Lawn Gator
Autonomous Lawn Care Robot
April 26, 2006
Michael Gregg
EEL 5666C Intelligent Machine Design Lab

Table of Contents

**Abstract**

This robot is designed to test the practicality of constructing an autonomous lawn care system. It will also allow experimentation with several technologies and help to investigate their practical applications. The first prototype of this platform will use an RF emitting wire as a containment field, random walk programming to control obstacle and field limit boundary reactions, and a linear motor controller implemented in software. The lawnmower will be an autonomous platform with the integral safety feature of obstacle avoidance. The robot is a robust outdoor platform designed to operate continuously for several hours. Future versions will be weatherproofed and will include an automatic recharging system. The sensor package includes an ultrasound sensor for obstacle avoidance, one RF field strength detector, and one digital compass to permit plow navigation programming.

**Executive Summary**

The robot completed this semester was built to integrate several technologies into a robust platform designed to meet a practical need. The robot was designed around a robust frame. The frame was made from aluminum plate welded together. The robot is powered by high torque DC brushless motors. The cutting motor is a motor and cutting head taken from an electric weed trimmer. The robot's front wheels are mounted in a way that permits a zero turning radius. The robots maneuvers by reversing the direction of one of the motors.

The robot's processor is the Atmel 128 chip implemented in the BD Micro Mavric IIB board. The robot's sensor suite consists of an ultrasound, digital compass, and an RF field detector. The robot uses a software implemented linear control scheme to change motor speeds. The programming for the robot is very simplistic and reliable.

The RF field detectors are modified electronic dog collars used as training devices for pets. The detectors were supplied by Radio Systems Corporation. The sensors were modified by removing a large voltage transformer from the circuit. This reduced the output voltage of the dog collar to a reasonable 5 volts. There is a separate transmitter which connects to a wire laid at the boundary of the yard.

## Introduction

In the last several years, autonomous platforms have crept into consumer's lives to handle menial, time intensive tasks. Several of these platforms have been designed for lawn care. However, the cost of these robot far outweighs the cost of having the lawn care handled by an external company. The goal of this project is to develop a platform that is completely autonomous, has the safety features that consumers expect (and the liability issues in the USA demands), and has a final cost under $500.

In the past, several autonomous lawnmowers have been created at the University of Florida. The earliest attempt was named the *Lawn Nibbler* [1] and was built around an electric weed trimming platform. This robot used an electronic pet fence in order to stay within the required boundaries. It also proposed and demonstrated a simple navigation system (gPS, ground positioning system) using triangulation of ultrasonic pulses emitting from several "satellites" placed in the yard. In 1999, Chandler and Meiszer [2] created the *Lawn Shark* using a modified Toro electric push mower. This mower used two ultrasonic sensors (sonar) and an improved gPS system. Meiszer [3] explored the use of genetic algorithms to optimal place the satellites for the gPS system. Chandler later explored the use of textural analysis for intelligent mowing in [4] and [5]. The robot described here will begin the process of improving on these previous designs by creating a robust and reliable platform that will adequately

and intelligently cover a consumer's yard, while also maintain safe and continuous operation. This platform will also be used for further research.

The discussion of this robot will begin with a description of the integrated system. In this section will introduce how the systems operate and interact with each other. Block diagrams of the sensor package are also displayed. The next section will describe the mechanical design of the platform. The design specifications are and relevant calculations are shown in this section. CAD drawings and pictures of the robust design of the lawn care system are displayed. The drive mechanisms and specifications are also covered in some detail. The next section includes an in-depth look at the sensors used in the present version of the robot, and discusses the design, theory, motivation, application, and the performance results. The last section describes the software that integrates the sensors and actuators with the required behaviors to enable the robot to meet all the performance goals.

**Integrated System**

Figure 1 shows a block diagram of the systems and subsystems in this lawn care robot. The system is centered on the BD Micro Mavric IIB microcontroller board. All the sensor outputs are inputs to this device and all the command functions are generated by this microcontroller board. The Mavric IIB is powered by the Atmega 128 14.75 MHz microcontroller/microprocessor and has 128K RAM in a very compact package. The microprocessor board is very powerful and more

than capable of handling this robot's present purpose and many of the future upgrades presently envisioned. The Atmega AVR format also has many enthusiasts and there is a great deal of information available online posted by experienced users. The BD Micro has built in capabilities for controlling six pulse width modulation (PWM) outputs to for motor or servo control. The two drive motors on the robot are controlled by a PWM signal sent from the microprocessor board to motor drivers contained within the motor assemblies. Two sensors are connected to the I2C connections on the board. Several I/O ports on this board allow other sensors and an LCD display to be readily integrated into the design

