University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory

WoMAN
(Work-Oriented Mobile Autonomous Neat-freak)

Name: Eric Donnelly
Date: 8/8/02
TAs: Tae Choi
Uriel Rodriguez
Instructor: Dr. A. A Arroyo
Dr. Eric M. Schwartz

Table of Contents

Executive Summary

WoMAN is designed for the purpose of vacuuming floors independently. Instead of developing complex algorithms to increase cleaning efficiency, which could only be used in a controlled environment, this robot design used a code based off of random numbers, routing the robot in different directions even when the same environment is used repeatedly. The robot may cross over its own path, but it will most likely never take the exact path twice. Therefore, WoMAN will eventually clean an entire room, no matter what its size or shape. Obstacle avoidance is achieved by using IR as well as bump sensors, to provide two levels of obstacle detection. The robot's reaction to these stimuli is also based on random numbers, in order to provide a great deal of robustness, and ensures that the robot cannot easily get "stuck", where one sensor sends the robot in one direction, then another sensor sends the robot back the same direction.

In order for WoMAN to be independent, it must be able to replenish its energy. In other words, the robot must be able to recharge its batteries. This is achieved by having the robot monitor its battery voltage, and when it drops below a predetermined threshold, it must begin searching for a place to recharge. WoMAN's recharging station's use a light bulb to act as a beacon. WoMAN uses CdS cells to detect this light beacon, and follow it until it makes contact with the recharging station. Once the robot is recharging its batteries, it monitors their temperature to detect when they are fully charged. When the batteries temperature goes above a predetermined threshold, the robot disconnects from the station, and begins to vacuum again.

Abstract

WoMAN is an autonomous vehicle that randomly moves around a room vacuuming the floor as it goes. WoMAN monitors its battery and returns to its recharging station when it becomes low on power. WoMAN will demonstrate obstacle avoidance of walls as well as objects in the room, including people.

Introduction

Nobody likes a dirty floor. However, the solution to this is a task that few enjoy. Having to take the time and effort to sweep or vacuum a floor when you could be doing something much more productive (like watching TV) is highly undesirable. Hiring a maid to do this job for you can be very costly; therefore, another solution is needed. WoMAN is the solution. WoMAN roams around a room sucking up any dust or dirt on the floor. It avoids any obstructions in the room (walls, boxes, people), and when it gets low on battery power, it shuts down its vacuum (to save power) and finds its way back to a recharging station, where it recharges its batteries and sets off on another cleaning rampage. It can be left to do this all day long, or can be switched on at night to clean your floors while you sleep, and stay out of your way.

Integrated System

The robot uses a cylindrical platform for optimum mobility. This allows the robot to turn in place. Mounted on this are two hacked Futaba S3004 servos to drive the system. The brains of WoMAN is an Atmel ATmega163 microcontroller mounted on a MegaAVR development board. This chip has 3 PWM channels, two of which are used to control the speed of the servos. Two Infrared (IR) sensors (Sharp GP2D12) are used to detect the presence of an obstacle in the robot's vicinity. These sensors are mounted cross-eyed at 20 degrees from straight forward. Six bump switches surround the platform to sense the presence of obstacles from all directions, and act as a back-up for the IR sensors. Two CdS cells are used for finding its recharging station, which uses a 75-Watt light bulb as its beacon. Two LM35DT temperature sensors are used to monitor the temperature of the batteries while being recharged. The robot connects with the recharging station through at plate mounted on the front of its body and its bumper rim. A block diagram of this system can be seen in Figure 1 on the following page.

