Wassim Tawil
TAs: William Dubel
Steven Pickles
Instructors: A. A Arroyo
E. M. Schwartz

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

FINAL REPORT

JAMESBOT

The Voice Controlled Spying Robot

Table of contents

Abstract

JamesBot is an autonomous voice controlled spying robot. His behaviors include avoiding obstacles, wall following, detecting dark areas, and voice recognition. This paper will explain the idea behind this robot, and the steps taken to build it. It will give details about the design, and each component used. Finally, it will explain how it was all put together to bring the final product.

Executive Summary

There were many steps involved in building JamesBot. First, I had to design the platform; I came up with a compact and original design. Then, I had to choose a microcontroller, so the TAs recommended the Mavric because of its versatility. Once the platform was cut out and put together, I attached the motors and the motor driver, and wrote a software the get it moving. I needed to avoid obstacles, so I placed three bump sensors and three IRs. I tested the IRs at different distances to measure its accuracy, and then wrote the software for the obstacle avoidance. For the second part of the project, I connected the photoresistors, and the voice circuit. I placed three photoresistors wrapped in duct tape at the bottom of the robot to detect the change in light. For the voice circuit, I tried many different commands until I found the ones with the most accuracy. I used a wireless microphone to communicate with the circuit. Once it was all working, I wrote a wall following software that used the left and front IRs. I then put all the software together to get the final code.

Introduction

       Is there someone close to you that's now trustworthy? Were you ever tempted to spy on them? Well, now you can with JamesBot. JamesBot is a voice controlled robot whose main function is to enter an unknown area, find a place to hide, and then transmit video and audio feedback. This would make a great toy for kids to play with around the house with or it can even be made into a security robot for outside use.

       I've always been interested in spy stories and gadgets, and this is how the idea came about. I want a robot that can be controlled from distances and that can transmit information back wirelessly. It also has to be inconspicuous. This report will give a detailed explanation on the different features of JamesBot and the components used to make him work. I will start with a general description of the system, then the platform and actuation, and finally the sensors and behaviors.

Integrated System

The most important part of any robot is the microcontroller; it is the brain of the robot, controlling all its behavior. This is why it's important to choose one that will do the required task. I chose the Mavric-IIB board from bdmicro.com which supports the Atmega128 microprocessor. This is my first time working on robots and I wasn't sure which board I would need, so the TAs recommended the Mavric-IIB. It has all the necessary components to build an autonomous robot and is programmable using C.