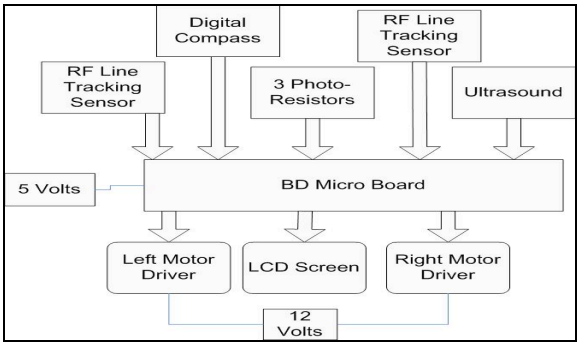


**Figure 1.** Preliminary Block Diagram of subsystems

The values from the boundary sensors (an electric pet fence) are analyzed using a fuzzy logic algorithm. When, for example, a boundary is found, the PWM signals to one or both of the drive motors are modified to turn the platform away from the boundary. The motor has onboard circuitry that controls speed and direction of the drive motor. The robot uses and LCD display driven by the microprocessor and mounted on the upper surface of the platform to show motor speeds and directions, the distance to an obstacle determined by the ultrasound sensor, and bearing determined by the compass.

## Mobile Platform

The platform (shown in Figure 2) was designed to support 30 pounds over 20 inches without significant deflection [6]. The platform is made of aluminum, and

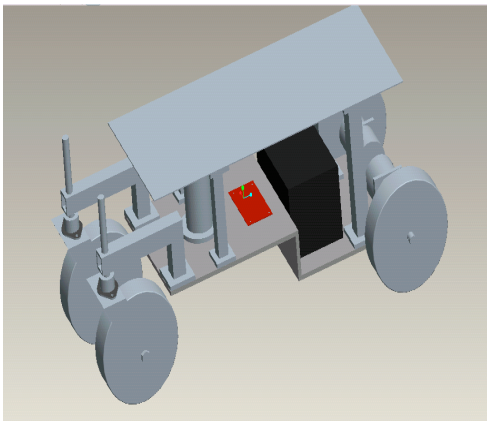

**Figure 2.** Pro Engineer Model of Robot

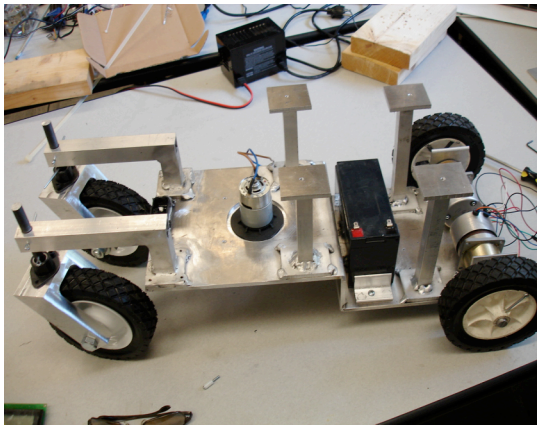

**Figure 3**. Lower Level of Robot

has two levels. The upper level is made of 1/16" sheet aluminum and supports the sensors and the microcontroller board. The lower level is made of a 1/4" aluminum plate which supports the drive motors, the cutting motors, and the batteries. The upper level is connected to the lower level by 4 supports visible in Figure 3. The supports are constructed from 3/4" square aluminum bars, with a wall thickness of 1/8". The upper level supports, motor mounts, and front wheel supports are welded into place as shown in Figure 4. Figure 5 shows the fully assembled platform.

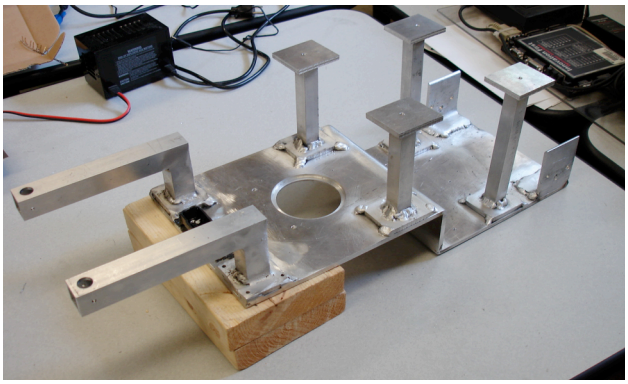

Figure 4. The Frame of the Robot

The platform is supported on four 6-inch diameter wheels. The platform currently has only one cutting height, however improvements in the design are being made to permit a range of cutting heights.

