OSCAR
(One Smart Car, an Autonomous Robot)

By Ernesto Cividanes

Fall 2001

IMDL 5666

Table of Contents:

Preface

Thank you to Dr. Arroyo, Dr. Schwartz, Aamir Qaiyumi and Scott Nortman for all their help and efforts this semester. They are all brilliant people who have helped me either this semester or in past semesters to get to this point. All of them deserve much of the credit for the success of OSCAR and I hope them the best. Thank you.

Abstract

OSCAR is One Smart Car, an Autonomous Robot that finds the room with the brightest light, senses for Anthrax and Anthrax incubators by detecting smoke and or heat. If smoke and or heat is detected, their corresponding lights go off. However, if both smoke and heat are detected then an alarm will go off simulated by flashing lights.

Executive Summary

Using four infrareds, 2 bump switches, 3 CdS cells and 2 motors, this robot uses find a room and using a smoke and heat detector, will warn the user that there is either smoke, heat, both or none in the room.

Introduction

Currently the United States is under attack by people that are trying to terrorize our lives. Their intent is clear and the solution is to be prepared and avoid harming any American lives in the process. It is the intent of this robot to aide those people that are responsible for dealing with the Anthrax scare. OSCAR has the ability to enter a room, detect Anthrax and Anthrax incubators. Although the sensors are crude, OSCAR paves the way for new technologies and ideas for further experimentation.

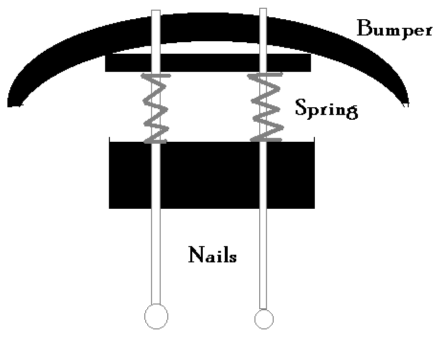
Integrated System

OSCAR uses the MC68HC11 as its microcontroller. This chip controls or receives input from the following:

System	**Qty.**	**Function**
MC68HC11	1	Microcontroller
Infrareds	4	Obstacle Avoidance
MC74HC4051(analog mux)	1	Extra Pins for the CdS cells
Motor Drivers	2	Front and rear motor driver
Smoke detector	1	Detect smoke
Heat detector	1	Detect heat
Bump switch	2	Finish testing and obstacle avoidance

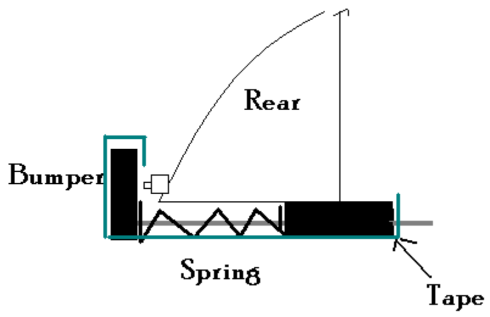
The 7805 is not controlled by the HC11 and does not receive input from it, however it does power the IRs and it is important to mention because without it, the board will continue to reset itself.

Mobile Platform
The RC car "The Shifter" from Radio Shack is the mobile platform for OSCAR. (For future projects, the motor drivers already in the car should be tested and tried to be implemented into the design). The aesthetics of this project are very important because it tries to keep the platform as untouched as possible which creates large problems when maneuvering. For example, OSCAR does not have the ability to move at a 90 degree turn. It however, must move backwards for a certain period of time, giving it enough room to turn and then move forward keeping in mind that when turning, it does not need to be "shy" of the wall unless it is too close to no longer turn. The front wheels of OSCAR turn very little so it is beneficial to open the frame of OSCAR and cut away some of the plastic that keeps it from moving.



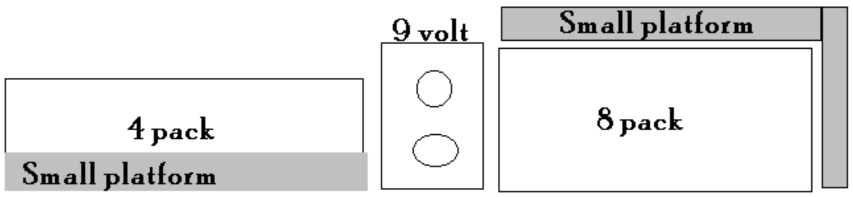
When installing the smoke detector, simply put it on top of the car and use the already made hole for the "Shifter" antenna and place the wires in there. For the heat detector, care needs to be given to the window because it is difficult to cut away the right amount of window needed. This can be done with power tools being careful not to crack the window. The rest of the fork is lights that fit comfortable at the front windshield of the car and no further changes need to be made.