JamesBot will need different kinds of sensors to accomplish his tasks successfully. These sensors will gather data and send it to the microcontroller, which it will use to output the next necessary behavior. For collision and obstacle avoidance, bump switches and IR sensors are needed. IRs will also be employed for wall following. Cds cells will be used to find the darkest area in a room. JamesBot will be voice controlled, so he will use a speech recognition circuit which will be interfaced to my board. Finally, I will have a wireless webcam for video and audio feedback which will not be connected to the microcontroller. It will use its built-in transmitter to send the video to the receiver, which in this case is my laptop. An LCD will also be used on the robot for feedback.
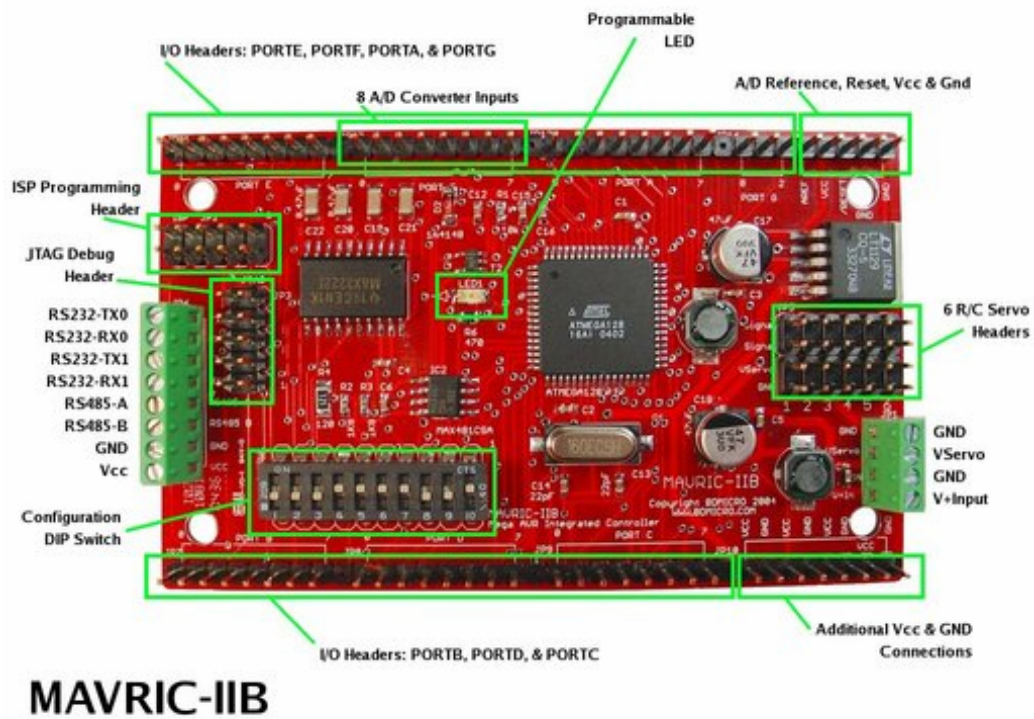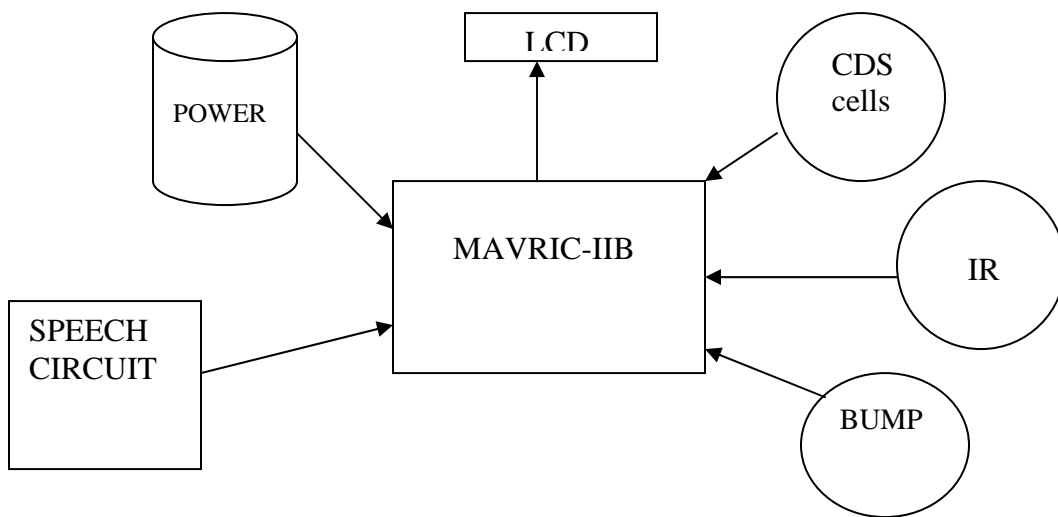
*Fig.1: Mavric-IIB board*

Fig.2: Interfacing sensors to board

Mobile Platform

JamesBot needs to enter a room and not get noticed. He also needs to be able to hide between furniture. For that I will try to make his platform as small as possible and try to fit all his components in an efficient way. I will use balsa wood which is supplied by the class to build the platform. I'm thinking of making a main body where all the electronics and the motors will be placed and then have an upper layer covering the body where the camera will be installed; it will be a very simple design. In the back I will place the wireless receiver and the speaker. It's still early to predict how the robot will look like exactly because I don't how big all the components are. As soon as I order them though, I will have a better idea.

Actuation

The only parts needed for the actuation of JamesBot are motors, a motor driver, and wheels. I will use two independent DC gearhead motors with a 4mm shaft, 290 RPM and 43:1 gear ratio. I think that should be enough to push my robot and all of his electronics. To control the speed and direction of the motors I will need a motor driver. Since I need to control each wheel separately, a dual H-Bridge motor driver will be required. The one I'm using has dual channels, a range of 4.8v-12vdc, and a peak current of 2A. This driver is capable of rotating clockwise, counterclockwise, and breaking a wheel. It will connect directly to the motors and will get the necessary signals from the microprocessor. The wheels I'm using are 2.25"D x 0.5"W Neoprene Tires and are