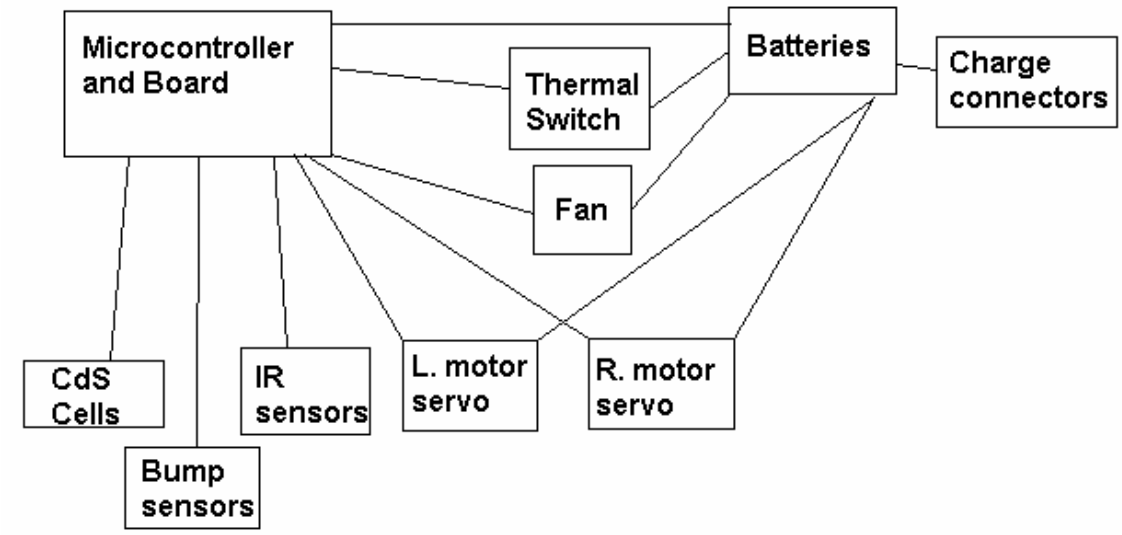

Fig. 1- Functional block diagram

Mobile Platform

The platform used for WoMAN is cylindrically shaped, in order to decrease the chances of the robot getting stuck where it cannot escape. This shape allows WoMAN to spin in place, enabling itself to easily escape from corners. The platform is 11" in diameter and 5" tall. The center of mass of the robot is near the rear of the platform, with the fan for the vacuum and the two battery packs, one for the fan and the controller, respectively. A nylon bulb is mounted under the rear of the platform to support WoMAN (along with the 2 wheels connected to the driving servos). CdS cells are mounted along with the IR detectors at the front of the platform. The temperature sensors are mounted on the battery packs, and adhered with heat-sink compound for good thermal conductivity. Around the rim of the robot are the six bump switches. A 12-Gauge insulated copper wire is glued to the switches and encircles the entire platform. This allows the robot to detect obstacles

from any direction. The front quadrant of this wire is stripped, exposing the copper. This wire is also connected the entire system's ground. This allows the bumper rim to act as one of the conduction plates to be connected to the recharging station. The other conduction plate is mounted flat on the front of the robot, under the IR detectors and CdS cells. This plate is connected to the positive terminal of the controller's battery pack. The placement of the conduction plates was decided after a dimensional analysis of the robot. Described in Cartesian coordinates; the robot moves in only two dimensions, along the xy-plane. Therefore all points on the robot along the z-axis will remain constant. Arbitrarily choosing any two points along this axis for placement of the conduction plates allows for predictable points of contact to the recharging station. This method is superior to any other DC conduction plate method placed in the xy-plane, such as the floor, which is vulnerable to reversed polarity across the plates. However, an AC in solution the xy-plane is acceptable, but requires a rectifier circuit onboard the robot.

Actuation

Two hacked Futaba servos are used to drive and steer the robot. PWM, which is integrated into the ATmega163, is used to determine the speed and direction the motors will be running. The fan uses a separate battery pack to isolate it from the microcontroller. The fan is controlled by the microcontroller via a TTL logic relay to shut it down when the battery is low. Another relay is connected between the battery packs to allow the microcontroller to bridge them together once the fan is turned off. This gives the servos twice as much power to find the recharging station before the batteries die. This circuit is shown in Figure 2 below.