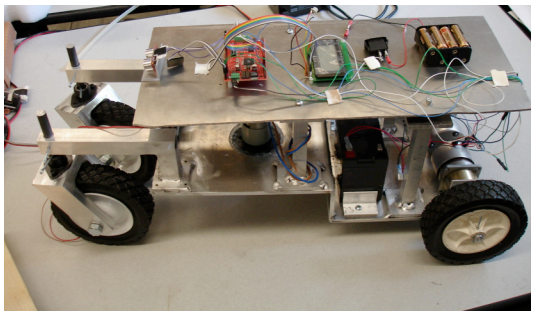

**Figure 5**. Completed Robot

**Actuation**

The robot is propelled by two motors, each providing 300 oz-in of torque. The force required to commence motion [6] of the lawnmower is determined by the equation

$$F = u \times N \ , \qquad \text{EQN. 1}$$

where F is the required force, u is the coefficient of friction between rubber and grass (~0.35) [7], and N is the normal force at each wheel. The normal force was set at 10 lbs to give a factor of reliability, resulting in a required force of 3.5 lbs. The required torque is determined by the equation

$$T = r \times F \ , \qquad \text{EQN. 2}$$

where T is the torque required, r is the radius of the wheels (3 inches), and F is the force (from Eqn 1). The torque required at each motor is 10.5 in-lbs. The motors selected were the Pittman N2300 Series Brushless Motors, with a torque rating of 18.75 in-lbs (see figure 6). The motors are connected to the wheels by a shaft. The shaft is firmly attached to the motor by setscrews and the wheel is held in place by cotter pins. The robot maneuvers by changing the speed and

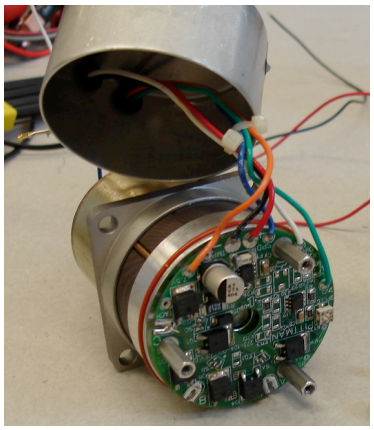

Figure 6. Motor with Internal Control Board Visible

direction of each of the two drive motors. These parameters are controlled by two PWM signals from the Mavric board sent to control boards integrated in the motor assemblies.

The control board used is the Pittman 2311 Onboard Feedback and Control Board. The controller uses 5 wires as inputs and 2 wires as outputs. The controller inputs power, direction, and speed. The speed is varied by a PWM signal. The range is 2 KHz to 10 KHz, and the signal is a 5 volt pulse with a duty cycle between 0 and 100% pulse width. The wide range of PWM frequencies made the process of integrating the drive motor with the Atmel 128's internal clock very easy. A pre-scalar is used to adjust the internal clock to a frequency suitable for PWM generation. Equation 3 [8] shows the formula required to calculate the pre-scalar value used with the PWM control, where $f_{clk\ I/O}$ is the clock speed of the microprocessor, $f_{OCnxPWM}$ is the PWM signal generated, N is the prescaler, and TOP is the maximum number in the timer's counter. Since this is an 8 bit timer, the value for TOP is 255. An 8 bit timer was used as opposed to the available 16 bit timer because the resolution in an 8 bit timer is more than adequate for the design purpose. The current design allows 256 possible motor speeds. A 16 bit timer would allow 65,536 possible motor speeds.