connected to the motors using a 4mm mounting hub. All these accessories used for actuation were purchased from lynxmotion.com.
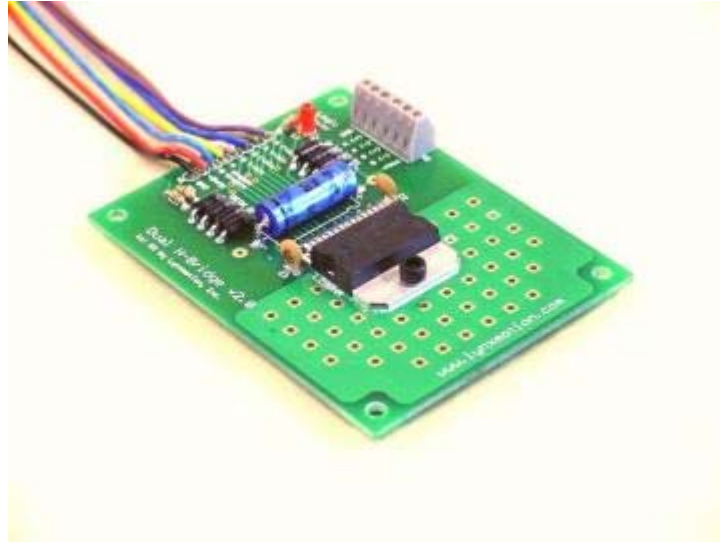
Fig.3: Dual H-Bridge motor driver

Sensors

<u>Bump switches</u>

This is the least expensive and most basic sensor I'll be using. I actually got them for free for IMDL lab, and they will only be used as backups for the IR sensors. They'll be used to inform the microcontroller whenever the robot hits an obstacle. The bump switch is a digital sensor because it sends either a low or high voltage depending on the situation. It's basically a pull-up resistor, and whenever the bump is hit, it causes the switch to close and make the input low. At all other times, the input stays high. On my robot, I will use two switches in the front, about 90 degrees apart, and two in the back. I

will then place a circular piece of wood all around connecting the four switches. I included a figure below to give you an idea.
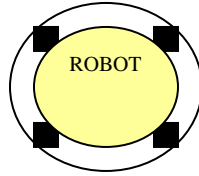
Fig.4: Bump switches

IR

      I will use the Sharp GP2D12 sensors as they seem to be the most popular and provide good distance measurement. On JamesBot, they will be used for obstacle avoidance and wall following. The sensor has an emitter which sends out an IR pulse and a receiver which records the reflected pulse. Depending on the received pulse, the sensor can approximate the distance of the obstacle. The distance is reported as an analog voltage with a range of 4'' to 30''. I'm thinking of using three of these sensors; one in the front and two on the sides. An image of the sensor is provided below.

Fig.5: Sharp GP2D12

CDS cells

        My robot will need to go into a room and hide in the darkest area. To be able to accomplish this, he will be equipped with photoresistors. Photoresistors are a type of light sensor, and can be described as variable resistors. Whenever there is a change in the light level, the resistance is changed. The weaker the light reflected, the greater the resistance, and vice versa. When it's connected to the microcontroller, the resistance will have to be converted to a voltage. This can be obtained by making a voltage divider circuit. An example can be seen below (acroname.com). If R1 is the photoresistor, then the voltage will increase with increasing light intensity.
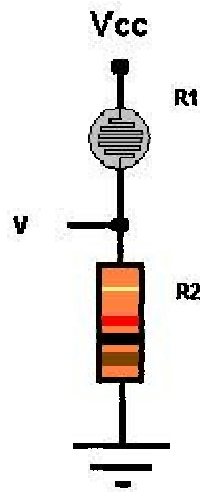
Fig. 6: Voltage divider for Photoresistor