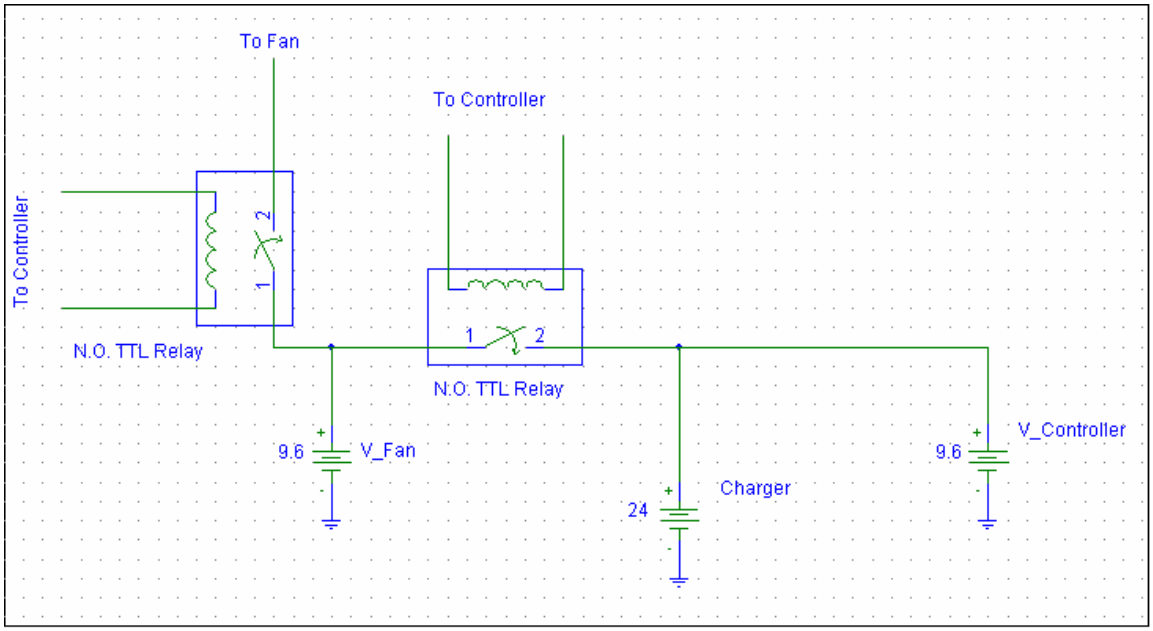

Figure 2. On-board relay and recharge circuitry

Sensors

The four types of sensors used in this robot are: IR detectors, Bump sensors, CdS cells, and temperature sensors. The CdS cells are used to detect a light beacon, in order for the robot to find its recharging station.

Infrared

Two infrared detectors are used for this robot. These are mounted on the front of the robot and angled 20 degrees inward and detect the presence of objects in front of it as shown in Figure 3. These sensors are the Sharp GP2D12 IR detectors which output an analog voltage relating to the amount of IR light bouncing back from an object. The closer the object (to a point), the greater amount of IR.

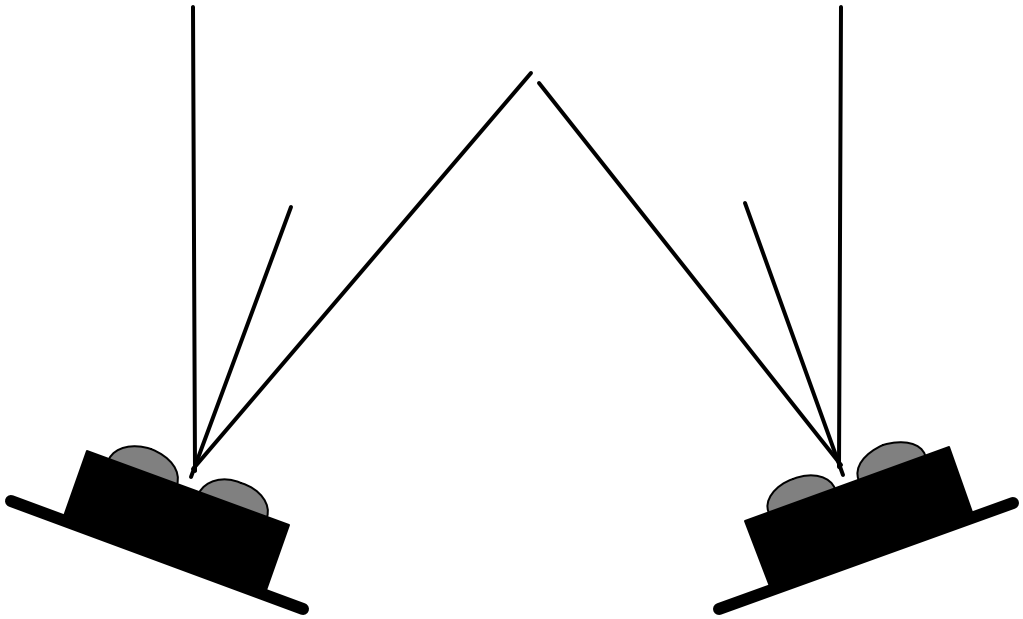

Figure 3. IR positioning

Bump Sensors

Six bump sensors are mounted around the robot to detect if the robot runs into anything. These are normally-open push-button switches with a solid 12-gauge copper wire surrounding the perimeter of the platform as shown in Figure 4. Each switch is connected to a specific pin on Port B of the Atmel microcontroller. The switches are all tied to ground. Port B of the microcontroller has its internal pull-up resistors set. So in order to detect if WoMAN hits something, the microcontroller looks at the logical values of the Port B pins. If it sees a '1', then nothing is hit. If a '0' appears on one of the pins, then that bump switch is currently triggered and the robot can steer away accordingly.

