board. There was three cables coming out of the PC board; one was ground, the other was VCC and the third was the signal cable. After further testing, I concluded that the third cable was not the analog signal but a modified DC signal. I then ran further tests and ended up tapping the signal out of pin 11 of the op-amp; which was the desired signal. I then fed the signal to the A/D port of the HC11.

Apart from the internal hacks, I also had to figure out how to reduce the sensitivity of the sensor. The sensor, has a potentiometer that reduces the sensitivity. I turned down the potentiometer and ran tests. It turns out that I could lower the sensitivity enough to where body heat was barely detected and intense heat was quite noticeable. I tested my hack by placing a soldering iron heated to 900° F. Final testing was done using a propane lantern. The lantern will also be used for the final demo.

Another problem was also present. The sensor has a very wide angled field of view and ironically, It was most sensitive at its widest angles and less sensitive at the center. I had to figure out a way to reduce the range of sensitivity. Since the plastic housing of the sensor was small enough and practical enough for me to use, I covered the front window of the housing with aluminum foil and electrical tape and only left a small slit exposed. I then mounted the housing on the inside of the car frame. to give the effect of the car changing directions by itself. This worked quite well. I managed to reduce the angle to almost a straight line and also reduce the total range. See figures 5, 6 and 7.

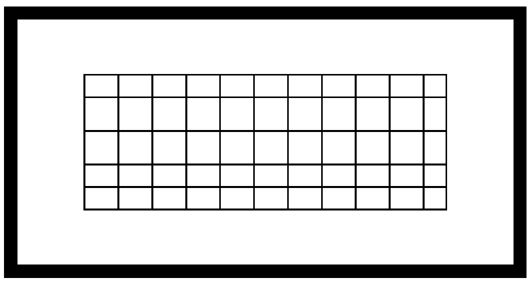


Figure 5- front view of sensor

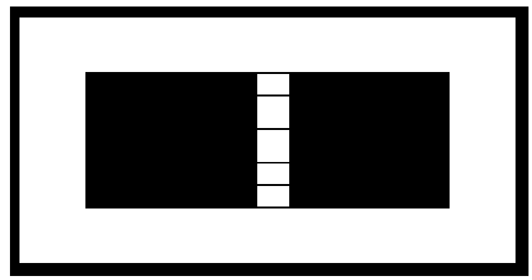


Figure 6 - front view of sensor after filtering

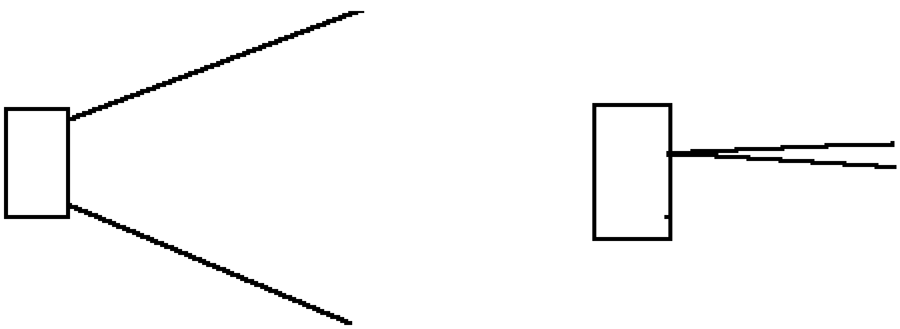


Figure 7 - Field of view before and after filtering

<u>Experiments and data gathering</u> I gathered data using two different methods. First, I wrote a small routine in IC, which can be seen in Appendix I, to read the A/D port of the HC11 and gather various data points and then plotted them using MS Excel. As can be seen in figure 8, there are positive to negative swings which represent movement of a heat source. All my tests where conducted using the hot soldering iron. For some reason unknown to me, the negative voltage spikes have a much bigger amplitude. Also, one can see from the graph that the frequency varies from left to right and in some points the amplitude is much smaller. The frequency and amplitude vary with the distance from the sensor to the subject. The farther away the subject is, the smaller the frequency and thus the smaller the A/D values. As I brought the source closer to the sensor, the frequency and amplitude of the signal increased.

As can be seen by figure 8, I first began gathering data far away from the sensor and then slowly approached it. Then I removed the iron from the field of view only exposing my body heat and then I put the iron in the path again and moved further away without exposing the iron. Please note that the values on the Y-axis are A/D values and not direct voltage readings. The red line represents the reference point where the pyro sensor is not being activated.