<u>Voice Recognition</u>

The most useful feature of JamesBot is its voice recognition system. It will be used to guide the robot to a specified location, and also in case his position needs to be changed while he's spying. I will be using the SR-07 Speech Recognition Kit from imageco.com. This kit has numerous features; it can handle twenty different words and can be programmed for either isolated or continuous speaking. It has non-volatile back up memory and can be easily interfaced to external circuits. When a trained word is recognized, the circuit outputs a digital number corresponding to this specific word. Since my robot will be controlled in places where I can't see it, I need to still be able to transmit my voice commands. This will be arranged by using a pair of 2-way radios, and placing one of them on the robot. Below I provided the SR-07's circuit.
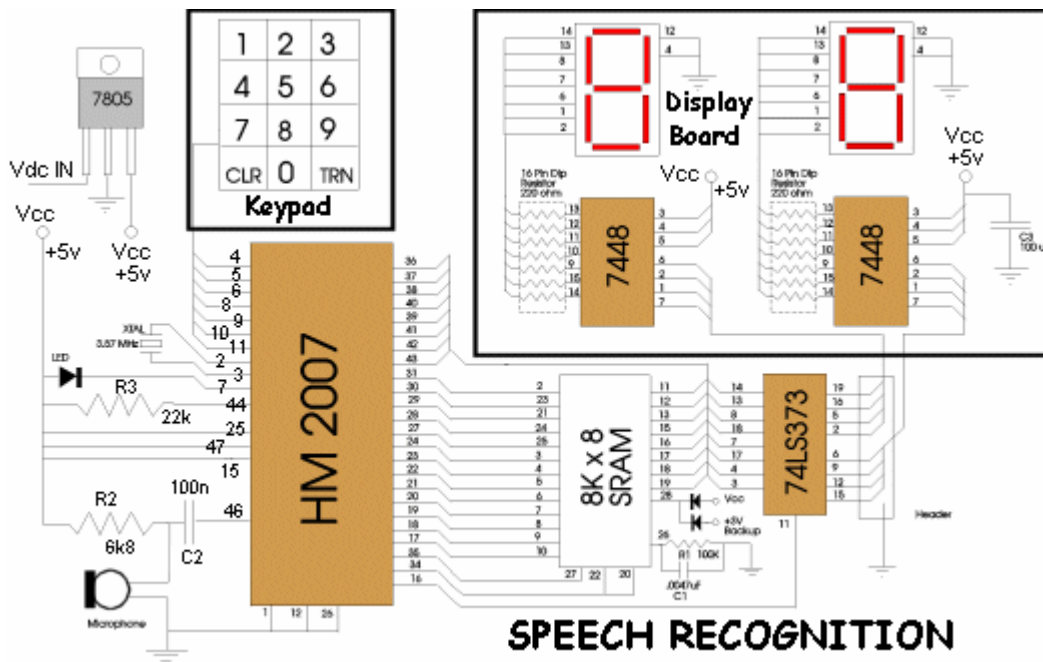
Fig. 7: Speech Recognition circuit

<u>Wireless camera</u>

Another addition to my robot will be a wireless camera. This is an important part since my robot's main task is to spy. I will use the X10 Nightwatch camera which is equipped with special sensors to see in dim areas. Unlike my other sensors, this one will not be connected to the microcontroller. Instead it will send video and audio feedback using its built in transmitter directly to a receiver plugged into my laptop.

Fig. 8: X10 wireless camera

Behaviors

JamesBot's behaviors include:

- Obstacle avoidance

- Speech recognition

- Wall following

- Detecting dark area

- Transmitting video and audio feedback

Ideally, this is what I'd like my robot to do. First I will guide him to whichever room or area I want him to spy using voice commands and a 2-way radio. Then, he will enter the room and begin wall following until he finds a place that's dark enough to hide in. Once he finds his spot, he will stop and transmit video/audio feedback back to my laptop.

Experimental Results

Bump Switches

      I connected all my bumps to a common pin. I used a voltage divider circuit so that each switch will output a different value. Since this value was between 0 and 5V, I had to convert it to a digital value using the on board A/D converter. The Mavric is equipped with eight A/D pins located on Port F. I used Pin 0 for the bump switches. This is a 10 bit converter, so the outputted value was between 0 and 1024. Below I provided a table showing the values obtained when I tested my switches.

BUMP	DIGITAL VALUE
Right	678
Front	484
Left	302
Right + Front	750
Left + Front	580

Figure 9

IR

       I used three sensors on my robot, one in the front and one on each side. I tested the IRs to see how well they work at different distances. I performed this test in a well lit room, and I got values starting at one inch from the wall up to thirty five inches. I found out that the range at which these sensors work best is between four and twenty eight inches. Below that range, the values seem to increase very fast, and above that, they stop changing. JamesBot will not need to see that far; he will only avoid obstacles and do wall following which require up to six inches of detection, and in which case, these IRs work fine. For my obstacle avoidance, I programmed him to avoid anytime he gets a value greater than 300, and it worked perfectly. Below, I provided a table with all the value that I obtained in my tests, as well as a graph.