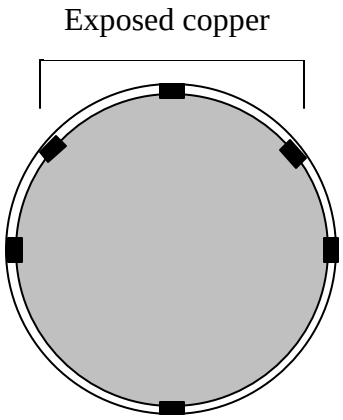

Figure 4. Bump Sensor Layout

CdS Cells

Cds cells are used to locate the recharging station. The recharging station has a 75-Watt light bulb attached to it, low to the ground, for the CdS cells to detect. When WoMAN is low on batteries, it begins to look for a light source. Since these cells are mounted inside pen caps, their light-sensitivity becomes very directional. By using two CdS cells, the direction of the light and therefore the recharging station can be determined.

Experimenting with different resistor values for the voltage divider circuits, I found that for indoor applications with indoor lighting conditions, a 33kO resistor provides the optimum sensitivity. Testing was does done on the mounted CdS cells to determine its ability to find the light beacon. A 75-Watt light bulb mounted inside a desk lamp, giving it a 100 degree swath, was placed on the ground. The robot was placed at various distances away and the CdS cell values were read at various rotations of the robot. The data collected is shown below in Table 1.

	0 degrees		20 degrees right		40 degrees right		20 degrees left		40 degrees left	
feet away	Left	Right	Left	Right	Left	Right	Left	Right	Left	Right
10	52	54	120	143	158	160	108	88	186	189
8	39	39	83	91	160	161	87	55	178	163
6	26	28	66	103	146	158	85	58	171	164
4	8	14	8	40	39	160	110	16	165	103
2	6	9	3	108	19	171	96	4	146	20
Note: the ambient room lighting gave values of 186 and 189, respectively.										

Table 1

Temperature Sensors

When WoMAN's batteries get low on power, it finds its way to a recharging station to recharge its batteries via the CdS cells. A temperature sensor on each battery pack monitors the temperature of the batteries while recharging. If they get too hot, the batteries are recharged and the robot disconnects from the charging station and continues to clean. These were mounted with heatsink compound to get good thermal conduction from the battery pack to the temperature sensors. To find what values the temperature sensors read from the batteries the sensors were sampled during a 16 minute recharge. The results are shown in figure 5.

