

University of Florida
Dept. of Electrical Engineering

Final Report

BOB the Builder
An Autonomous Bulldozer Robot

By
Salman Siddiqui
August 3, 2004

EEL5666C
Intelligent Machine Design Lab
Summer 2004
Dr. Arroyo
&
Dr. Schwartz

Table of Contents

<u>Abstract</u>	3
Executive Summary.....	4
<u>Introduction</u>	5
<u>Integrated System</u>	6
<u>Mobile Platform/Actuation</u>	7
<u>Sensors</u>	8
<u>Behaviors</u>	13
<u>Experimental Layout and Results</u>	14
<u>Conclusion</u>	15
<u>Parts</u>	16
<u>References</u>	17
<u>Source Code</u>	18

Abstract

Bob, the builder, is an autonomous bulldozer robot that performs a regular bulldozer's tasks without any input. Bob is planned in such a way so that it would be completely independent in its tasks and would not require any outside assistance for its operations. It searches for nearby objects and with the help of its moveable shovel, it picks the object up and carries it towards a dumping site. Once it has dumped its load it searches for more stuff to pick up. This way the cycle continues and Bob ends up cleaning the whole site by itself.

Executive Summary

Bob is an autonomous, self-trained, work-loving robot bulldozer. Its sole purpose is life is to play with dirt, move dirt, carry dirt, drop dirt and spread dirt. In short it likes dirt.

Its basic platform is made out of a toy bulldozer with a working shovel attached to it. To aide in its work it has a bunch of sensors. These sensors work together in their individual capacity to make Bob the way it is. For example:

Sonar is supposed to search for objects, which it successfully does. Once it has done so it proceeds towards the object.

Now the IR sensors up front kick in. They are suppose to align Bob with the object so that it can easily pick stuff up.

Once that has been accomplished the two, awesome, gear-head motors come into play. They go full speed ahead to provide the necessary force to lift the objects up.

Just at the right time the shovel goes up, taking the object with it.

Now Bob, with the help of four IR receivers starts homing in on the IR transmitter.

Once it gets close to the transmitter, Bob uses its photoreflector to detect for a black line.

As soon as a black line is detected, Bob knows to drop the stuff in its shovel and to return to the original site.

Introduction

Heavy equipment vehicles have always interested me. I am left in awe by watching such huge machines being operated by a human operator. These heavy-duty machines are responsible for getting the work done early, on time and appropriately. However, these machines are still what they are, stupid machines. They need to be operated by a human being in order to get the work done. I think that there is need for intelligent machines that can get the simpler work done by them selves. Simple tasks such as digging along a straight line, filling earth in a hole or moving sand/gravel from one place to another place are tasks that do not require much intelligence and they basically follow the same pattern over and over again.

My robotics project is based upon this observance. Bob is a bulldozer robot that is most useful at a construction site. It finds nearby objects, such as sand, gravel or iron, picks them up and moves them to a designated area. Bob is equipped with a wide variety of sensors to aid it in its tasks. These sensors enable it to “know” when the work has finished.

In this way a lot of manpower would be saved and that manpower can be employed elsewhere doing some intelligent work. Companies using these robots would be able to save a lot of money in terms of wages, thus gaining more profits.

The idea behind Bob’s construction is defined in the following pages. The various sections have been divided up according to the nature of the material and all aspects of Bob have been discussed at length.