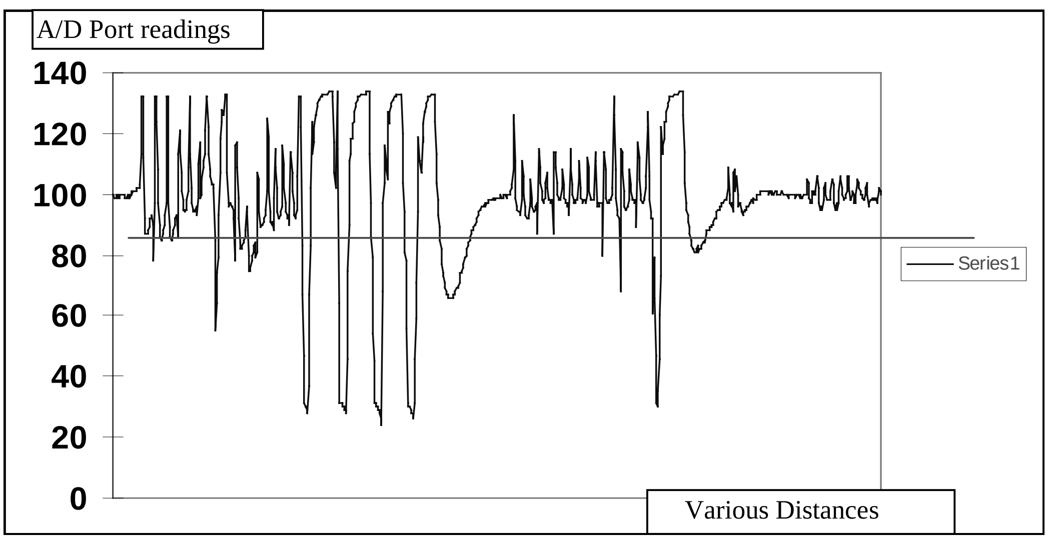
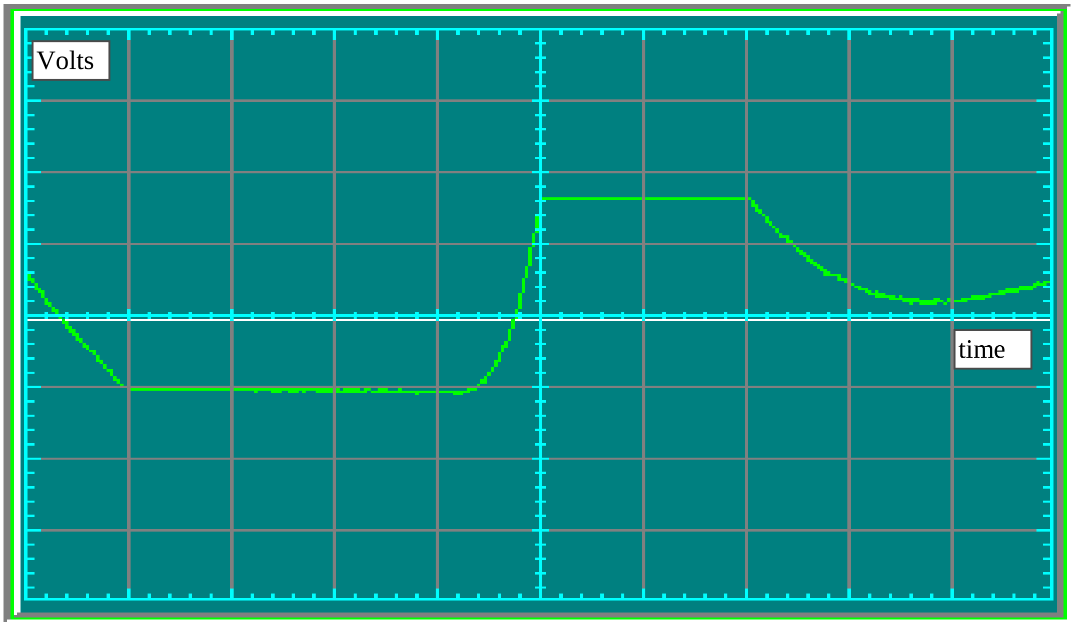


Figure 8 - A/D data points plot

The other method for gathering data was to use the Digital Visual Oscilloscope software and hardware in the MIL lab. Figure 9 show a trace of the soldering iron being swept in front of the sensor. The reference voltage is 2V DC and there is a voltage swing between 2.7 V to -225 mV. which verifies my data gathered from the A/D.