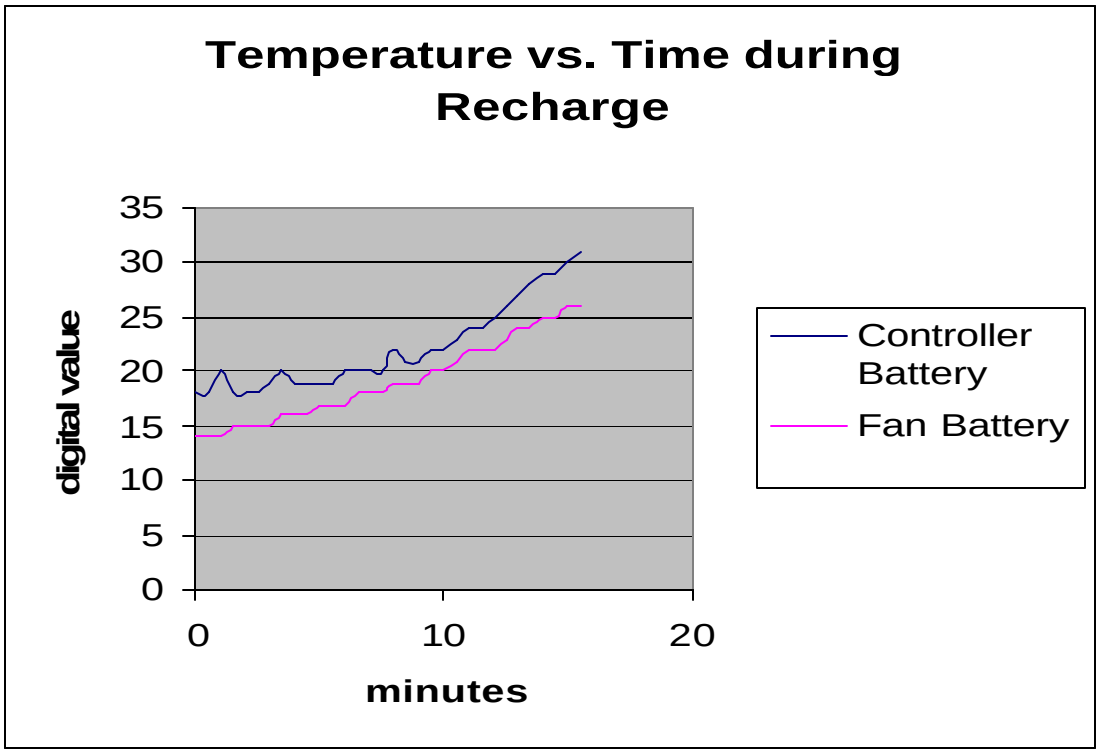

Figure 5

Note: the starting voltages for the controller and fan batteries were 7.68V and 9.79V, respectively. The final voltages were 11.14V and 11.31V respectively.

Behaviors

WoMAN demonstrates obstacle avoidance by using its infrared and bump sensors. When an object is detected, it chooses a new random direction to follow by taking the Mod of the Timer/Counter0 register. Note: the timer/counter used to create the random number MUST be different from the timer/counter used for the sensor interrupt service routine if a timer overflow interrupt is used. Otherwise, the random number will be determined at the same timer value each time. Using random numbers ensures that the ROBOT won't take the exact same path every time it is activated, so it will most likely cover the entire

room to be cleaned using a very simple programming algorithm. When WoMAN's batteries get low, determined by a voltage divider connected to one of the A/D converter pins, it shuts down its fan to conserve power, bridges the batteries together to give more power to the servos, and searches for its recharging station. WoMAN uses a "dizzy search" method to find the recharging station. It starts monitoring its light sensors, looking for a high amount of light (determined by a threshold value). Every 10 seconds it spins in place for one full rotation (2 seconds), checking the light sensors every 10 ms. This gives the robot a resolution of 1.8 degrees. Once the light source is found, WoMAN locks onto the light source and drives to it. The robot stops when a large voltage is present across its batteries (meaning it has made contact with the recharging station). While recharging, if WoMAN comes disconnected from the station, it drives forward and even begins searching for light until it re-establishes contact with the recharging station. WoMAN monitors the temperatures on the battery packs while recharging. If the fan's battery pack is finished recharging first (which should happen since the fan draws less current than the controller), the relay bridging the batteries is shut off. Once the controller's battery pack is fully recharged (determined by its temperature), the robot disconnects from the recharging station, turns its fan back on, via the relay, and continues vacuuming the room.

Conclusion

This project set out to create an autonomous robot which could vacuum a room, avoid obstacles, and recharge itself when its power was low. WoMAN, the robot designed to accomplish these tasks, does this, although modestly. For a first design this robot is a success, for lessons are learned throughout the design process, and an even more robust second version could be created at the same expense of WoMAN. Some improvements could be made with the platform design. Having the bumper rim at a lower height would greatly improve the robot's obstacle avoidance capabilities as shoes and other objects with low heights are commonly laying around on household floors. Protection of the circuitry would also greatly improve the design, as all of WoMAN's circuitry resides exposed on the top of its platform. This project as strengthened my belief that an independent household vacuum is feasible. Hopefully sometime in the near future the household chore of manual vacuuming will no longer be necessary, as it will be taken over by autonomous robots.

Documentation

Special thanks to Dr. Eric M. Schwartz, Dr. Antonio Arroyo, Tae Choi, Uriel Rodriguez, and Dr. David Bloomquist for their contributions to this project.

Atmel Corp., “Atmel AtMega163 Datasheet”, http://www.atmel.com/atmel/acrobat/doc1142.pdf.

Progressive Resources, “MegaAVR Development Board”, http://www.prllc.com/MegaAVR_Dev.pdf.