The bump switches are by far the hardest to install than any other part of OSCAR. Tape seems to work the best when putting the bumper back after it has been made. Look at the figure on the right.



OSCAR has two motor drivers installed on the sides of it. Two regular screws hold them. The infrareds are position in the front and by the top of the front wheel cover. This configuration allows for the front bump switches to be disregarded and compensated by programming. The smoke detector is on top of OSCAR allowing the sensor to work as best as possible, which is the same as the heat detector. It is positioned at the top rear so it can be heated easily and not melting the smoke detector. The CdS cells are positioned at the front below the frame of the car to hide them as best as possible. For these to work correctly, they need to be covered some so regular tubing will work just fine. All of the switches are placed at the rear of OSCAR to allow easier programming, debugging, and charging. On the rear right side sits the 7805 and the analog mux is sitting at the bottom of OSCAR where the originally battery used to sit.
The rest of the of the circuitry is wire wrapped to allow for a cleaner design and sits inside the plastic covering. The power lights of the car are placed on the rear wheel cover and the alarm LEDs are mounted at the top of the car using a small board cut out and wrapped in electric tape. They are held down by glue.

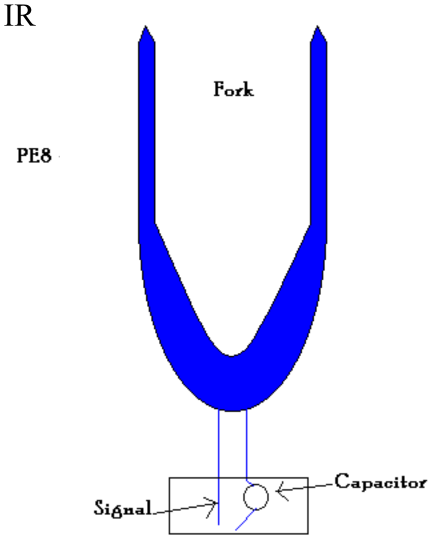
All three battery packs of OSCAR as situated on top of the inner plastic covering. A small platform holds the 8 pack from moving back and forth when car is moving. Also, some plastic taped down to the frame of the inner covering can hold the 4 pack down.



The most important aspect of the platform is to continue to use wire wrap because without it, the robot becomes an incredible mess. Also, until the program seems to run well, do not make a large hole to put in the wall IRs. Taping them until a good angle has been decided for them is a plus..

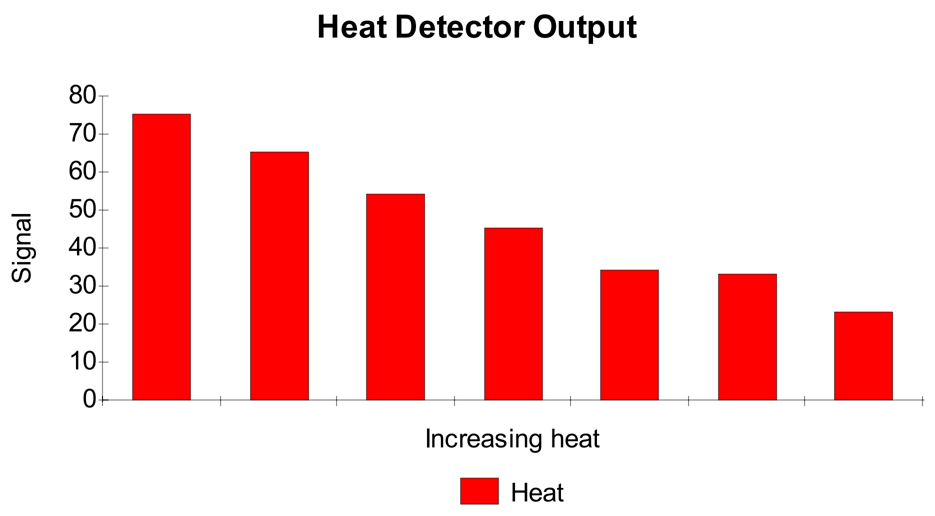
Actuation
The robot moves around using the original motors that the RC car provided. Since these motors draw a large amount of current, it is important to use good motor drivers that can drive at least an output of 3 amps. The motor drivers are free from National.com and receive four inputs, PWM, direction, +5 and ground. The PWM is controlled by an output compare(See Appendix) with a large period of 60000 for both the front and back. The direction can be any port and it is simply a digital output. To move left and right, there needs to be a degree of control because turning to hard could damage the wheels. Speed control is important here because it keeps the car from breaking the plastic case and it does not drain the battery as fast. A slow speed to turn is recommended and when straightening, a fast jerk in the opposite direction for a short period of time works well. Testing this time makes the programming easier.
Testing whether you have the right motor driver is a plus. Speed control is plausible and very important. If the motors receive a digital signal to the PWM, the car will run faster than the IR's can detect things. The motor drivers have their own batteries because if the car slows down, it will only slow down and not reset the board.