Distance (inches)	Digital Value
1	192
2	382
3	551
4	480
5	405
6	334
7	301
8	250

9	233
10	209
11	199
12	182
13	172
14	160
15	141
16	138
17	131
18	123
19	114
20	112
21	107
22	103
23	96
24	95
25	92
26	94
27	95
28	101
29	120
30	127
31	130

32	130
33	130
34	130
35	130

Figure 10

Digital Value
y

x
Disantce (in)
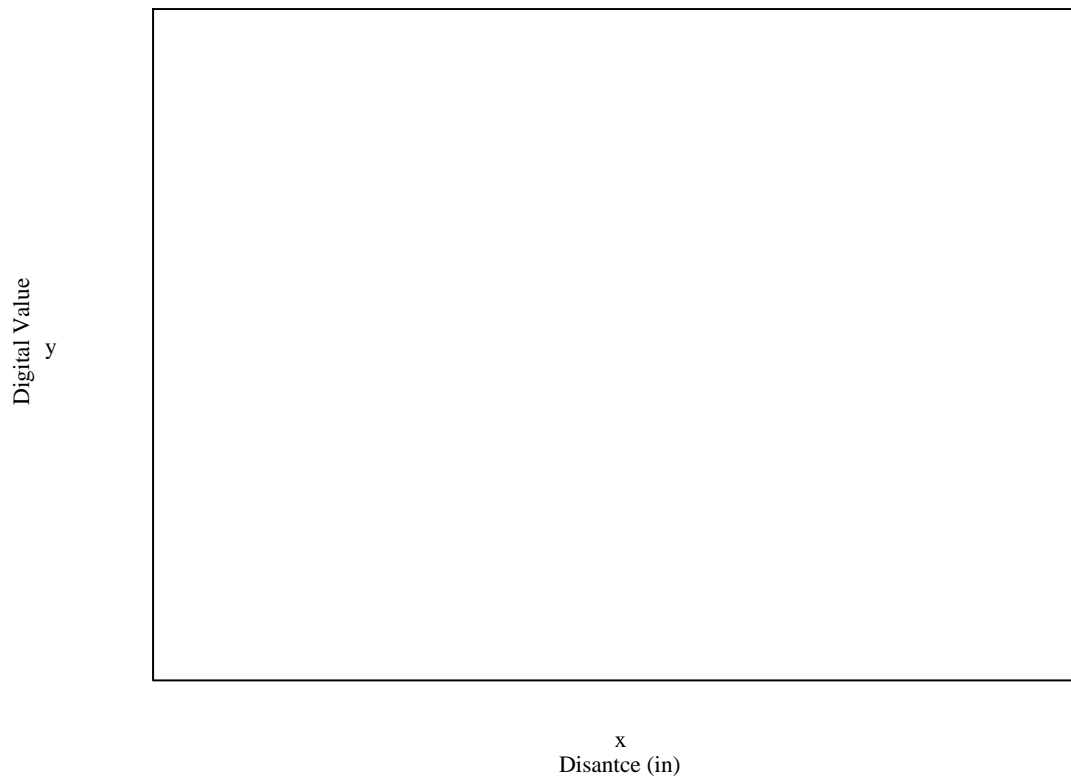
Figure 11

CDS Cells

I covered the cells with duct tape to isolate the light received. JamesBot is

equipped with three photoresistors. One is placed in the front, and two behind the tires. I

wanted to make sure his whole platform is in the dark before he stops; that's why I placed

these sensors this way. Whenever all three sensors detect the dark area, he will stop wall

following, and begin spying. If only one or two of the sensors detect the dark area, then

he keeps moving, until all three agree. I tested these sensors in my room, and found all

the typical places that JamesBot could hide under. I then placed him in these areas and

read the value. I got values under a chair, a desk, and under the bed. I connected the

photoresistor to a voltage divider circuit with a 15Kohms resistor, and connected the

changing voltage to a pin on the A/D converter. Here are the value that I obtained.

JamesBot position	Digital Value
Not hiding	640
Under chair	550
Under desk	450
Under bed	140

Figure 12

Voice Recognition

This was the most difficult sensor I worked on because it had many factors. While

testing this sensor, I had to account for such things as the tone of voice, the word length,

outside noise, and the placement of the microphone with respect to my mouth. Since I

was using a two way radio to send the commands, my voice came out less clear, which

meant I had to be even more careful with my choice of words. The circuit can recognize

forty 0.96s words or twenty 1.92s words. I configured it to twenty so that I can get more

choices for words, and because I will only be using about six or seven words. I first

tested out the kit in my room with the door shut and without the two way radio. After

trying many different words, I found out that the circuit works much better with words