Appendix

Program Code

```
/*****************************************
Project : WOMAN
Version : 3.5
Date    : 6/17/2002
Author  : Eric Donnelly
Company :
Comments:Final Version
Interrupt used to read IR and bump, new random generator using mods
light sensing used, new interrupt routine based on priority sensors
***adding battery monitor, relays for battery bridge and fan
*** could be last version
*** heat monitors integrated also

Chip type           : ATmega163
Program type        : Application
Clock frequency     : 6.000000 MHz
Memory model        : Small
Internal SRAM size  : 1024
External SRAM size  : 0
Data Stack size     : 256
*****************************************

analog PORT definitions
0     right IR
1     left IR
2     left CdS
3     right CdS
4     bad?
5     uC temp sensor
6     fan temp sensor
7     voltage monitor   --- .078V per tick, 8.5V = 108, 12V= 155

BUMP SENSORS
PORTB
0     rear
2     left side
3     left front
4     front
5     front right
6     right

SERVOS
PORTD.5     Left
PORTD.4           Right

Relays
PORTD.2           FAN Enable
PORTD.6           Power Bridge Enable

Push-Button
```

```
PORTB.7           find recharge station
*/

#include <mega163.h>
#include <delay.h>


#define FIRST_ADC_INPUT 0
#define LAST_ADC_INPUT 7

#define L_EYE_TOLERANCE 60 //IR tolerance
#define R_EYE_TOLERANCE 60 //........
#define MOTOR_STOP 18
#define     LOW_BATT 108     //ADC.7 value
#define     RECHARGE 155      //ADC.7 value
#define CPU_TEMP 25     //battery temperature threshold's
#define FAN_TEMP 23     //.......
#define LIGHT_TOLERANCE 70

static short int speed_val_l = 0;         //current value
static short int speed_val_r = 0;
static short int speed_temp_l = 0;
static short int speed_temp_r= 0;
//unsigned char light_val_l= 0;
//unsigned char light_val_r = 0;
unsigned char random_num;
bit sense = 0;   //tells if something's detected
int temp;
int count = 5;
char batt_count = 0;
int i = 0;
bit home_lock = 0;
bit find_home = 0;
bit look = 0;

void battery(void);
void eyes(void);
void skin(void);
void light(void);
void drive(int, int);
void light(void);
int random(unsigned char);
void looky(void);

// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
if(look) return;
PORTC.1 = !PORTD.2;     //turn on LED if fan on
PORTC.0 = !PORTD.6;     //turn on LED if batteries bridged
if(count-- ==0){
      PORTC.7 = PORTC.7 ^ 1;
      PORTC.2 = !home_lock;   //turn on if home locked
      battery();
      light();
      eyes();
            skin();
```

```
      if(sense == 1){
        drive(speed_temp_l, speed_temp_r);
        if(!home_lock) delay_ms(200*(random(7)+2) );//do it for mx 1sec
        else delay_ms(200);
        sense = 0;
        PORTC = PORTC | 0b01110000;    //turn senses LEDs off
        }
        count = 3;
      }
}

unsigned char analog[LAST_ADC_INPUT-FIRST_ADC_INPUT+1];
#define ADC_VREF_TYPE 0xE0          //0x60 --external 5V, 0xE0--
internal 2.56V
// ADC interrupt service routine
// with auto input scanning
#pragma savereg-
interrupt [ADC_INT] void adc_isr(void)
{
#asm
    push r26
    push r27
    push r30
    push r31
    in   r30,sreg
    push r30
#endasm
register static unsigned char input_index=0;
// Read the 8 most significant bits
// of the AD conversion result
analog[input_index]=ADCH;
// Select next ADC input
if (++input_index > (LAST_ADC_INPUT-FIRST_ADC_INPUT))
   input_index=0;
ADMUX=FIRST_ADC_INPUT|ADC_VREF_TYPE|input_index;
// Start the AD conversion
ADCSR|=0x40;
#asm
    pop  r30
    out  sreg,r30
    pop  r31
    pop  r30
    pop  r27
    pop  r26
#endasm
}
#pragma savereg+

void woman_init(void){
// Input/Output Ports initialization
// Port A initialization
// Func0=In Func1=In Func2=In Func3=In Func4=In Func5=In Func6=In
Func7=In
// State0=T State1=T State2=T State3=T State4=T State5=T State6=T
State7=T
PORTA=0x00;
```

```
DDRA=0x00;

// Port B initialization
PORTB=0xFD;
DDRB=0x00;        //PORTB.1 is connected to LED
// Port C initialization
PORTC=0x7F;       //leave right light on
DDRC=0xFF;
// Port D initialization

PORTD=0x00;
DDRD=0x74;

//Timer 0 set to clock
TCCR0=0x01;
TCNT0=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 23.438 kHz
// Mode: Ph. correct PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0xA1;
TCCR1B=0x04;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=18;
OCR1BH=0x00;
OCR1BL=18;

TCCR2=0x00;
ASSR=0x00;
TCNT2=0x00;
OCR2=0x00;

GIMSK=0x00;
MCUCR=0x00;

TIMSK=0x04;

UBRRHI=0x00;

ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 93.750 kHz
// ADC Voltage Reference: Internal 2.56V