Sensors
The sensors of this robot are pretty much straight forward. The four IRs are from Sharp and already provide themselves with the clock and correct frequency so all that is needed is to receive the input. These new IRs work great when using them for obstacle avoidance. Testing the IRs is necessary because they can vary some between one and the other. To compensate for the front bump switches, if one IR is very close but the wall one is also close, then back up. It does take some time and OSCAR does sit there trying to move forward but it will soon back up and move away. (Using front bumpers on OSCAR would ruin the aesthetics)



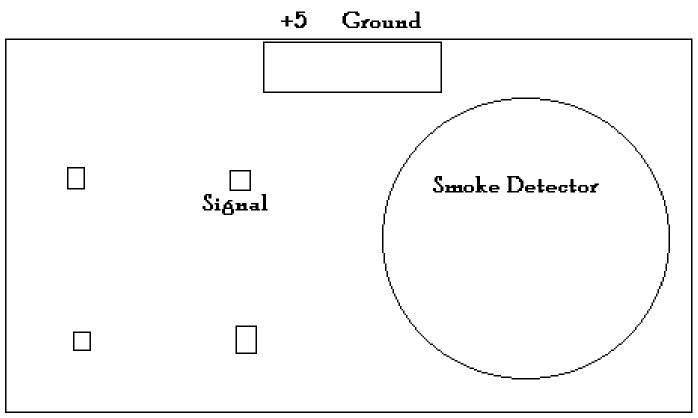
The CdS cells are used as any other. In the program, if the analog input is larger than the other, go in that direction. (See Appendix) Since there are only enough analog inputs for the IRs, the analog mux is used to add ports. Choose between one and the other by choosing the right combination on the select lines. See the figure on the right for a connection option. The "Heat" input will be discussed below.

The heat detector is very easy to use. It is the Electronic Thermometer Meatfork from Walmart. The signal easily is taken from the bottom of the fork and fed into the Mux as shown above. The signal that comes out, stays constant at room temperature and decreases as the temperature rises. As shown to the right, the signal is merely a wire that runs out of the fork and can be extended to reach the mux. The output signal is shown in the next page as a graph.

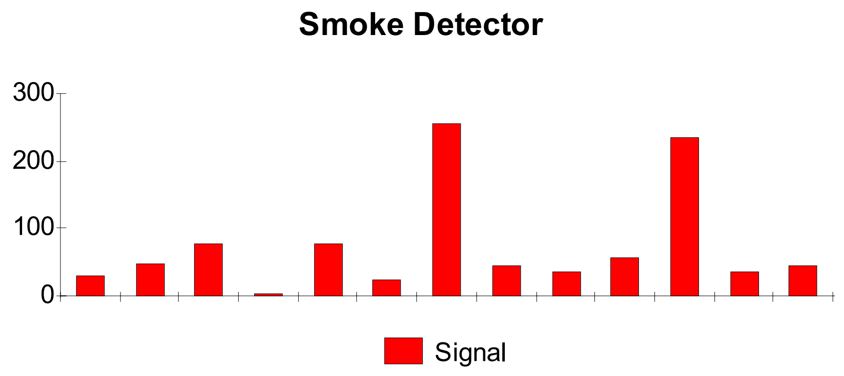


This signal is easy to read because the programming is easy. If a specific heat index is wanted, pick an analog signal corresponding with that temperature and added to the code. The lights that come with fork turn on at the signal 30 so OSCAR waits for that signal before turning on the lights.