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)} \qquad \text{EQN. 3}$$

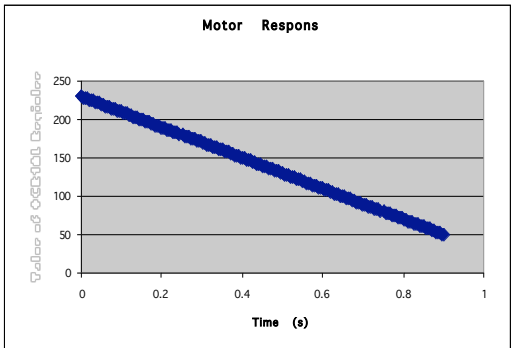


**Figure 6**. Plot of Linear Motor Response

Using this equation and pre-scalar of 8, a 7.2 KHz signal is generated. The PWM signal is made by setting the OCR1AL and OCR1BL registers to a certain value. OCR1AL controls the right motor, while OCR1BL controls the left motor. The timer continuously counts from 0 to 255 and then resets itself to 0. When the timer matches the value set in OCR1AL or OCR1BL, the output pin on the Mavric board which controls the motors' PWM signal is set to +5 volts. When the timer returns to zero, the output pin is set to 0 volts. This continuous pattern creates a square wave which comprises the PWM signal. The lower the value in OCR1AL or OCR1BL is, the sooner the square wave starts, and the longer the square wave lasts. This in turn increases the output of the motor. At extreme values near 255 the motor does not recognize the short spike and the opposite effect is produced. The control board also has a difficult time with the signal if it is too close to zero. The motors are controlled in the software by a simple while loop with several if statements. Short delays are built into the function to smooth out the motor reaction. The result is a smooth linear reaction to a motor speed change (Figure 6). A sample of the motor driver code is shown in Figure 7

```
void motor(int dirl, int speedl….){
   while(speed_val_l != speedl ||
speed_val_r != speedr….
          ……
          if(speed_val_l > speedl){
                    OCR1BL = OCR1BL +1;
                    speed_val_l--;}
                       …….
          delay_ms(5);
```

**Figure 7.** Sample Code for Motor Drivers

The front wheels are mounted in a fashion that permits a zero turning radius (see Figure 8. The cutting motor is a 12 Volt DC motor with a weed trimmer head taken from a cordless weed trimmer. Currently it is turned on and off through the microprocessor board in response to sensor readings. The drive and cutting motors will be powered by a 12 Volt, 7 Amp-hour lead acid battery mounted on the lower platform. The drive motors draw .25 Amps each, giving the motors a projected run time of several hours.

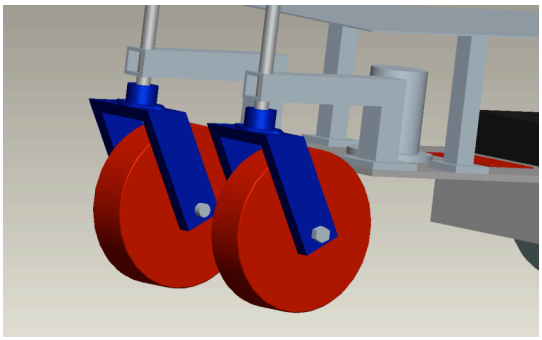

**Figure 8.** Front Wheel Mounting

**Sensors**

The lawnmower's sensors are designed to ensure safe and efficient operation of the robot. The robot has a Devantech SRF-08 Ultrasound Sensor for obstacle avoidance. The sensor is mounted in the front facing forward with no obstructions. The performance data for this sensor is shown in Figure 9 The range is accurate from 30 inches to 1 inch. The sonar will only detect large objects beyond 36 inches. For redundancy, the robot has several bump sensors mounted on the front and sides to detect collisions. The robot also has a digital compass which can be used for a rudimentary plow navigation pattern. The compass chosen was the

Devantech CMPS-03 Digital Compass. Both of the Devantech sensors are connected with an I2C connection to the microprocessor board. There are also photo-resistors mounted in the cutting section to detect light. This is a safety feature to prevent unsafe lifting while the cutting motor is on.

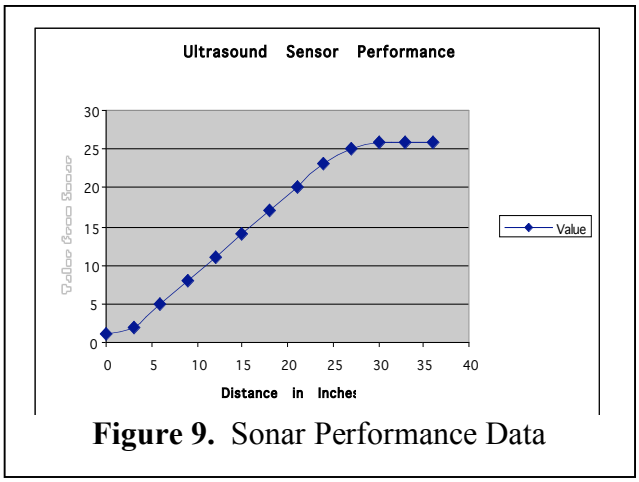


**Figure 9.** Sonar Performance Data

The boundary is controlled by an electronic pet fence, supplied by Radio Systems Corporation. The system includes a transmitter with a boundary wire, and separate collar unit.