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=FIRST_ADC_INPUT|ADC_VREF_TYPE;
ADCSR=0xCE;
```

```
}


void drive(short int speedl, short int speedr){ //speed from -2 to 2
   if(speedl > 2) //can't overdrive servos
   speedl =2;                        //.8 sec max total to change speeds
   if(speedl < -2)
   speedl = -2;
   if(speedr > 2)
   speedr = 2;
   if(speedr < -2)
   speedr = -2;
      while(speed_val_l != speedl || speed_val_r != speedr){
      if(speed_val_l > speedl){
            OCR1AL = OCR1AL -1;
            speed_val_l--;
      }
      if(speed_val_l < speedl){
            OCR1AL = OCR1AL +1;
            speed_val_l++;
      }
      if(speed_val_r > speedr){
            OCR1BL = OCR1BL +1;     //installed backwards
            speed_val_r--;
      }
      if(speed_val_r < speedr){
            OCR1BL = OCR1BL -1;
            speed_val_r++;
      }
      if(!home_lock) delay_ms(100);       //.2s delay on each increment
to protect motor
                                    //unless following light

      }
}


int random(unsigned char range){          //generates random numbers
between 0-range, max 254
      range++;    //so not 0-range-1 since using mods
      return TCNT0 % range;
}

void eyes(void){
 //   PORTC = PORTC ^ 0x80;
      if(home_lock) return;
      if(analog[1] > L_EYE_TOLERANCE || analog[0] > R_EYE_TOLERANCE){
            sense = 1;
            PORTC.6 =0;       //IR LED
            PORTC.4 = 1;            //light LED off
            temp= random(3) - 2; //between -2 and 1
            if(analog[1] > analog[0] + 10) // some lee-way
            {     speed_temp_l=  temp;
                  speed_temp_r= temp + random(3)+1;   //always will
                                                      //make point away
                }                                     // from obstacle
```

```
            else if(analog[0] > analog[1] + 10) //some lee-way
            {     speed_temp_l=  temp + random(3)+1;
                  speed_temp_r= temp;                 //always will
                                                      //make point away
                }                                     // from obstacle

                else
                      {speed_temp_l = random(4) -2;   //turn in place
                                                      //if straight
                  speed_temp_r = -speed_temp_l;       //forward
            }
        }
}

void light(void){
if(((analog[2] < LIGHT_TOLERANCE) || (analog[3] < LIGHT_TOLERANCE)) &&
find_home){
            sense =1;
            home_lock = 1;
            PORTC.4 = 0;            //turn on light sense LED
            if(analog[2] < analog[3] - 20 && analog[3] >= 20)
            {           speed_temp_l =0;
                   speed_temp_r = 2;
                   return;
            }
            if(analog[3] < analog[2] - 20 && analog[2] >=20)
            {     speed_temp_r =0;
                  speed_temp_l = 2;
                  return;
            }
            if(analog[2] < analog[3] - 5 && analog[3] >=5)
            {     speed_temp_l =1;
                  speed_temp_r = 2;
                  return;
            }
            if(analog[3] < analog[2] - 5 && analog[2] >=5)
            {     speed_temp_r =1;
                  speed_temp_l = 2;
                  return;
            }
      }
      else home_lock = 0;
}

void skin(void){
      if(home_lock) return;
    if(PINB==125 || PINB==127 || PINB == 253 || PINB ==255) return;
    sense =1;
    PORTC.4 =1;         //light LED off
    PORTC.5 =0;         //skin LED
    PORTC.6 =1;         //IR LED off
    while(1){
      if(!PINB.4){   //front
            drive(-1, -1);    //back up
            delay_ms(300);
                  speed_temp_l = random(4) -2;        //turn in place
                                                      //if straight
```

```
            speed_temp_r = -speed_temp_l;       //forward
            break;
      }
      else if(!PINB.3){    //left front
             speed_temp_l= -1;      //right will reverse harder
             speed_temp_r= -2;
             break;
      }
      else if(!PINB.5){    //right front
             speed_temp_r= -1;         //left will reverse harder
             speed_temp_l= -2;
             break;
      }
      else if(!PINB.2){    //left side
             speed_temp_l= random(1)+1;
             speed_temp_r= 0;
             break;
      }
      else if(!PINB.6){ //right side
             speed_temp_r= random(1)+1;
             speed_temp_l= 0;
            break;
      }