Visual SCOPE Statistics

	Vmin-C2	Freq-C2	Vmax-C2	T-C2
Max	2.0 V	88.9 Hz	2.8 V	1.5 s
min	-225 mV	0 Hz	2.1 V	0 s
Avg	1.6 V	3.8 Hz	2.5 V	258.2 ms

Figure 9 - Figure taken from a Digital Oscilloscope. It measures voltage-Vs-time and gives statistics of measurements.

BEHAVIORS

Search For Fire

Marshall's main behavior is to randomly search for a fire. It's algorithm makes it go straight for five seconds and then rotate 360°. It will run this routine until interrupted by a pyro-sensor read, an IR sensor read or a bump read. The algorithm can be seen in Appendix A.

Collision Avoidance

Another of Marshall's functions is collision avoidance. This behavior is executed by the IR emitter/receiver system. The sensors indicate the distance between the robot and an object. The algorithm does the necessary calculations to tell the robot to change course when the IR threshold has been reached; thus avoiding collision with another object. The control algorithm can be seen in Appendix A

Bump Sensing

The bump sensors are used as a backup system to the collision avoidance circuits. There is three circuits in total; thus three separate behaviors can be achieved. The first circuit makes the robot turn right if the left side and front bumpers have been activated. The second circuit makes the robot turn left if the right side bumpers have been activated.

The third circuit checks to see if the rear bumper has been activated and makes the robot go forward.

Found fire, "Extinguish it!"

When the pyro-sensor has detected a fire, the robot goes into the "fire extinguishing" behavior. The algorithm causes PORTD2 to go hi and thus trigger a "Reed" relay that activates the lights and voice module. Then, Marshall will slowly approach the fire until it reaches a second threshold The second threshold will activate the water pump circuit and spray water in an attempt to extinguish the fire. The water pump circuit is triggered when PORTD3 goes low. The pump circuit consist of a TIP41 NPN transistor and one 2 KΩ and one 10 KΩ resistors. The water pump is basically a 12V automobile window-washer pump. It is powered by a separate 9.6V power supply that can supply the desired current of three Amperes. A separate supply was need to keep the main power supply from draining too quickly. Although the pump requires 12V DC., I found that 9.6V was sufficient to cause the desired behavior. Marshall will spray to a distance of three feet.

CONCLUSION

Fire fighting is a very hazardous way to make a living. Why not let a machine do the job without risk to human life.

My attempt to design a totally autonomous fire fighting system was with the vision that someday it might be implemented in a real-life scenario.

By incorporating collision avoidance, bumpers and most important a system that will warn of threat and actually take action to correct the problem I have attempted to realize a possible design for the future. Although my design is very small scale, it may open the doors for others to follow suit including myself.

This project has opened my eyes to a technology that is still in a very infant stage and has tremendous growth potential. It has also been a very valuable learning experience as to what entails carrying out a real Engineering project from start to finish.