Family Guard Smoke Alarm is the smoke detector for OSCAR. The signal from the smoke detector is also easy to read but it must be done several times before the right one is picked out. After detaching the speaker from the smoke detector, the signal is taken from the output to the speaker. The signals give 4 options: an 9V, 8V, 3V and ground. It is best to grab the 3 V because it guarantees the signal will not damage the board. The signal that is taken is then directly put into an analog port. This analog signal is easy to use also because it peaks out around 255 if there is an alarm. To test the smoke alarm, all that is needed is to use the test button.
(NOTE: The signal is sitting at the bottom of the smoke detector.)



The signal need is shown to the below. The peaks show when there is smoke around. This signal is not always easy to use because it happens for a brief period of time so when testing to see if there is a signal, the code must be easy, quick and loop until something happens.



The signal will not peak until the test has been set or there is smoke, however it will still have this pattern when the alarm has been set.

Behaviors
The behaviors of OSCAR are find Anthrax in a specified area and its incubators. To find the specified area, OSCAR uses the CdS cells to find the brightest spot in the room. This is merely a small simulation of what more money and tools that are more sophisticated can accomplish.
Once it finds the area, it will detect for anthrax. Reason why this works with a smoke detector is the smoke alarm has radioactive material in it. When that radioactive material interacts with an electrically neutral material such as Anthrax, it ionizes it and that ionized air is what sets off a smoke alarm. Also, when there is an incubator, there is heat. Therefore, OSCAR also detects heat using its heat detector.
Another behavior is obstacle avoidance. When OSCAR is put inside of an arena with obstacles, its first priority to move is the CdS cells but it will turn only if there are no obstacles in front. Obstacle avoidance proved to be difficult because of the limited speed control. When OSCAR is moving, it needs time to see a wall infront of it, However, it needs less time if it is moving slowly. So to make it shy is important if it is moving fast but not at slow speeds. A solution to this is turning off the motor after running for some time. However, a better solution would be to use an optical mouse because it opens up many different ways of perfecting speed control and direction.
Using the optical mouse, OSCAR could know if it has been moving and whether it needs to slow down. Also, it could determine its speed and decrease the its own speed. Optical mouse would make this project a huge success. Turning is also important. The original RC car had a sensor that would tell the car to stop turning. That would be useful because currently the motor continues to move until it is turned off.

CLOSING

OSCAR moves around a room avoiding obstacles and on back up, it might run into an obstacle because the rear bumpers are set to reset the test and not to avoid obstacles. It will find the light and stop in front of it and begin testing. The heat and smoke detector work just fine because of the simplicity of its design. OSCAR is limited by the fact that it does not have good speed control and can not slow down after some time because it varies with the charge of the battery and how long it has had to move forward. The batteries do not die as quickly as the others because there are motor drivers and voltage regulators in places where lot of amperage is drawn. Switches make the design easy to use and debug. The aesthetics is what makes OSCAR so much fun to work with however, all the challenges that it presented required many hours of work. Hopefully in the future, someone can implement a mouse into OSCAR and use better sensors for more sensitive temperatures. The front bumpers could help the design of OSCAR however, the next step is definitely the mouse.

TIPS:

1. Try using RC car's motor drivers
2. Try using the front turn motor sensor
3. Wire wrap everything
4. Good switches well placed.
5. Test everything separately
6. Do not add IRs into hardware until a good position is tested
7. Bypass capacitors need to be well placed before powering IRs
8. When connecting wires from the outside of the car to the inside, use male and female headers for everything. (Allows removal of frame from the rest of the car)
9. Keep the back two screws untouched to secure the car when the frame is shut.
10. Use black tape for anything that needs tape. Works well and hides itself.

```
/////////////////////////////////////////////
//Program:     WORKS1.c                     //
//Description: OBSTACLE AVOIDANCE AND CDL   //
// Date:        November 28, 2001           //
/////////////////////////////////////////////

#include <stdio.h>  //Standard includes
#include <mil.h>
#include <hc11.h>
#include <analog.h>

extern void _start(void); //This MUST be placed here RIGHT after the
includes, it is for the reset vector

#define DUMMY_ENTRY (void (*)(void))0xFFFF //This is a void function
declaration for the interrupt vector table
#define PERIOD 30000  //User defines
#define PERIOD2 60000  //User DEFINED FOR TOC2
    // SENSORS
#define BUMPER     analog(0)
#define RIGHT_IR   analog(2)
#define LEFT_IR    analog(3)
#define WLEFT_IR   analog(4)
#define SMOKE      analog(5)
#define WRIGHT_IR  analog(6)
#define CDL       CandH(1)
#define CDR       CandH(2)
#define CDC       CandH(3)
#define HEAT      CandH(4)

#pragma interrupt_handler TOC2_isr, TOC3_isr;  //This is where
interrupt service routines are specified

void init_pwm(void);   //Function Prototypes
void TOC3_isr(void);
void TOC2_isr(void);
void WAIT();
void MOVE();
void TURN();
int CandH();
//Global variables
int forward,backward;
int right, left;
int duty3;
int duty2;
int SPEED;
int STARTSPEED;
int BACKSPEED;
int flag;
int counter;
int turnFlag;
int tcounter;
int rightFlag;
int loop;
int count;
int heatFlag;
int smokeFlag;
```

```
void main (void){      //Beginning of main routine
  char clear[]= "\x1b\x5B\x32\x4A\x04";
      // INITIALIZATIONS
      init_pwm();
      init_analog();
      init_serial();
      SET_BIT(DDRD, 0x04); // TURN PORT
      SET_BIT(DDRD, 0x10); //
      SET_BIT(DDRD, 0x08); // LED SMOKE
      SET_BIT(PACTL, 0x08); // LED GREEN/HEAT

      duty3 = 0;
      duty2 = 0;
      right = 1;
      left = 2;
      forward = 2;
      backward  = 1;
      STARTSPEED =80;
      SPEED = 50;
      BACKSPEED = 60;
      flag = 0;
      counter = 0;
      turnFlag = 0;
      tcounter = 0;
      rightFlag = 2;

      while(1){
            MOVE(STARTSPEED, forward);
            WAIT(10);
            if(flag >= 100) {
                  MOVE(SPEED, forward);
            }
            if(RIGHT_IR > 35) {
                  if(WRIGHT_IR < 15) {
                        TURN(25, right);
                        rightFlag = 1;
                  }
                        else if(WLEFT_IR < 15) {
                        TURN(25, left);
                        rightFlag = 0;
                  }
                  tcounter = 0;
            }
            else if(LEFT_IR > 35) {       // SEES SOMETHING
                  WAIT(10);
                  if(WLEFT_IR < 15) {
                        TURN(25, left);
                        rightFlag = 0;
                  }
                  else if(WRIGHT_IR < 15) {
                        TURN(25, right);
                        rightFlag = 1;
                  }
            }
            WAIT(10);
            if(LEFT_IR > 35 && RIGHT_IR > 35
            && WLEFT_IR > 20 && WRIGHT_IR > 20) {
                        TURN(0,0);
                        MOVE(BACKSPEED, backward);
                        WAIT(600);
                        flag = 0;
                  if(WRIGHT_IR < 15) {
                        TURN(25, right);
                        rightFlag = 1;
                  }
                  else if(WLEFT_IR < 15) {
                        TURN(25, left);
                        rightFlag = 0;
                  }
            }
            if(turnFlag >= 90) {
                  tcounter++;
                  turnFlag = 0;
                  if(tcounter >= 4) {
                        if(rightFlag == 1)      {
                              TURN(90, left);
                        }
                        else TURN(90, right);
                        WAIT(20);
                        TURN(0, 0);
                        rightFlag = 2;
                  }
            }
        }//End while(1)
}//End main

/////////////////////////////////////
//// IF BOTH SMOKE AND HEAT PRESENT//
//// ALARM TOGGLES BETWEEN LEDS    //
/////////////////////////////////////

void ALARM(void) {
      loop = 0;
      CLEAR_BIT(PORTA, 0x08);
      CLEAR_BIT(PORTD, 0x08);

      while(BUMPER < 50) {
            CLEAR_BIT(PORTA, 0x08);
            WAIT(50);
            SET_BIT(PORTD, 0x08);
            WAIT(50);
            SET_BIT(PORTA, 0x08);
            WAIT(50);
            CLEAR_BIT(PORTD, 0x08);
            WAIT(50);
            count++;
      } // END OF WHILE
      count = 0;
}
```

```
/////////////////////////////////////
//// TESTS SPECIAL SENSORS HERE /////
/////////////////////////////////////

void SENSORS(void) {

      while(BUMPER < 50) {

      // TOGGLE GREEN TO DENOTE TESTING
            count = 0;
            heatFlag = 0;
            smokeFlag = 0;
            loop = 1;

            while(count < 5) {