      else if(!PINB.0){  //forward!!
             speed_temp_r= random(1)+1;
             speed_temp_l= random(1) +1;
            break;
      }
      else{
      return;
      }
      }
}

void battery(void){
      if(find_home ==1){
            PORTC.3 = PORTC.3 ^ 1;        //toggle LED
            return;     //no need to check
      }
      if(analog[7] > RECHARGE){       // if accidentely on
                                    //station get away
            drive(-2, -2);
            delay_ms(200);
            drive(2, -2);
            delay_ms(1000);
      }
      if((analog[7]< LOW_BATT)|| PINB.7){       //possible low battery
            if((++batt_count == 10) || PINB.7){       //so no flukes
                  find_home = 1;          //find home
                  PORTD.2 = 0;            //turn off fan
                  PORTD.6 = 1;            //bridge batteries
            }
      }
      else batt_count =0;                 //reset if not consecutive
}

void charge(){
```

```
      //found_home = 1;       //so won't look at sensors
       //   delay_ms(200);    //let wheels stop moving
      PORTC = 0xFF;
      while(1){
      while(analog[7] > RECHARGE){        //if recharging
            if(PORTC == 0x00)
                  PORTC = 0xFF;
            else
            PORTC= PORTC << 1;


            DDRB.1 = 1;       //turn on recharge light
            TCCR1B = 0x00;          //don't look at sensors
            drive(0,0);
            if(analog[6] > FAN_TEMP)      //if fan battery toohot, cut
      //bridge
                  PORTD.6 = 0;
     //                 PORTC.0 = !PIND.7;
            if((analog[5] > CPU_TEMP) || PINB.7){     //if CPU charged
//or override
                  home_lock = 0;
                  find_home = 0;          //stop searching
                  PORTD.6 = 0;            //unbridge
                  PORTD.2 = 1;            //turn on fan
               //       PORTC.0 = !PIND.7;
            //           PORTC.1 = !PIND.2;
                  drive(-1, -1);    //back up
                  delay_ms(500);
                  drive(2,-2);            //turn around;
                  delay_ms(1000);
                  TCCR1B=0x04;            //look at sensors again
                  DDRB.1 = 0;       //turn off recharge light
                  PORTC = 0xFF;
                  return;
            }
      delay_ms(65);
      }
      PORTC = 0xFF;
      DDRB.1 = 0;       //turn off recharge light
      TCCR1B=0x04;            //enable timer for PWM
      drive(1,1);       //reconnect
      }

}

void looky(void){
      while(home_lock){
      drive(2,2);
      if((analog[7] > RECHARGE) && find_home){  //on station
                        drive(0,0);     //stop
                        charge();   //goto charge routine
            }
      }
if(find_home){
      look = 1;   //don't look at senses
            drive(-2,2); //spin in place to look for light for 2 secs.
```

```
            for(i = 0; i < 120; i++){ //look for light every 20ms 150
//for carpet
            PORTC.3 = PORTC.3 ^ 1;        //toggle LED
                  light();
                  if(home_lock){
                        drive(speed_temp_l,speed_temp_r);
                        look = 0;
                        break;
                  }
                  else delay_ms(20);
            }
            look = 0;
      }

}

void main(void)
{
woman_init();

while(PINB & 0b00000001){                 //tap on back  to start
    if(PORTC == 0x7F)                    //do cool light show
      while((PINB & 0b00000001) & (PORTC != 0xFE)){
            PORTC = PORTC >> 1;
            PORTC.7 = 1;
      delay_ms(40);
      }
    else{
      PORTC = PORTC << 1;
      PORTC.0 = 1;
      }
            delay_ms(40);

}
PORTC= 0x7F;
// Global enable interrupts
#asm("sei")

PORTD.2 = 1;            //turn on fan
drive(2, 2);
find_home = 0;
while (1)
      {
      //    PORTC = PORTC ^ 0xFF;
      looky();

            random_num = random(125);     //number between 0 and 125
            while(random_num-- != 0){         // max drive in one
//direction =12.5s
                  drive(2,2);
                  delay_ms(100);
                  if(home_lock) looky(); //if seeing light
        }

//        while(home_lock);         //if found home

        drive(random(4)-2, random(4)-2);  //change directions
```

```
            random_num = random(11);
            while(random_num-- != 0){         // max change direction
//time =1.0s
                  if(home_lock) looky();
                  delay_ms(100);
        }
       }
}
```