Appendix A

```
/* this piece of code reads values form analog(4); and displays those values
on the screen to help gather pyro-sensor data.  */

void main()
{
  int i;
  init_serial();                      /* call serial.c  */
  write("Here's the Data!\n");

  for (i = 0; i<300; ++I)             /* loop to read 300 values from A/D ports */
  {  write_int(analog(4));
     sleep(.05);
  }
 }
```

```
/* Steven Logreira
   Marshall
   fall/96  */

/********************************************************
* Global Variables
********************************************************/
int PORTD;
int DDRD;
int IR_emmit;

int stop_moving;
int forward;
int backward;
int right_turn;
int soft_right_turn;
int left_turn;

int ir_sense;
int bump_switch;
int pyro_sense;
int find_fire;
int look_for_fire;

int ir_command;
/* int bump_command; */


/**************************************************************
*
* INITIALIZE PORTD 2&3 AS OUTPUTS FOR LIGHTS AND WATER
PUMP   *
* AND SET TO LOW TO START                              *
**************************************************************
*/
void initialize()
{
 int temp;
 temp = peek(DDRD);  /* read DDRD current values            */
 temp|= 0b00001100;  /* mask DDRD to enable output DDR2 DDR3 */
```

```
 poke(DDRD,temp);

 temp = peek(PORTD); /* read PORTD values        */
 temp&= 0b11110011;  /* mask PORTD 2&3 initialize */
 poke(PORTD,temp);   /* Initialize PORTD 2&3 to 0 */
}
/**************************************************************
*
* ACIVATES MOTORS FOR DIFFERENT DIRECTIONS                    *
**************************************************************
*/

void direction(int action)
{
  if(action == stop_moving)
     { motor(0, 0.0); motor(1, 0.0); }
  else if(action == forward)
     { motor(0, 80.0); motor(1, 100.0); }
     /* note: motor 0 = 80 to adjust for unwanted deviations */
  else if(action == backward)
     { motor(0,-100.0); motor(1, -80.0); }
  else if(action == left_turn)
     { motor(0, -60.0); motor(1, 100.0); }
  else if(action == right_turn)
     { motor(0, 100.0); motor(1, -60.0); }
  else if(action == soft_right_turn)
     { motor(0, 50.0); motor(1, 0.0); }


  defer();
}


/**************************************************************
*
* ROUTINE TO CONTROL MOTOR FUNCTIONS                          *
**************************************************************
*/

void motor_control()
```

```
{
 while(1)
 {
  if(ir_sense)              /* check for IR sensor redings */
   direction(ir_command);
/*     else if(bump_switch)  */     /* check for bump sensor readings */
/*       direction(bump_command); */
/*    else if(pyro_sense)  */      /* check for pyro-sensor readings */
/*     direction(action); */
  else if(find_fire)
   direction(look_for_fire);
  else direction(stop_moving);     /* stop */
  defer();
 }
}

/************************************************************
*
* RANDOMNLY RUN AROUND AND LOOK FOR A FIRE                   *
*************************************************************
*/
void find_the_fire()
{
 while(1)
 {
  look_for_fire = forward;
  find_fire = 1;
  sleep(10.0);

  look_for_fire = soft_right_turn;
  sleep(3.0);
  defer();
 }
}

/************************************************************
*
* IR_SENSOR COLLISION AVOIDANCE ROUTINE                  *
*************************************************************
*/
```

```
void IR_sensors()

{
  int bumperir_right,far_right,bumperir_left,far_left,thr0,thr1;

      poke(0x7008,0xff);         /* turn on IR Led's */
      while(1) {
            thr0 = 100;
            thr1 = 100;
            far_left = analog(0);
              bumperir_left = analog(1);    /* number represents PE0-PE3 pins
43-49 */
              bumperir_right = analog(2);
            far_right = analog(3);

            if (far_right > thr0)
             { ir_sense = 1; ir_command = left_turn; }

            if (far_left > thr0)
             { ir_sense = 1; ir_command = right_turn; }

            if (bumperir_left > thr0)
            {  ir_sense = 1; ir_command = right_turn; }

            if (bumperir_right > thr1)
            {  ir_sense = 1; ir_command = left_turn;  }

            if (bumperir_left == bumperir_right)
               ir_sense = 1; ir_command = backward;  }

            if ((bumperir_left < thr0) && (bumperir_right < thr0))
               {ir_command = forward; ir_sense = 0; }

            if ((far_left < thr0 ) && (far_right < thr0))
               {ir_command = forward; ir_sense = 0; }
      }
         defer();
}
/*************************************************************
*******
```

```
* BUMPER SENSORS ROUTINE                                    *
*************************************************************
*******/
/*
void bumper()
{
  while(1)
  {
      if (analog(5) <= 1)
        {
         bump_switch = 1;
         bump_command = right_turn;
         sleep(1.5)
         bump_command = forward;
        }

      if (analog(6) <= 1)
        {
         bump_switch = 1
         bump_command = left_turn
         sleep(1.5);
         bump_command = forward;
        }
      if (analog(7) <=1)
        {
         bump_switch = 1;
         bump_command = backwards;
         sleep(3.0);
         forward();
        }
      defer();
     }
*/

void main()
{
  PORTD = 0x1008;
  DDRD = 0x1009;
  IR_emmit = 0x7000;
```

```
  stop_moving = 0;
  forward = 1;
  backward= 2 ;
  right_turn = 3;
  left_turn = 4;

  ir_sense = 0;
  bump_switch = 0;
  pyro_sense = 0;
  find_fire = 0;
  look_for_fire = 0;

  ir_command = 0;

  initialize();
  sleep(3.0);
  charge();
  start_process(IR_sensors());
  start_process(motor_control());
  start_process(find_the_fire());

  looney_tune();

/* initialize global variables */

}
```