W.E.S.L.E.Y.

Waste Eliminating System for Lazy Engineering Youths

Final Report
December 9, 2003
John Mercado

<u>Table of Contents:</u>

Abstract.. 3
Executive Summary.. 3
Introduction.. 3
Integrated Systems.. 4
Mobile Platform.. 4
Actuation.. 5
Sensors.. 5
Behaviors.. 6
Experimental Layout and Results.. 6
Conclusion.. 7
Acknowledgements.. 8
Appendices.. 8

Abstract:

Wesley is an autonomous mobile trash can built to help lazy people throw things away without getting up from wherever they may be lounging. When Wesley's master calls him, Wesley leaves his home position in search of his master. Homing in on his master's location, Wesley moves skillfully towards his destination, avoiding obstacles along the way. Upon arriving at his master, Wesley opens his lid and waits to receive trash. When Wesley is done receiving the refuse, he returns to his home elsewhere in the room to wait for another call from his master.

Executive Summary:

Wesley was designed to help a lazy or immobile person throw items in the trash. Wesley is capable of navigating towards his user while avoiding obstacles. Once Wesley arrives at his destination, he will open his lid and wait for trash to be thrown in. Once either trash is thrown in, or a long time has passed, Wesley will turn around and go to his home base and wait for the next call.

The microprocessor utilized in this robot is an Atmel ATmega128. The custom built board houses the ATmega128 mini header board, along with ports for all the parts of the robot. A holder for 6 AA batteries is built into the board.

Two Sharp GP2D120 infrared sensors, mounted on the front right and front left areas of the platform make it possible for the robot to avoid obstacles while moving. They are capable of sensing obstacles from 4 to 30 cm away. These sensors, combined with obstacle avoidance software are a very effective way to keep Wesley from crashing.

The beacon system for this robot is somewhat unique in that there are two infrared beacons on the same frequency. This is accomplished by controlling the beacons from the robot. The main board has an RF transmitter, and each beacon has a RF receiver. Depending on the situation, the robot will transmit a signal that turns on just one of the beacons. It then uses infrared sensors to track down the beacon and move towards it.

It is important for Wesley to know when trash has been thrown in, otherwise he may leave without accomplishing his task, or he might stay longer than is necessary. The trash is detected using a spread laser break beam. The laser is projected onto three cds cells. If the light going to any of the three is interrupted, the robot will know that an object has passed through the plane of the laser.

Introduction:

There is no better feeling than wasting valuable time lying on a comfortable couch and watching some TV. Standing up to throw something in the trash unnecessarily interrupts the quality time that one's rear end should be spending with the couch. Wesley was designed to keep this interruption to laziness from occurring.

Wesley's functions are made possible though the use of various sensors and actuators controlled by a microprocessor. The details of Wesley's design are presented in this report.

Integrated System:

Wesley's design consists of several systems using an Atmel ATmega128 for processing. A block diagram of the systems involved is provided in Figure 1.

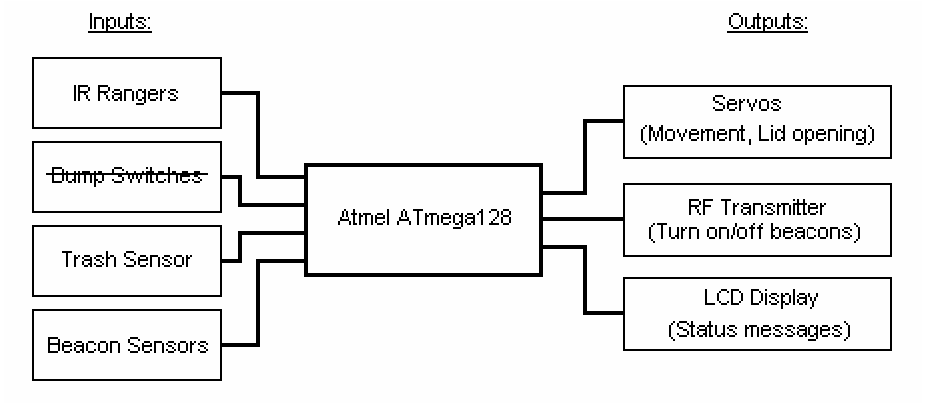
Inputs:
IR Rangers
~~Bump Switches~~
Trash Sensor
Beacon Sensors
Atmel ATmega128
Outputs:
Servos (Movement, Lid opening)
RF Transmitter (Turn on/off beacons)
LCD Display (Status messages)