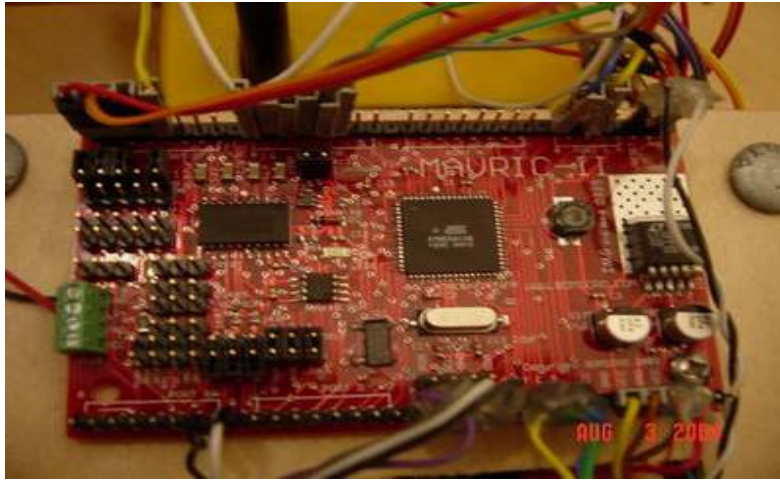
Integrated System

Bob's brain is the MavricII development board that's running a Atmega128 Microprocessor. The kit for the board was ordered from www.bdmicro.com and I soldered the components on the board. However this method proved to be a disaster as I ended up buying a new soldered board anyways. The micro controller is the main part of the robot that is actually responsible for whatever little intelligence Bob possesses. It is used to operate all the sensors, drivers and the LCD display. Every sensor provides feedback to the microprocessor and then based on that data, future commands are given to the other components. Software for the Microprocessor is written in the C language and then loaded into the onboard memory using the AVRISP programmer available from digikay.com. First the code for each sensor and each behavior was written and only after many hours of agonizing work this code was optimized to work with the final code. This technique of breaking code up into many small functions is found to be extremely useful by me and is recommended to everyone.

Of all the sensors the LCD is the most useful one. It provides instant feedback to the user in visual form and is an extremely important debugging tool. For this reason, the LCD is interfaced to the microprocessor first and foremost. It paves the way for the other devices as well.

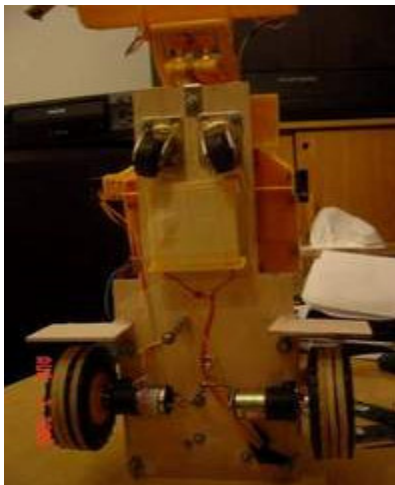
Power to the board and the sensors is provided by an eight-pack rechargeable battery pack that outputs 11V when fully charged. Rechargeable batteries are being used for this robot as to minimize the cost factor. There is a separate four-battery pack for the shovel. All the wires go through the on/off switch providing an instant kill to all on-board systems. A reset switch is directly interfaced to the MavricII board.

The following picture shows the connections of all my sensors to the board itself:



Mobile Platform/Actuation

My platform is really simple because it is, for the most part, a toy bulldozer. For the purpose of moving fast, I used two DC gear head motors and a motor driver to operate them. The DC gear head motors are required due to the large amount of torque needed to get Bob moving fast. I installed the two motors with the rear wheels and the castors were used up front. Here is an under-view of my robot:



I had to use the toy bulldozer because it would have been hard to create a properly working shovel. Therefore the whole body of the bulldozer was also used which gave a nice look to my robot:



As you can see from the picture my Sonar and two IRs are mounted on the shovel by Aluminum brackets. Aluminum was not used at first buy then it had to be used because its is much lighter than any other metal and provides more flexibility.

The four IR receivers on the top are my beacon receivers and are mounted in the most optimum configuration.

Sensors

A robot would be a dumb machine, like all the other machines, if it was not for its sensors. For this project I utilized a variety of sensors, ranging from SONAR to photo reflectors and IR transmitter and receivers.

Sharp GP2D12 IR:

These are the most popular of all the sensors, as every one seems to be using them. These provide pretty accurate distance measurement and are mainly used for object detection and obstacle avoidance. The sensor is comprised of an emitter for sending out the IR pulse and a receiver to record the reflected pulse. These provide the much needed

precise movement to the robot. They are also responsible for object avoidance and for centering the robot in line with the objects to be picked. Two of these have been mounted upfront on the shovel. For this reason they are only operational with the shovel down. This limits their usage but I don't really need them with the shovel up.