                  SET_BIT(PORTA, 0x08);
                  WAIT(50);
                  CLEAR_BIT(PORTA, 0x08);
                  WAIT(50);
                  count++;
            }

            CLEAR_BIT(PORTD, 0x08);

            while(loop) {

                  smokeFlag = 0;

                  // DETECT SMOKE AND TURN ON SMOKE LED
                  if(SMOKE > 200) {
                        SET_BIT(PORTD, 0x08); // SMOKE LED
                        smokeFlag = 1;
                  } // END OF IF

                  // IF NO SMOKE AND HEAT, TURN ON GREEN LED
                  if(HEAT < 30 && smokeFlag != 1) {
                        SET_BIT(PORTA, 0x08);
                        heatFlag = 1;
                  } // END OF IF

                  if(heatFlag == 1 && smokeFlag == 1) {
                        ALARM();
                  }

            }// END OF WHILE(loop)

      }// END OF WHILE(1)

}// END OF SENSOR TESTS
```

```
//////////////////////////////////////
////  CHOOSE USING MUX WHICH  SENSOR //
////  TO USE BETWEEN THE CdS CELLS   //
////  AND THE HEAT DETECTOR          //
//////////////////////////////////////
int CandH(int temp) {
      if(temp == 1) {
            // LEFT
            CLEAR_BIT(PORTA, 0x10);
            CLEAR_BIT(PORTD, 0x10);
            return(analog(1));
      }
      else if(temp == 2) {
            // RIGHT
            CLEAR_BIT(PORTA, 0x10);
            SET_BIT(PORTD, 0x10);
            return(analog(1));
      }

      else if(temp ==3) {
            // CENTER
            SET_BIT(PORTA, 0x10);
            CLEAR_BIT(PORTD, 0x10);
            return(analog(1));
      }

      else if(temp == 4) {
            SET_BIT(PORTA, 0x10);
            SET_BIT(PORTD, 0x10);
            return(analog(1));
       }

}

// MOVE FORWARD OR BACKWARD ROUTINE

void MOVE(int speed, int direction) {

      duty3 = speed;
      if (direction == 1) {SET_BIT(PORTA, 0x08);}           // FORWARD
      else if(direction == 2) {CLEAR_BIT(PORTA, 0x08);}     // BACKWARD

}

void TURN(int speed, int direction) {

      duty2 = speed;
      if(direction == 1){SET_BIT(PORTD, 0x04);}       // RIGHT
      else if (direction == 2){CLEAR_BIT(PORTD, 0x04);}     // LEFT
      if(direction == 0) {duty2 = 0;}                       // STOP

} // ENF OF TURN

void WAIT(int time) {
      int temp;
      temp = 200;
      while (time > 0) {
            time--;
            while(temp > 0) {
            temp--;
            }
            temp = 200;
      }
}

void init_pwm(void){

        INTR_OFF();     //Turns off interrupts

        SET_BIT(TMSK1, 0x40);       //Set up TOC2
        SET_BIT(TCTL1, 0x80);
        CLEAR_BIT(TCTL1, 0x40);

        SET_BIT(TMSK1, 0x20);       //Set up TOC3
        SET_BIT(TCTL1, 0x20);
        CLEAR_BIT(TCTL1, 0x10);

        INTR_ON();

}//End init_pwm


//////////////////////////////////////////////////
//INTERRUPT SERVICE ROUTINE THAT HANDLES TURNS
//////////////////////////////////////////////////

void TOC2_isr(void){

        int temp = 0;
        if(duty2 != 0) {
                  turnFlag++;
            }
        CLEAR_FLAG(TFLG1, 0x40);
        CLEAR_BIT(CFORC, 0x40); // THIS WAS NOT USED IN ORIGINAL TOC2

        temp = (duty2 / 100.0) * PERIOD2;

        if(temp < 500){

               CLEAR_BIT(TCTL1, 0x40);
               SET_BIT(CFORC, 0x40);
        }

        else if(temp > (PERIOD2 - 500)){

               SET_BIT(TCTL1, 0x40);
               SET_BIT(CFORC, 0x40);
        }

        else if(TCTL1 & 0x40){

               CLEAR_BIT(TCTL1, 0x40);
               TOC2 += temp;
        }

        else {
               SET_BIT(TCTL1, 0x40);
               TOC2 += (PERIOD2 - temp);
        }

}//End TOC2

//////////////////////////////////////////////////
//INTERRUPT SERVICE ROUTINE: FORWARD OR BACKWARD
//////////////////////////////////////////////////

void TOC3_isr(void){

        int temp = 0;
            flag++;

        CLEAR_FLAG(TFLG1, 0x20);
        CLEAR_BIT(CFORC, 0x20);

        temp = (duty3 / 100.0) * PERIOD;

        if(temp < 500){

               CLEAR_BIT(TCTL1, 0x10);
               SET_BIT(CFORC, 0x20);
        }

        else if(temp > (PERIOD - 500)){

               SET_BIT(TCTL1, 0x10);
               SET_BIT(CFORC, 0x20);
        }

        else if(TCTL1 & 0x10){

               CLEAR_BIT(TCTL1, 0x10);
               TOC3 += temp;
        }

        else {
               SET_BIT(TCTL1, 0x10);
               TOC3 += (PERIOD - temp);
        }

}//End TOC3

#pragma abs_address:0xffd6

/* change the above address if your vector starts elsewhere */

void (*interrupt_vectors[])(void) =

        {
```

```
        DUMMY_ENTRY,   /* SCI, RS232 protocol */
        DUMMY_ENTRY,   /* SPI, high speed synchronous serial*/
        DUMMY_ENTRY,   /* Pulse accumulator input edge */
        DUMMY_ENTRY,   /* Pulse accumulator overflow */
        DUMMY_ENTRY,   /* Timer overflow */
        DUMMY_ENTRY,   /* TOC5 */
        DUMMY_ENTRY,   /* TOC4 */
        TOC3_isr,      /* TOC3 */
        TOC2_isr,       /* TOC2 */
        DUMMY_ENTRY,   /* TOC1 */
        DUMMY_ENTRY,   /* TIC3 */
        DUMMY_ENTRY,   /* TIC2 */
        DUMMY_ENTRY,   /* TIC1 */
        DUMMY_ENTRY,   /* RTI */
        DUMMY_ENTRY,   /* IRQ */
        DUMMY_ENTRY,   /* XIRQ */
        DUMMY_ENTRY,   /* SWI */
        DUMMY_ENTRY,   /* ILLOP */
        DUMMY_ENTRY,   /* COP */
        DUMMY_ENTRY,   /* CLMON */
        _start         /* RESET */

        };
#pragma end_abs_address
```