Figure 1

Obstacle avoidance is accomplished mainly through the use of infrared rangers. These devices tell the processor how close an object is to the robot. The processor then decides what to do and tells the servos to move accordingly. In the original design, bump switches were going to be used because the infrared rangers have blind spots which may lead to collisions. Towards the end of the project, it was clear that the infrared rangers did the job well enough without the use of bump switches, so the bump switches were omitted from the final design.

The trash sensor is a custom sensor that will detect when a piece of trash has been thrown into the trash can. This sensor will send a signal to the processor and Wesley will know that his job is done and he can go home.

The top mounted infrared beacon sensors are essential for the operation of this robot. These sensors find which direction the beacon is relative to the robot. This signal is sent to the processor, which decides which way to turn in order to keep moving towards its destination.

There will be two beacons in the room, one at Wesley's home base, and one at Wesley's master. These beacons will both operate at the same frequency, so they can't be on at the same time. The processor will decide when it is appropriate to have one beacon or the other on. Using the RF transmitter, Wesley will transmit an instruction that will determine which beacon is active.

Information on what Wesley is doing at any given moment can be viewed on Wesley's LCD display. The LCD panel will display useful debugging messages such as "Waiting for Trash" or "Beacon lost."

Mobile Platform:

The platform for Wesley is a round platform made of light wood. A round design is ideal for this robot because there are no corners which could bump into obstacles and make turning difficult. The platform is made of two pieces made of balsa wood. The top piece of the platform is attached to the small round trash can. It is nearly identical to the bottom piece of the platform. The bottom piece has a specially designed hole to hold the PVC pipe which runs up the rear of the robot. The bottom also has holes for the circuit board which consists of the microprocessor and other electronics essential to the robot. The servos and IR sensors are attached to the bottom half of the platform. The only thing attached to the top piece of the platform is the trash bin, which makes it easy to remove the top half to plug in wires and make adjustments to the electronics. Many parts of the platform are held together with hot glue. A lot of hot glue was used during the construction of this robot.

Actuation:

Wesley uses three servos for actuation. All three servos are GWS high torque servos. Two servos are hacked to allow continuous motion, and the other is in its original condition. The two hacked servos act as motors for the wheels. They are both connected to the 16 bit Timer1 port of the ATmega128, working in inverted PWM mode.

The third servo, the un-hacked one, is used to open and close the lid of the trash can. This servo is un-hacked so that the processor can control how open the lid is. This servo is connected to the 16 bit Timer3 port of the ATmega128, working in inverted PWM mode.

Sensors:

The sensors on this robot can be separated into three systems: obstacle avoidance, beacon finding, and trash detecting. All sensors are connected to the A/D port of the ATmega128.

The obstacle avoidance sensors consist of two Sharp GP2D120 infrared rangers and a few bump switches. The infrared rangers detect the distance of an object between 4cm and 30cm. After thorough experimentation, it was possible to ascertain which voltages read on the ADC pin corresponded to specific distances. These signals are sent to the processor which will be running an obstacle avoidance algorithm.

The beacon finding system was originally supposed to utilize three hacked Lite-On infrared sensors, one pointing left, one center, and one right. The week before demo day, I realized that three of my eight analog ports were not functioning. The center sensor had to be sacrificed, and new software had to be written. The 56.8 kHz Lite-On sensors were hacked using Michael Hatterman's method to get an analog signal from the sensor. Wesley selects which beacon is active using a Laipac TLP-434 RF transmitter, and then it is up to the beacon finding sensors and software to find the signal and follow it. The beacons consist of a Laipac RLP-434 RF receiver, a decoder, a MOSFET used as a switch, and a 555 timer circuit to output a square wave with a 56.8 kHz frequency (see Special Sensor Report). The 555 timer circuit used on both of the beacons is a modified version of the circuit found on the IMDL website (http://www.mil.ufl.edu/imdl/handouts/timer.pdf).