Devantech SRF04 Ultrasonic Range Finder (SONAR):

This is the most popular type of sonar device available. Its popularity is based on the fact that it is really simple and easy to use and provides excellent results. This is an excellent choice for any one that has to measure distance and proximity. The SRF04 model is a bit less sensitive than the SRF08 one but still it is all I required for my project and was amazed at the sensitivity of this thing.

This SONAR serves as my primary object detection sensor. The main scanning takes place with the help of this sensor. The breakout is given below:

- When Bob is turned on and left of the ground it starts rotating slowly on its axis.
- All the while it's rotating it also comparing the values given out by the SONAR sensor.
- Once it has found a suitable match to the value of the SONAR, Bob stops and proceeds in that direction.

In this way Bob is able to find the closest possible object to itself.

Hamamatsu P5587 Phototransistor:

This tiny phototransistor has been put on a t-tech board. The board was designed by Mr. Kosala, many thanks to him, and used by me. It is a design for only a single photorelector to be mounted on there but in my case there was no need for that.

This photoreflector is mounted under the robot just a quarter inch above the ground. This is a requirement for it to work properly. The sensor is also slightly slanted with respect to the ground and is not exactly parallel to it. This has been proven to be the best mount for this sensor and provides optimum results this way.

There is a black line surrounding the drop-off area. As soon as the robot passes over the black line the output of the photo-reflector changes and lets the microprocessor know that the dumping site has been reached.

Special Sensor Report

After Bob has picked up something in its shovel, it is required for it to proceed towards the dumping site. Therefore, a sensor is required to guide Bob towards this dumping site. To fulfill this purpose an IR beacon, comprised of two IR LEDs, was assembled in the lab on a PC board. Four IR receivers were ordered from www.jameco.com and used for detecting the beacon. This beacon and receiver combine together to form my special sensor.

Components

LMC 555CN CMOS Timer

This CMOS timer chip from National Semiconductor was utilized to create the 56.8kHz beacon to be placed at the dumping site. The timer was used in an astable because a continuous train of a 56.8 kHz frequency train is required. The power consumption for this chip is really low and is capable of being operated at 5V. However, for my purpose, 9V – 10V of voltage supply was used.

Data sheet can be found at: <http://www.national.com/ds/LM/LMC555.pdf>

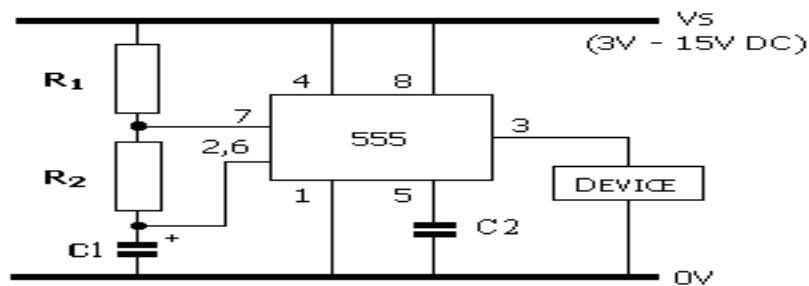
LITEONLTM9034 56.8kHz Receiver

This is a very common IR detector used in numerous appliances to receive IR signals such as TVs, VCRs and home stereos. Originally the output of this device is analog but by using Michael Hatterman's Hack, this receiver was modified to output analog voltage. See references for detail.