that have more than one syllable. For example, words like "left" or "right" were

sometimes confused. I then used my two way radio instead, and noticed that the

recognition accuracy slightly decreased. It was still good enough though. I also found out

that the distance between the microphone and the radio is very important. When I moved

the radio away, the circuit would not recognize the commands as well. Another crucial

factor is the tone of the voice. The accuracy got much better when each word was trained

with a different tone of voice. The tricky part though is that every word had to be spoken

with the same level of excitement as when it was trained. For example, if I trained the

word "forward" in a happy voice and then repeated that same word with a sad voice, the

circuit would not recognize it. So, to get the highest accuracy possible, the words need to

have more than one syllable, they need to have a different tone, and they have to be

repeated with that same tone, and the radio has to be kept the same distance from the

microphone. I had some of my friends try to speak into the circuit to test if it recognized

their voice, and it came out that it depended on the person. My brother, for example, who

sounds like me, was able to get the circuit to recognize his voice most of the time. As for

the other persons, the circuit would sometimes recognize their voice, depending on the

word. The final test was to try the circuit while there was outside noise, like the

television. The kit was placed inside the robot, so it was shut from outside interferences.

When I spoke into radio though, and the noise was loud enough, the circuit would

sometimes capture that noise mixed with my voice and get very confused. That was the worst case scenario, and even then, it still had more than fifty percent accuracy. I ended up using a wireless microphone that I ordered form Ebay, because it works much better than the two way radio.

Conclusion

At the end, I was satisfied with the performance of my robot; it was very close to what I was expecting. He responds well to different commands, and when he enters a room, he never fails to find the dark area. I wrote a simple wall following program that works very well, and is the best feature on my robot. The only downfall is that it wasn't very smooth, but that could have been fixed in software if I had more time. The voice circuit was the most irritating sensor to work on. I would not recommend it to anybody looking for a hundred percent accuracy. It responded to my commands about ninety percent of the time; that was the best that I could get from it. The IRs used in this project worked very well, and in different lighting condition. They also perform perfectly at short distances. I had no problems with the photoresistors detecting change in light, as long as you isolate the light received. That can be accomplished by wrapping duct tape around the photoresisor. I was not satisfied with the bump sensors though; whenever the robot hit a wall, they occasionally got activated. I got them from the IMDL lab for free, so I'm not complaining. If I were to start over with the project, I would make some slight changes. First, I would redesign the platform and try to make the components fit nicer, and be a

little more organized. I would also use better bump sensors. This has definitely been the

hardest class I've taken at UF; it's very time consuming and very stressful. The

advantages though, are that you gain valuable experience, and it looks great on your

resume. I would, without a doubt, recommend this class.