Trash detection is accomplished through the use of a laser and three collimated photo-resistors. The resistors were modified using heat shrink tubing, in order to collimate their line of sight to block out other light. When the laser's path to any one of the resistors is blocked, the processor will detect that an object has been put into the trash bin. The laser does not have a lens on it, so it projects a line. This line, when projected onto the round inside of the trash bin, creates an arc of about 75°. This makes it possible to have a large area of detection, allowing only very small objects, placed perfectly into the blind spot to slip by unnoticed. A diagram of the plane of the laser break beam can be seen in Figure 2 below. The software checks the sensors reasonably fast, so a piece of trash has to be thrown very quickly in order to slip by unnoticed.

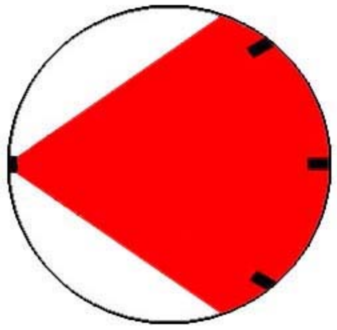

Figure 2

Behaviors:

In order to be a good trash can robot, Wesley must be able to successfully perform a few behaviors. Wesley's behaviors consist of obstacle avoidance, moving towards his master (beacon following), trash collecting, and returning home (beacon following).

To keep Wesley from crashing into objects, an obstacle avoidance algorithm was written. The algorithm basically tells the robot that if an obstacle is detected on either side of the robot, turn away from the obstacle, and if the obstacle is straight ahead, turn sharply to one of the sides. The algorithm is a bit more complicated than this, with instructions added here and there to improve accuracy.

Locating his master is an important function for Wesley. An algorithm was written to take input from the beacon finding sensors and adjust Wesley's direction of motion to stay headed in the right direction. This algorithm also is able to determine when the robot is close enough to the beacon to begin the trash collecting behavior.

Wesley's trash collecting behavior is a three step procedure. Once Wesley is close enough to the beacon, the lid controlling servo opens the lid to the trash can. Wesley then waits for trash. If no trash is thrown in after a long period of time, Wesley gives up and goes home. Once trash has been detected, the lid closes, and Wesley begins his returning home behavior.

The returning home behavior is nearly identical to the moving towards the master behavior. The only difference is that a different beacon is being followed. The RF transmitter mounted on Wesley is responsible for selecting which beacon is active.

Experimental Layout and Results:

Several experiments have been performed along the way to make it easier to arrive at the final product. Through the course of the semester, it was necessary to test and demonstrate the functionality of the microprocessor board, LCD display, IR sensors, obstacle avoidance behavior, and the laser break beam sensor.

The microprocessor board test simply consisted of running a program that lit up LEDs to show that the processor boots and runs code. The code was written in assembly. This test had successful results on the first try.

The LCD display test was combined with the IR sensor test. The test consisted of running a program that read a value from the IR sensor, and displaying the value on the LCD display, along with how the software interpreted the data. The software interpreted the data as either far, close, or very close based on the thresholds hard coded into the program. This test had successful results on the first try.

The obstacle avoidance algorithm was tested by letting Wesley roam my living room. The first test had very bad results. The turns were too tight and often times the wheels would get snagged on the obstacle that the robot was trying to avoid. It was clear that this problem was caused by the wheels being foolishly placed over an inch away from the base of the platform. I adjusted the wheels so that they were very close to the platform and conducted further tests. The adjustments proved to solve the problem, and Wesley is now able to avoid most obstacles. Two problems that still exist are small blind spots in the IR "vision", and Wesley's inability to escape from a corner.

Additional tests were conducted on the IR beacon system. To test the RF system I wrote a program that turned on and off the beacon using RF. Then I took the beacon and walked around my apartment to test the range and accuracy of the RF transmitter/receiver pair. I was amazed to see that the transmission was successful up to and probably beyond the entire distance from one end of my apartment to the other (about 15 yards through walls). It was clear that this wouldn't be a problem, so I didn't bother to test further distances.