Construction

56.8 kHz Transmitter

Using the following circuit design, found at: www.eleinmec.com/article.asp?3, a 56.8 kHz frequency generator was constructed using a 555 timer chip:



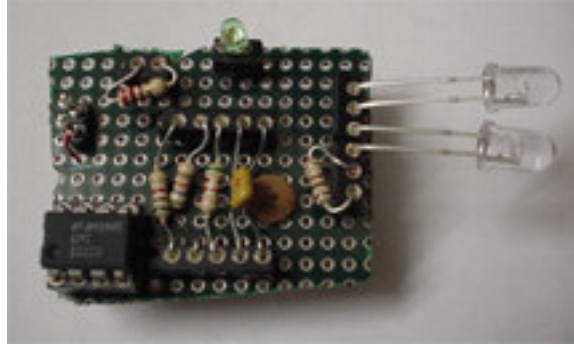
$$f = 1/T = 1.44/((R1 + 2R2)C1)$$

The above given equation for generating the desired frequency was obtained from the above given website. The values for the passive components were calculated and were found to be:

$$R1 = 5.6 \text{ K} \quad R2 = 10 \text{ K}$$

$$C1 = 1 \text{ nF} \quad C2 = 0.01 \text{ uF}$$

Real values of the resistors found to be the closest to the ones found were used in the design. The circuit was quite easy to construct and was laid out on a perforated board. An image of the constructed device is given below:



In our case the device, the load, is comprised of a 100-ohm resistor in series with two IR LEDs. As the circuit is an astable circuit, we obtain a continuous IR frequency of 56.8 kHz.

The LiteOn IR receiver was modified using a previous design. The details are provided in the reference section.

Failed Sensor Implementation: Digital Compass (Dinsmore 1490 Digital Compass)

I was thinking about implementing this digital compass on Bob as well. This would help bob in returning to the pickup site quickly. But unfortunately this didn't go as planned as I could not get the compass to work. Thus this project was abandoned after some experimentation.

Behavior

The following points summarize Bob's expected behavior. This is the idealistic version, meaning that I would like Bob to do all this, but then all the behaviors may not be feasible to implement in the robot at the end of the semester.

- When Bob is turned on and set on the ground, he immediately starts searching for close by objects by rotating on its axis.
- Upon detecting the closest object it proceeds towards that object
- Using the Sonar and IRs, it positions itself in the center of the object
- With a swift forward motion of the shovel it picks the object up in its shovel
- Upon picking up the object it then proceeds to the dumping site denoted by the IR beacon

Upon reaching the black line surrounding the beacon Bob stops and tries to find the original site by trial and error.

Experimental Layout and Results

Many experiments were performed during the course of the construction of my robot. Many involved the correct angle of the Soar and IR sensors of the ground, while others called for the calibration of the IR receivers on Bob's top.

First I had planned to use only two IR receivers at the top but then this didn't seem practical later on so I ended up ordering two more. The most important thing to keep in mind is to have a good idea about what your goals are and what you are going to need. This helps in lowering the costs and to finish the project in a timely manner without any wait for last minute shipments.

The project always seems to take a direction of it's own. There is no telling what idea might pop up in your mind. More often than not I ended up taking a whole new approach to a problem and solving it in a totally different manner.

Conclusion

This class has been one of the most exhausting classes I have ever taken. Even though my project was relatively simple and straight forward, even then it took me countless number of hours to make it work perfectly. This class is somewhat along the line of Microprocessor applications class in the sense that you gain a lot of knowledge by having first hand experience of everything. It is definitely an excellent class to take for the person who can handle the stress and depression of things not working right.

Special thanks to Mr. Max, Mr. William, Mr. Kosala and Mr. Steve for all their help and advice.

Also, a special thanks to Dr. Arroyo for teaching the class and Dr. Schwartz for making attendance mandatory and for all the threatening emails. Those were the real motivation behind all my effort.

Lessons Learnt:

- 1) Lynxmotion motors are a piece of CRAP, stay away from them.
- 2) On Atmega128 if you want to use all 8 A/D conversion ports, you have to write a 1 to the mMCUCSR bit twice consecutively.
- 3) Always make videos of your working robots! They might prove to be a life saver in the end.