/* ADD THE FOLLOWING TO MAIN ROUTINE TO USE THE CdS CELLS SEPERATE

```
while(1){


                  WAIT(10);
                  if(CDC > 200) {
                        TURN(0, right);
                        MOVE(0, forward);
                        SENSORS();
                  }
                  else if(CDL > CDR) {
                        MOVE(40, forward);
                        TURN(25, left);
                        WAIT(200);
                        TURN(90, right);
                        WAIT(20);
                        TURN(0, left);
                  }
                  else if(CDR > CDL) {
                        MOVE(40, forward);
                        TURN(25, right);
                        WAIT(200);
                        TURN(90, left);
                        WAIT(20);
                        TURN(0, right);
                  }
        }//End while(1)
}//End main
*/
```

References:
Radio Schack
    http://www.radioshack.com/
Family Guard Smoke Alarm
    BRK Brands, Inc.
Electronic Thermometer Meatfork
    Wal-Mart
Sharp GP2D12 Infrared Ranger
    http://www.acroname.com/robotics/parts/R48-IR12.html
Mux, mc74hc4051
    http://mil.ufl.edu/~datasheets/Analog_MUX_DEMUX/
National Semiconductor
    http://www.national.com/
Mekatronix
    http://www.mekatronix.com/

Future Work:
UART for Optical Mouse
    http://www.national.com/parametric/0,1850,225,00.html
Optical Mouse
    Programming
        http://mega.ist.utl.pt/~fjds//mousepolltut.html
    Protocol
        http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/PS2/ps2.htm
    Interfacing
        http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/mouse/mouse.html