The beacon sensors were tested by hooking them up to a bread board and moving the beacon around in front of them while measuring the output voltage from the sensors. The maximum distance appeared to be about 5 to 6 feet, with a viewing angle of about 28°. The viewing angle was calculated holding a beacon 2 feet away and moving it side to side to see when the voltage significantly dropped. The maximum distance to either side from the center was about 6 inches. Using the equation: $\theta = 2 * \tan^{-1}(0.5/2)$, I was able to find out the viewing angle of the sensor.

The remainder of the tests were simply turning the robot on and observing it in different situations. I must have done this over one hundred times. It makes easier to understand the code that you wrote when you see it in action. Sometimes I will see the robot doing one thing and wonder what it's doing and why it's doing that, then after looking closely at the code, I am able to understand it and tweak the code to get the results I want.

Conclusion:

I believe that I have accomplished all the goals that I have set for this project. Wesley is capable of performing all the behaviors proposed in the beginning of the semester. There have been a few obstacles along the way, including the loss of three analog inputs causing me to redesign the beacon following hardware and software, and the random resetting problem that has plagued me all semester (due to poor software). Luckily I was able to adapt the software to eliminate the resets, and to get by with the analog inputs that I had. This was a very challenging and time consuming project, but I learned more in this class than in any other class I have taken so far.

Acknowledgements:

I would like to thank the following, without them I would not have been able to complete this project:
Dr. Arroyo - for the wealth information throughout the semester and for making IMDL possible
Dr. Schwartz - for teaching me so many useful things in EEL 4744
Uriel Rodriguez and Louis Brandy - for their many suggestions and expertise
Michael Hatterman (IMDL Spring 2002) - for his method of hacking the Lite-On IR sensors
The University of Florida - for all the parts I took
Nathan at Sparkfun.com - for his help with the RF modules
Jameco.com - for selling everything and shipping fast
Hot glue - for holding everything together

Appendices:

Disclaimer: I am not, nor have ever claimed to be a good programmer. In fact, I didn't know C before taking this class. The following code is ugly. It was written a while ago, and then modified at least a hundred times since then. Some parts may not make any sense at all. These parts are probably remnants of code that I have since decided against using. The bottom line is: it works. I don't want to clean it up out of fear of making it not work. So here is the source code for Wesley, in its unpleasant entirety:

```
// wesley.c
// by John Mercado
// IMDL Fall 2003
// final version
// completed 12/8/03

#include <mega128.h>
#include <math.h>
#include <delay.h>
#asm
   .equ __lcd_port=0x15
#endasm
#include <lcd.h>
bit randbit1;
bit randbit2;
bit randbit3;
unsigned char currentL = 0x0D;
unsigned char currentH = 0x24;
signed int motorL = 0;
signed int motorR = 0;
unsigned char decision = 0;
unsigned char lastdecision = 0;
unsigned char decidecount = 0;
unsigned char started = 0;
int calib = 0;
int calib1;
int calib2;
int calib3;
unsigned int zeroL;
unsigned int zeroR;
char turn = 0;

#define ADC_VREF_TYPE 0x00
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

char obstacle(void);
char getir(unsigned char);
void motors(signed int,signed int);
void servo(unsigned char);
char decide(char,char);
void zeroir(void);
void homespin(void);
void gotobeacon(void);
void homebeacon(char);
void destbeacon(char);
void gettrash(void);

void main(void)
{
// Declare your local variables here
TCCR0=0x01; // turn on timer0 for random #
PORTA=0x80;
DDRA=0xFF;
PORTB=0x00;
DDRB=0xC0;
PORTC=0x00;
DDRC=0x00;
PORTD=0x00;
DDRD=0x00;
PORTE=0x00;
DDRE=0x20;
PORTF=0x00;
DDRF=0x00;
PORTG=0x00;
DDRG=0x00;

OCR1AH=0x27;
```

```
OCR1AL=0x10;
TCCR1B=0x11;
// LCD module initialization
#asm("cli");
delay_ms(500);
lcd_init(24);

lcd_gotoxy(0,0);
lcd_putsf("Start");

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AREF pin
// ADC High Speed Mode: Off
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x83;
SFIOR&=0xEF;

while(1){
homebeacon(0); //everything off
destbeacon(0);
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Home Beacon OFF");
lcd_gotoxy(0,1);
lcd_putsf("Dest Beacon OFF");
delay_ms(1000);
zeroir();
destbeacon(1); // destination beacon on
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Dest Beacon ON");
gotobeacon(); // go to destination
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Reached Destination");
destbeacon(0); // destination beacon off
gettrash(); // wait for trash
delay_ms(1000);
zeroir();
turn = 0;
homebeacon(1); // home beacon on
homespin();
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Home Beacon ON");
gotobeacon(); // return home
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Reached Home");
motors(-2,2); // timed spin around
delay_ms(2500);
motors(0,0);
}
```

```
}

void gettrash(void) {
int cdsvalue;
int cdsvalue1;
int cdsvalue2;
char gottrash = 0;
unsigned int waiting = 0;

servo(1);
calib1 = read_adc(2); //get current light level
calib2 = read_adc(2);
calib3 = read_adc(2);
calib = ((calib1+calib2+calib3)/3);
lcd_clear();
lcd_gotoxy(0,0);
lcd_putsf("Waiting for trash");
while ((gottrash == 0) && (waiting < 0x0FF0)) {
      cdsvalue1 = read_adc(2);
      cdsvalue2 = read_adc(2);
      cdsvalue = ((cdsvalue1+cdsvalue2)/2);
      if (cdsvalue < (calib - 15)) {
            gottrash = 1;
            lcd_clear();
            lcd_gotoxy(0,0);
            lcd_putsf("Got it!!!");
            servo(0); }
      else {
            waiting++;
            delay_ms(100); }
      }
if (waiting >= 0x0FF0) {
      lcd_clear();
      lcd_gotoxy(0,0);
      lcd_putsf("Nothing!!!");
      servo(0); }

}

void homebeacon(char onoff) {
if (onoff==1) {
      PORTA=0b10111111;
      delay_ms(1);
      PORTA=0b00111111; }
else {
      PORTA=0b10110000;
      delay_ms(1);
      PORTA=0b00110000; }
delay_ms(750);
}
void destbeacon(char onoff) {
if (onoff==1) {
      PORTA=0b10101111;
      delay_ms(1);
      PORTA=0b00101111; }
else {
      PORTA=0b10100000;
      delay_ms(1);
      PORTA=0b00100000; }
delay_ms(750);
}

void zeroir(void) {