Parts

- MavricII board (2; first one broke)-----\$160
www.bdmicro.com
- Sonar, IRs, Photorelector-----\$75
www.junun.org
- Motor Driver, Motors-----\$95
www.lynxmotion.com
- LCD, Switches-----\$20
www.junun.org
- AVR ISP Programmer-----\$35
www.digikey.com
- 4 Liteon receivers-----\$15
www.jameco.com
- Various stuff from Lowes-----\$40
(nuts, bolts, screws, castors....etc.)
www.lowes.com
- Toy Bulldozer-----\$28
www.toysrus.com
- Rechargeable batteries and charger---\$35
www.samsclub.com
- Relays, perforated boards-----\$20
www.radioshack.com
- Switches, wire,nuts,bolts,solder-----Free from Lab

****All prices include shipping

References

LiteOn 56.8 kHz receiver hack By Michael Hatterman. IMDL. Spring 2002

http://www.mil.ufl.edu/imdl/papers/IMDL_Report_Spring_02/michael_hatterman/hacked_ir.pdf

LiteOn 56.8 kHz receiver Data Sheet

<http://jameco.com/Jameco/Products/ProdDS/176541.PDF>

Referred to Jeff Panos final report. IMDL. Spring 2004

http://www.mil.ufl.edu/imdl/papers/IMDL_Report_Spring_04/panos_jeff/gimp-ss.pdf

Referred to Palani Rathinasamy final report. IMDL. Spring 2003

http://www.mil.ufl.edu/imdl/papers/IMDL_Report_Spring_03/rathinasamy_palany/golfcaddy.pdf

Referred to the following websites

<http://www.eleinmec.com>

<http://www.national.com>

www.bdmicro.com

Source Code

```

//Salman Siddiqui
//IMDL summer 2004
//Source Code for Bob the Builder
//This code is only 90% done so far; so dont count on the other 10%
//Get the LCD code from http://jump.to/fleury super code...guranteed to work

#include <stdlib.h>
#include <avr/io.h>
#include "lcd.h"
#include "adc.h"
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include <avr/signal.h>
#include <stdio.h>

///delay function
void delay (uint16_t n)
{
double b;
for (b=0;b<n;b++)
{}
}

/*
code for 6.5 u sec delay
*/
void delay_10us (int t)
{
int n =800;
while (t--);
{
n = 800;
while (n!=0)
{
n--;
}
}
}

//Define all the movements over here
//the following are the timer 1 values for the
//motion.I am using 2 out of 3 pwms
void slowforward(int speed)
{
for (int a=0;a<speed;a++)

```

```

{
for (int b = 0;b<(speed/20);b++)
{
  outp(0xe0, OCR1AL);
  outp(0x47, OCR1AH);
  outp(0xe0, OCR1BL);
  outp(0x47, OCR1BH);
  outp(0xe0,OCR1CL);
  outp(0X47,OCR1CH);
}}

```

```

void mediumforward(int speed)
{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
  outp(0x50, OCR1AL);
  outp(0x47, OCR1AH);
  outp(0x50, OCR1BL);
  outp(0x47, OCR1BH);
  outp(0x50,OCR1CL);
  outp(0X47,OCR1CH);
}}}

```

```

void maxforward(int speed)
{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
  outp(0x00, OCR1AL);
  outp(0x46, OCR1AH);
  outp(0x00, OCR1BL);
  outp(0x46, OCR1BH);
  outp(0x00,OCR1CL);
  outp(0X46,OCR1CH);
}}}

```

```

void fastleftturn(int speed)
{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
  outp(0x50, OCR1AL);
  outp(0x46, OCR1AH);
  outp(0x50, OCR1BL);