References

- www.acroname.com

- www.bdmicro.com

- www.imagesco.com

- www.lynxmotion.com

- www.x10.com

Appendices

```c
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/pgmspace.h>
#include <stdio.h>
#include "LCD.h"
#include "motor.h"
#include "ADC.h"

#define OCR_1MS 125


volatile uint16_t ms_count;


/*
 * ms_sleep() - delay for specified number of milliseconds
 */
void ms_sleep(uint16_t ms)
{
 TCNT0  = 0;
 ms_count = 0;
 while (ms_count != ms)
   ;
}


/*
 * millisecond counter interrupt vector
 */
SIGNAL(SIG_OUTPUT_COMPARE0)
{
 ms_count++;
}

void init_timer0(void)
{
 TCCR0 = 0;
 TIFR  |= _BV(OCIE0)|_BV(TOIE0);
 TIMSK |= _BV(TOIE0)|_BV(OCIE0);        /* enable output compare interrupt */
 TCCR0  = _BV(WGM01)|_BV(CS02)|_BV(CS00); /* CTC, prescale = 128; 16Mhz /
128 = 8us */
```

```
  TCNT0  = 0;
  OCR0   = OCR_1MS;                    /* 8us * 125 = 1 ms */
}


uint8_t input;
uint16_t irm;
uint16_t irl;
uint16_t irr;
uint16_t bump;
uint16_t prm;
uint16_t prr;
uint16_t calr;
uint16_t calm;
uint16_t setr;
uint16_t setm;


int main(void)
{
        int value;
        int avoid;


        //INITIALIZATIONS
        LCD_init();
        init_timer0();
        motors_init();
        fdevopen(LCD_sendByte,NULL,0);
        adc_init();
        PORTC = 0x00;
        DDRC = 0x00;
        sei();

        bump = adc_readn(0, 5);

        while (bump < 350)                // Photoresistors calibration
                {
                        bump = adc_readn(0, 5);
                        calr = adc_readn(4, 5);
                        calm = adc_readn(5, 5);

                        setr = calr;
                        setm = calm;

                        LCD_clearScreen();
```

```c
                LCD_delayLong();

                printf("%d", setm);
                printf(" ");
                printf("%d",setr);
                ms_sleep(1000);
        }

while (1)                               // Main code
{
        input = PINC;
        input &= 0x0F;

        if (input == 0x01)
        {
                slight_left();

                LCD_clearScreen();
                LCD_delayLong();
                printf("T");
                LCD_delayLong();
                printf("u");
                LCD_delayLong();
                printf("r");
                LCD_delayLong();
                printf("n");
                LCD_delayLong();

                while (input == 0x01) {

                        input = PINC;
                        input &= 0x0F;
                        }
        }


        else if (input == 0x02)
        {

                LCD_clearScreen();
                LCD_delayLong();
                printf("F");
                LCD_delayLong();
                printf("o");
                LCD_delayLong();
                printf("r");
```

```
                    LCD_delayLong();
                    printf("w");
                    LCD_delayLong();
                    printf("a");
                    LCD_delayLong();
                    printf("r");
                    LCD_delayLong();
                    printf("d");
                    LCD_delayLong();

                    while (input == 0x02) {

                            input = PINC;
                            input &= 0x0F;

                            irm = adc_readn(1, 5);
                            irl = adc_readn(2, 5);
                            irr = adc_readn(3, 5);
                            bump = adc_readn(0, 5);

                            //if (bump >= 250)
                             // {
                                    //avoid = 7;
                             // }

                            if (irm > 300 && irl < 300 && irr < 300)
// front sensor
                             {
                                    avoid = 1;
                             }

                            else if (irm > 300 && irl > 300 && irr < 300)
// front and left sensor
                             {
                                    avoid = 2;
                             }

                            else if (irm > 300 && irl < 300 && irr > 300)
// front and right sensor
                             {
                                    avoid = 3;
                             }

                            else if (irm < 300 && irl > 300 && irr < 300)
// left sensor
                             {
```

```
                              avoid = 4;
                            }

                            else if (irm < 300 && irl < 300 && irr > 300)
// right sensor
                            {
                                    avoid = 5;
                            }

                            else if (irm > 300 && irl > 300 && irr > 300)
// all sensors
                            {
                                    avoid = 6;
                            }

                            else
                            {
                                    avoid = 0;
                            }

                            if (avoid == 1)
                            {
                                    motors_brake();
                                    ms_sleep(100);
                                    drive_backward();
                                    ms_sleep(400);
                                    motors_brake();
                                    ms_sleep(100);
                                    turn_right();
                                    ms_sleep(irm);
                                    motors_brake();
                                    ms_sleep(100);
                                    drive_forward();
                            }

                            else if (avoid == 2)
                            {
                                    motors_brake();
                                    ms_sleep(100);
                                    turn_right();
                                    ms_sleep(350);
                                    motors_brake();
                                    ms_sleep(100);
                                    drive_forward();
                            }
```

```
else if (avoid == 3)
  {
        motors_brake();
        ms_sleep(100);
        turn_left();
        ms_sleep(350);
        motors_brake();
        ms_sleep(100);
        drive_forward();
  }

else if (avoid == 4)
  {
        motors_brake();
        ms_sleep(100);
        turn_right();
        ms_sleep(350);
        motors_brake();
        ms_sleep(100);
        drive_forward();
  }

else if (avoid == 5)
  {
        motors_brake();
        ms_sleep(100);
        turn_left();
        ms_sleep(350);
        motors_brake();
        ms_sleep(100);
        drive_forward();
  }

else if (avoid == 6)
  {
        motors_brake();
        ms_sleep(100);
        drive_backward();
        ms_sleep(400);
        motors_brake();
        ms_sleep(100);
        turn_right();
        ms_sleep(700);
        motors_brake();
        ms_sleep(100);
        drive_forward();
```

```
                }

                else if (avoid == 7)
                 {
                        motors_brake();
                        ms_sleep(100);
                        drive_backward();
                        ms_sleep(350);
                        motors_brake();
                        ms_sleep(100);
                        turn_right();
                        ms_sleep(500);
                        motors_brake();
                        ms_sleep(100);
                 }

                else
                 {
                        drive_forward();
                 }
        }

}

else if (input == 0x03)
{
        drive_backward();

        LCD_clearScreen();
        LCD_delayLong();
        printf("R");
        LCD_delayLong();
        printf("e");
        LCD_delayLong();
        printf("v");
        LCD_delayLong();
        printf("e");
        LCD_delayLong();
        printf("r");
        LCD_delayLong();
        printf("s");
        LCD_delayLong();
        printf("e");
        LCD_delayLong();

        while (input == 0x03) {
```

```
                        input = PINC;
                        input &= 0x0F;


                        }
}

else if (input == 0x08)
{
        motors_brake();

        LCD_clearScreen();
        LCD_delayLong();
        printf("S");
        LCD_delayLong();
        printf("t");
        LCD_delayLong();
        printf("o");
        LCD_delayLong();
        printf("p");
        LCD_delayLong();

        while (input == 0x08) {

                input = PINC;
                input &= 0x0F;
                }

}

else if (input == 0x04)
{
        LCD_clearScreen();
        LCD_delayLong();
        printf("S");
        LCD_delayLong();
        printf("e");
        LCD_delayLong();
        printf("a");
        LCD_delayLong();
        printf("r");
        LCD_delayLong();
        printf("c");
        LCD_delayLong();
        printf("h");
        LCD_delayLong();
```

```c
                printf("i");
                LCD_delayLong();
                printf("n");
                LCD_delayLong();
                printf("g");
                LCD_delayLong();
                printf(".");
                LCD_delayLong();
                printf(".");
                LCD_delayLong();
                printf(".");
                LCD_delayLong();

                while (input == 0x04) {

                        input = PINC;
                        input &= 0x0F;
                        bump = adc_readn(0, 5);
                        irm = adc_readn(1, 5);
                        irl = adc_readn(2, 5);
                        irr = adc_readn(3, 5);
                        prr = adc_readn(4, 5);
                        prm = adc_readn(5, 5);

                        if (prm <= setm && prr <= setr)
                        {
                                value = 7;
                        }


                        //else if (bump >= 200 && bump < 600)
                        //        {
                        //                value = 6;
                        //        }

                        else if (irl > 150 && irl < 450 && irm < 300)
// Left sensor sees wall
                          {
                                value = 1;
                          }

                        else if (irl < 150)              // Left sensor too far
                              {
                                value = 2;
                              }
```