unsigned int irbL1;
unsigned int irbL2;
unsigned int irbL3;
unsigned int irbL4;
unsigned int irbR1;
unsigned int irbR2;
unsigned int irbR3;
unsigned int irbR4;

irbL1=read_adc(3);
irbR1=read_adc(6);
irbL2=read_adc(3);
irbR2=read_adc(6);
irbL3=read_adc(3);
irbR3=read_adc(6);
irbL4=read_adc(3);
irbR4=read_adc(6);
zeroL = (irbL1+irbL2+irbL3+irbL4)/4;
zeroR = (irbR1+irbR2+irbR3+irbR4)/4;
}

void homespin(void) {
char gotir = 0;
unsigned int irbL;
unsigned int irbL1;
unsigned int irbL2;
unsigned int irbL3;
unsigned int irbL4;
unsigned int irbR;
unsigned int irbR1;
unsigned int irbR2;
unsigned int irbR3;
unsigned int irbR4;

motors(-2,2);
delay_ms(400);
while (gotir==0) {
      irbL1=read_adc(3);
        irbR1=read_adc(6);
        irbL2=read_adc(3);
        irbR2=read_adc(6);
        irbL3=read_adc(3);
        irbR3=read_adc(6);
        irbL4=read_adc(3);
        irbR4=read_adc(6);
        irbL=(irbL1+irbL2+irbL3+irbL4)/4;
        irbR=(irbR1+irbR2+irbR3+irbR4)/4;
      if ((irbL > (zeroL+7)) || (irbR > (zeroR+7))) {
            gotir = 1; }
      }
delay_ms(300);
motors(0,0);
}

void gotobeacon(void) {
unsigned int irbL;
unsigned int irbL1;
unsigned int irbL2;
unsigned int irbL3;
unsigned int irbL4;
unsigned int irbR;
unsigned int irbR1;
unsigned int irbR2;
unsigned int irbR3;
unsigned int irbR4;
char thereyet = 0;
char waitcount = 0;
char irL;
char irR;
char close = 0;

started=0;

while (thereyet<3) {
      irbL1=read_adc(3);
        irbR1=read_adc(6);
        irbL2=read_adc(3);
        irbR2=read_adc(6);
        irbL3=read_adc(3);
        irbR3=read_adc(6);
        irbL4=read_adc(3);
        irbR4=read_adc(6);
        irbL=(irbL1+irbL2+irbL3+irbL4)/4;
        irbR=(irbR1+irbR2+irbR3+irbR4)/4;
        if (irbL <= zeroL) {
            irbL = zeroL; }
      if (irbR <= zeroR) {
            irbR = zeroR; }
      if (waitcount >= 100) {
            waitcount = 0;
            started = 0;
            lcd_clear();
            lcd_gotoxy(0,0);
                  lcd_putsf("Waiting for beacon"); }
      else {
            irL=0;
            irR=0;
            if ((irbL-zeroL) > 160 || (irbR-zeroR) > 160) {
                  close = 1;
                  irL=getir(0);
                  irR=getir(1); }
            if ((close==1)&& ((irL>0) || (irR>0))) {
                  thereyet = thereyet+1;
                  motors(0,0); }
            else if ((irbL-zeroL) > 135 || (irbR-zeroR) > 135) {
                        lcd_clear();
                  lcd_gotoxy(0,0);
                        lcd_putsf("Aquired beacon signal");
                  thereyet = 0;
                  waitcount=0;
                  if ((irbL-zeroL) > ((irbR-zeroR)+10)) {
                        motors(0,1); }
                  else if (((irbL-zeroL)+10) < (irbR-zeroR)) {
                        motors(1,0); }
                  else if ((irbL-zeroL)==(irbR-zeroR)) {
                        motors(1,1); }
                  else {
                        motors(1,1); } }
            else {
                  thereyet = 0;
                  if ((irbL > (zeroL+5)) || (irbR > (zeroR+5))) {
                              lcd_clear();
                        lcd_gotoxy(0,0);
                              lcd_putsf("Aquired beacon signal");
                        started=1;
                        waitcount=0;
                        if ((irbL-zeroL) > ((irbR-zeroR)+15)) {
                              motors(1,2);
                              turn = obstacle();
                              }
                        else if (((irbL-zeroL)+15) < (irbR-zeroR)) {
                              motors(2,1);
                              turn = obstacle();
                              }
                        else if ((irbL-zeroL)==(irbR-zeroR)) {
                              motors(2,2);
                              turn = obstacle();
                              }
                        else {
                              motors(2,2);
                              turn = obstacle();
                              }}
                  else {
                        if (started==1) {
                              lcd_clear();