```

```

outp(0x40, OCR1BH);
outp(0x50,OCR1CL);
outp(0X40,OCR1CH);
}}}
```

```

void slowleftturn(int speed)
{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
outp(0x50, OCR1AL);
outp(0x47, OCR1AH);
outp(0x50, OCR1BL);
outp(0x40, OCR1BH);
outp(0x50,OCR1CL);
outp(0X40,OCR1CH);
}}}
```

```

void fastrightturn(int speed)
{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
outp(0xe0, OCR1AL);
outp(0x40, OCR1AH);
outp(0x50, OCR1BL);
outp(0x46, OCR1BH);
outp(0x50,OCR1CL);
outp(0X46,OCR1CH);
}}}
```

```

void slowrightturn(int speed)
{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
outp(0x00, OCR1AL);
outp(0x40, OCR1AH);
outp(0x50, OCR1BL);
outp(0x47, OCR1BH);
outp(0x50,OCR1CL);
outp(0X47,OCR1CH);
}}}
```

```

void reverse(int speed)
```

```

{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
outp(0x90, OCR1AL);
outp(0x49, OCR1AH);
outp(0x90, OCR1BL);
outp(0x49, OCR1BH);
outp(0x90,OCR1CL);
outp(0X49,OCR1CH);
}}
}

void stop(int speed)
{
for (int a=0;a<speed;a++)
{
for (int b = 0;b<(speed/20);b++)
{
outp(0x50, OCR1AL);
outp(0x42, OCR1AH);
outp(0x50, OCR1BL);
outp(0x42, OCR1BH);
outp(0x50,OCR1CL);
outp(0X42,OCR1CH);
}}
}

//Functions for the AD conversion
//taken from www.bdmicro.com
uint16_t adc_read(void)
{
return ADC;
}

void adc_wait(void)
{
/* wait for last conversion to complete */
while ((ADCSR & BV(ADIF)) == 0);
}
/*
* adc_start() - A/D start conversion
*
* Start an A/D conversion on the selected channel
*/
void adc_start(void)
{
/* clear conversion, start another conversion */

```

```

    ADCSR |= BV(ADIF);
}

void adc_chsel(uint8_t channel)
{
    /* select channel */
    ADMUX = BV(REFS0);
    //ADMUX = BV(REFS1);
    ADMUX = (ADMUX & 0xf0) | (channel & 0x07);
}
//void selectchannel(int a)
//{
//ADMUX = a
void initadc(void)
{
    outp(0xe7,ADCSRA);
}

uint16_t adc_readn(uint8_t channel, uint8_t n)
{
    uint16_t t;
    uint8_t i;
    adc_chsel(channel);
    adc_start();
    adc_wait();
    adc_start();
    /* sample selected channel n times, take the average */
    t = 0;
    for (i=0; i<n; i++) {
        adc_wait();
        t += adc_read();
        adc_start();
    }

    /* return the average of n samples */
    return t / n;
}

/****THE FOLLOWING CODE WORKS PERFECTLY FOR SONAR
//I didnt use interrupts but used this code
//also found in palanis report from spring 2003
int sonar (int times)
{
    delay_10us(10);
    int i,counter=0;
    for (i=0;i<times;i++)
    {
        cbi(PORTE,0);

```

```
sbi(PORTE,0);
delay_10us(4);
cbi (PORTE,0);
```

```
while (bit_is_clear(PINE,2))
{
}
while (bit_is_set(PINE,2))
{
delay_10us(6);
counter++;
}
}
return (counter/times);
}
```

```
//Shovel operation code
//This is specifically to turn the relay
//connected to my port E on to make the shovel go up or down
void shovelup (void)
{
int a,b;
//delay(6000);
delay(6000);
delay(6000);
for(a=0;a<600;a++)
{
for(b=0;b<1000;b++)
{
outp(0x01,PORTG);
}}
outp(0x00,PORTG);
delay(6000);
delay(6000);
delay(6000);
}
```

```
//shovel down code
void shoveldown (void)
{
int a,b;
//delay(6000);
delay(6000);
delay(6000);
for(a=0;a<600;a++)
{
for(b=0;b<1000;b++)
{
```

```

outp(0x04,PORTG);
}}
outp(0x00,PORTG);
}

```

```

//look to find item
//searching for item
void look (void)
{
int a,b,c,d;
b=0;
c=0;
for(;;)
{
slowrightturn(40);
a=sonar(10);
printf("a=%d ",a);
if (a<400)
{
stop(20);
return;
}
delay(6000);
lcd_clrscr();
b=a;
}
}