30

```
                    else if (irl > 450 && irm < 300)              // Left   sensor
too close
                            {
                                    value = 3;
                            }

                    else if (irl > 450 && irm > 300)              //   Left    and
middle sensor too close
                            {
                                    value = 4;
                            }

                    else if (irl < 450 && irm > 300)              //        Middle
sensor too close
                            {
                                    value = 5;
                            }


                    if (value == 1)
                            {
                                    drive_forward();
                            }

                    else if (value == 2)
                            {
                                    motors_brake();
                                    ms_sleep(100);
                                    turn_left();
                                    ms_sleep(200);
                                    motors_brake();
                                    ms_sleep(100);
                                    drive_forward();
                                    ms_sleep(450);
                            }

                    else if (value == 3)
                            {
                                    motors_brake();
                                    ms_sleep(100);
                                    turn_right();
                                    ms_sleep(100);
                                    motors_brake();
                                    ms_sleep(100);
                                    drive_forward();
                                    ms_sleep(400);
```

```
                }

        else if (value == 4 || value == 5)
                {
                        motors_brake();
                        ms_sleep(100);
                        turn_right();
                        ms_sleep(350);
                        motors_brake();
                        ms_sleep(100);

                }

        else if (value == 6)
                {
                        motors_brake();
                        ms_sleep(100);
                        drive_backward();
                        ms_sleep(350);
                        motors_brake();
                        ms_sleep(100);
                        turn_right();
                        ms_sleep(150);
                        motors_brake();
                        ms_sleep(100);
                }

        else if (value == 7)
                {
                        motors_brake();
                        //ms_sleep(100);
                        //turn_right();
                        //ms_sleep(500);
                        //motors_brake();

                }

        else
                {
                        drive_forward();
                }

                }
}
else
{
```

```
                    motors_brake();

                    LCD_clearScreen();
                    LCD_delayLong();
                    printf("E");
                    LCD_delayLong();
                    printf("r");
                    LCD_delayLong();
                    printf("r");
                    LCD_delayLong();
                    printf("o");
                    LCD_delayLong();
                    printf("r");
                    LCD_delayLong();
            }

        }

        return 0;
}
```