                              lcd_gotoxy(0,0);
                              lcd_putsf("Beacon lost!");
                              motors(1,1);
                              delay_ms(750);
                              if (turn == 1) {
                                    motors(1,-1); }
                              else if (turn == 2) {
                                    motors(-1,1); }
                              else {
                                    homespin();
                                    motors(1,1); }
                              delay_ms(750);
                              turn = obstacle();
                        }
                        else {
                              motors(0,0);
                              waitcount=waitcount+1; }
                        }} }

      }
started = 0;
}

char obstacle(void) {
char ir_result1;
char ir_result2;
char randchar;
char whichway;

randchar = TCNT0/64;
randbit1=TCNT0/128;
delay_ms(randchar);
randchar = TCNT0/64;
randbit2=TCNT0/128;
delay_ms(randchar);
randchar = TCNT0/64;
randbit3=TCNT0/128;
delay_ms(randchar);
randchar = TCNT0;  //get semi random bits and char
ir_result1 = getir(0);
ir_result2 = getir(1);
decision = decide(ir_result1,ir_result2);
if (decision==lastdecision && decision!=0) {
      decidecount++; }
else {
      decidecount = 0; }
if (decidecount<5) { // normal operation
      if (decision==8|decision==4) {
            motors(-1,-1);
            delay_ms((int)randchar+1000);
            TCCR1A=0x00; // make sure it doesnt keep going back.
            randbit1=(randbit1)*pow(-1,((int)randbit3+1)); // could be -1,0,1
            randbit2=(-1)*randbit1;
            motors(randbit1,randbit2);
            if (randbit1 > randbit2) {
                  whichway = 2; }
            else {
                  whichway = 1; }
            delay_ms((int)randchar+1000); }
      else if (decision==7|decision==6) {
            motors(-1,-1);
            delay_ms((int)randchar+1000);
            motors(1,-1);
            whichway = 2;
            delay_ms((int)randchar+1200); }
      else if (decision==5|decision==2) {
            motors(-1,-1);
            delay_ms((int)randchar+1000);
            motors(-1,1);
            whichway = 1;
            delay_ms((int)randchar+1200); }
      else if (decision==3) {
            motors(1,0);
            whichway = 2;
            delay_ms((int)randchar+1200); }
      else if (decision==1) {
            motors(0,1);
            whichway = 1;
            delay_ms((int)randchar+1200); }
      else {
            whichway = turn; } } //do nothing
else {  // its been repeating
      lcd_clear();
      lcd_gotoxy(0,0);
      lcd_putsf("I am confused!");
      motors(-1,-1);
      delay_ms(1000);
      TCCR1A=0x00;
      motors(1,-1);
      delay_ms(5000); }
lastdecision = decision;
return whichway;
}


char decide(unsigned char var1, unsigned char var2) {
unsigned char value=0;
if (var1==2) {
      if (var2==2) {  // 2,2 = case 8
            value = 8; }
      else if (var2==1) { // 2,1 = case 7
            value = 7; }
      else { // 2,0 = case 6
            value = 6; }}
else if (var1==1) {
      if (var2==2) {  // 1,2 = case 5
            value = 5; }
      else if (var2==1) { // 1,1 = case 4
            value = 4; }
      else { // 1,0 = case 3
            value = 3; }}
else {
      if (var2==2) {  // 0,2 = case 2
            value = 2; }
      else if (var2==1) { // 0,1 = case 1
            value = 1; }
      else { // 0,0 = case 0
            value = 0; }}
return value;
}



char getir(unsigned char adcpin) {
int adcresult;
int adcresult1;
int adcresult2;
int adcresult3;
int irout;
adcresult1 = read_adc(adcpin);
adcresult2 = read_adc(adcpin);
adcresult3 = read_adc(adcpin);
adcresult=(adcresult1+adcresult2+adcresult3)/3;
if (adcresult >= 370) {
      irout = 2;
      }
else if (adcresult >= 170 && adcresult < 370){
      irout = 1;
      }
else if (adcresult < 170){
      irout = 0;
      }
return irout;
}


void motors(signed int speedL, signed int speedR)
{
if (motorL==speedL && motorR==speedR) {
      // do nothing
      }
else {