```

```

//Searching for item
//actually cheking the IR for alignment
void search (void)
{
lcd_clrscr();
uint16_t snr,z,y;
for (;)
{
snr=sonar(5);
z=adc_readn(0,5);
y=adc_readn(1,5);
mediumforward(30);
printf("  S= %u          L=%u  R=%u",snr,z,y);
if(z>100)
{
stop(500);
reverse(600);
fastrightturn(450);
}
if(y>100)

```



```

{
stop(500);
reverse(600);
fastleftturn(850);
}
if ((z>100) && (y>100))
{
stop(50);
reverse(800);
stop(500);
maxforward(800);
shovelup();
mediumforward(500);
slowforward(500);
stop(50);
return ;
}
if (snr<40)
{
stop(50);
reverse(800);
stop(500);
maxforward(890);
shovelup();
mediumforward(600);
slowforward(600);
stop(50);
return ;
}
delay (6000);
lcd_clrscr();
}
}

//Go to home base code
//this code is to make bob compare different receiver values
//and to make it to the home base transmitter
void gohome (void)
{
int valueofpreflector;
uint16_t left,right,front,back;

int a,b,c,d,e,f,g,h;
a=0;
b=0;
c=0;
d = 0;
stop(2000);

```

```

for (;;)
{
left=adc_readn(2,5);
right=adc_readn(3,5);
front=adc_readn(4,5);
back=adc_readn(5,5);
lcd_clrscr();
printf("a=%d b=%d c=%d d=%d",a,b,c,d);
e=0;
f=0;
g=0;
h=0;
if (front>left && front >right && front >back)
{
a++;
e++;
mediumforward(100);
}
//reverse
if (back>left && back > right && back > front)
{
b++;
f++;
fastrightturn(1600);
}
//leftturn
if (left > right && left > front && left > back)
{
c++;
g++;
fastrightturn(100);
}
//rightturn
if (right>left && right>front && right > back)
{
d++;
h++;
fastleftturn(100);
}
if (bit_is_clear(PINE,4))
{

stop(1000);

if (e==1)
{
fastrightturn(1600);
}
}

```

```

if(g==1)
{
fastrightturn(1150);
}

if(h==1)
{
fastleftturn(1150);
}

stop(30);
return;
}

delay(6000);
lcd_clrscr();

}
}

//Port Initialization of all ports
//good thing to have...just write all the port information
//over here to initialize them properly
void initports(void)
{
//for ADC setup
DDRF = 0x00;
PORTF = 0x00;
outp(0x80,MCUCSR);//write twice clear the jtd flag to disable jtag interface
outp(0x80,MCUCSR);//clear the jtd flag to disable jtag interface

//for sonar
outp(0x01,DDRE);

//for Motor control
outp(0xff, DDRB);

//for shovel
outp(0xFF,DDRG);
}

///LCD initialization
//code taken from peter fluery website
//excellent code
//thanks to mr.kosala on helping me with
//my LCD
void initlcd (void)

```

```

{
lcd_init(LCD_DISP_ON);
lcd_clrscr();
fdevopen(lcd_putc, NULL, 0);
}

//initialize timer1 for motor control
//Timer initialization for PWMs
void inittimer1(void)
{
outp(0xa8, TCCR1A); /* init the counter */
outp(0x12, TCCR1B); /* init the counter */
outp(0X4e,ICR1H);
outp(0X20,ICR1L);
outp(0x00, TCNT1L); /* value of T/C1L */
outp(0x00, TCNT1H); /* value of T/C1L */
}
//The main source code
//all you have to do is to call functions
int main (void)
{
uint16_t z,y,left,right,front,back,snr;
int volts;
initlcd();
initadc();
initports();
inittimer1();
for (;;)
{
look();
stop(2000);
mediumforward(900);
search();
delay(10000);//the delays have been added to give it time to stop
delay(10000);
gohome();
}
}

```