      if ((motorL>0 && speedL>0)||(motorL<0 && speedL<0))
            {
            if ((motorR>0 && speedR>0) || (motorR<0 && speedR<0))
                  {}
            else {
                  TCCR1A=0x00;
                  delay_ms(500); } // wait, for protection
            }
      else {
                  TCCR1A=0x00;
                  delay_ms(500); } // wait, for protection
      motorR=speedR; // update globals with current speed.
      motorL=speedL;
      if (speedR==-1) {
              OCR1BH=0x23;
            OCR1BL=0xF0; // Period = 20ms, 1.6ms pulse
            TCCR1A=0x3F; // Start the PWM
              }
```

```
        else if (speedR==-2) {
              OCR1BH=0x24;
            OCR1BL=0xF0; // Period = 20ms, 1.6ms pulse
            TCCR1A=0x3F; // Start the PWM
            delay_ms(100);
                OCR1BH=0x23;
//          OCR1BL=0x28; // Period = 20ms, 2.0ms pulse
            OCR1BL=0x48;
            TCCR1A=0x3F; // Start the PWM
                }
        else if (speedR==1) {
                OCR1BH=0x24;
            OCR1BL=0x54; // Period = 20ms, 1.4ms pulse
            TCCR1A=0x3F; // Start the PWM
            }
        else if (speedR==2) {
                OCR1BH=0x24;
            OCR1BL=0x54; // Period = 20ms, 1.4ms pulse
            TCCR1A=0x3F; // Start the PWM
            delay_ms(100);
                OCR1BH=0x24; // was 25
//          OCR1BL=0x1C; // Period = 20ms, 1.0ms pulse
            OCR1BL=0xFC;
            TCCR1A=0x3F; // Start the PWM
                }
        else {
            OCR1BH=0x24;
            OCR1BL=0x22; // Period = 20ms, 1.5ms pulse
                }

        if (speedL==1) {
                OCR1CH=0x23;
            OCR1CL=0xF0; // Period = 20ms, 1.6ms pulse
            TCCR1A=0x3F; // Start the PWM
                }
        else if (speedL==2) {
                OCR1CH=0x23;
            OCR1CL=0xF0; // Period = 20ms, 1.6ms pulse
            TCCR1A=0x3F; // Start the PWM
            delay_ms(100);
                OCR1CH=0x23;
            OCR1CL=0x48; // Period = 20ms, 2.0ms pulse
            TCCR1A=0x3F; // Start the PWM
                }
        else if (speedL==-1) {
                OCR1CH=0x24;
            OCR1CL=0x54; // Period = 20ms, 1.4ms pulse
            TCCR1A=0x3F; // Start the PWM
            }
      else if (speedL==-2) {
                OCR1CH=0x24;
            OCR1CL=0x54; // Period = 20ms, 1.4ms pulse
            TCCR1A=0x3F; // Start the PWM
            delay_ms(100);
```

```
                OCR1CH=0x24;
            OCR1CL=0xFC; // Period = 20ms, 1.0ms pulse
            TCCR1A=0x3F; // Start the PWM
                }
        else {
            OCR1CH=0x24;
            OCR1CL=0x22; // Period = 20ms, 1.5ms pulse
                }
        }
if (speedL==0 && speedR==0) {
      TCCR1A=0x00; }
}
void servo(unsigned char position)
{
unsigned int destination = 0;
unsigned int current = 0;
unsigned char loop = 0;

motors(0,0);
delay_ms(1000); //make sure servos are stopped.
if(position==1){//open lid
      destination=0x24F8;
      }
else{
      destination=0x2408;
      }

TCCR3B=0x11;
TCCR3A=0x0F;
OCR3AH=0x27;
OCR3AL=0x10; /* TOP = 0x2710 = 10000 */
current = (unsigned int) (currentH*256)+currentL;
loop=0;
while (loop<60){
      if (current>destination){
            current=current-5;
            currentH=(unsigned int) (current/256)&0x00FF;
            currentL=(unsigned char) current&0xFF;
            OCR3CL = currentL;
            OCR3CH = currentH;
              delay_ms(65);
            }
      else if (current<destination){
            current=current+5;
            currentH=(unsigned int) (current/256)&0x00FF;
            currentL=(unsigned char) current;
            OCR3CL = currentL;
            OCR3CH = currentH;
              delay_ms(65);
            }
      else {
            current=destination;
            currentH=(unsigned int) (current/256)&0x00FF;
            currentL=(unsigned char) current;
            OCR3CL = currentL;
            OCR3CH = currentH;
            }
      loop++;
}
if (position!=1){
      TCCR3A=0x00; } //let lid drop if its supposed to be closed.
}
```