The transmitter sends a 10 KHz sine wave over the boundary wire. When the collar is within range, it emits an electric shock. The unit was opened up, and the transformer that generates the large voltage shock was removed from the circuit. The output signal is taken across the gap where the transformer was removed. The output is a constant 5 volt signal when it is activated, and 0 volts when off. One sensor is currently mounted and implemented. Another sensor is in the development process. When an additional sensor is mounted, the robot will not

only know that it reached the boundary, but also which side hit the boundary and the approximate direction that the robot reached the boundary.

**Behaviors**

The robot uses random travel and reactions for its behavior. The robot travels in its initial start heading until an obstacle or the boundary is reached. When either of these circumstances occurs, the robot reverses both motors, turns to a random direction greater than 90 degrees and less than 270 degrees. The robot then continues this until another obstacle is hit or the boundary is reached and continues in this pattern until statistically, the yard has been covered. In the future, the robot will use the RF boundary sensors to “wall follow” with the boundary, then use the compass to complete a plow pattern. The behavior is implemented by running if statements in the main program loop, which compare the value of both the ultrasound and the input pins of the RF field detector sensor. A more efficient method using interrupts is being implemented.

**Cost**

The total replacement cost for this robot is just under $700 (as detailed in Table 1). In the above bill of materials, the items with a star were donated by companies or were previously owned by the Machine Intelligence Lab. The actual cost to build this robot was $397. It is my belief that the actual robot can still achieve its goal of a production cost of less than $500. I believe that if this robot were to be mass produced, an agreement with Radio Systems Corporation could be made, or circuitry could be designed to achieve a similar goal.

| Budget | Qnty | Price | Total |
|---|---|---|---|
| BD Micro Board | 1 | $100.00 | $100.00 |
| Compass | 1 | $ 50.00 | $ 50.00 |
| Ultrasound | 1 | $ 50.00 | $ 50.00 |
| Photoresistors | 4 | $ 2.50 | $ 10.00 |
| Petsafe Radio Fence* | 1 | $254.95 | $254.95 |
| Cutting Motor | 1 | $ 30.00 | $ 30.00 |
| Battery* | 1 | $ 29.00 | $ 29.00 |
| Batteries | 1 | $ 10.00 | $ 10.00 |
| Setscrews | 2 | $ 1.00 | $ 2.00 |
| Wheels | 4 | $ 6.00 | $ 24.00 |
| Bearings | 2 | $ 9.09 | $ 18.18 |
| Cotter Pins | 3 | $ 1.50 | $ 4.50 |
| Screws | 1 | $ 5.00 | $ 5.00 |
| Al Plate | 1 | $ 44.04 | $ 44.04 |
| Al Sheet* | 1 | $ 12.94 | $ 12.94 |
| Al Rod | 1 | $ 4.88 | $ 4.88 |
| Al Wheel fork | 2 | $ 7.00 | $ 14.00 |
| Al Square Bar | 2 | $ 13.21 | $ 26.42 |
| Al Motor Mounts | 1 | $ 4.00 | $ 4.00 |
| | | | |
| Total | | | $693.91 |

**Table 1.** Cost of Completed Robot
* denotes donated items

## 4. Conclusion

This paper was written to demonstrate the feasibility of constructing an autonomous lawn care system for small applications and further research. Presently, this system only has the capabilities of a safe an autonomous lawnmower, similar to other robot lawnmowers now available for purchase. Past research has shown that a combination of plow cutting, wall-following and random travel based on the parameters of a given yard will be necessary to cover the yard efficiently (without full knowledge of the yard layout).

Future designs may include a navigation system that permits even more efficient coverage.

## 5. Future Work

Work will be continued on this robot or a second prototype during the summer of 2006. During the next phase of the project, a recharging system will be added to the lawn care system. Other mechanical additions adjustments for cutting height, an additional motor and cutter for edging and a camera mounted on the front. The camera will be used to reproduce some of Chandler's work to detect the difference between cut and uncut grass. This will allow us to more efficiently determine a cutting pattern in open areas. The camera will also be used to determine the boundary between the sidewalk (or road) and the grass. We also intend to use the camera to detect the presence of weeds, grass in need of fertilizer and ant hills. Once these turf problems are found, the lawn care robot will dispense the appropriate chemical treatment.

## References


[1] Kevin Hakala, A. Antonio Arroyo, K. L. Doty, Scott Jantz, Erik de la Iglesia, "LawnNibber: An Autonomous Lawnmower and Navigation System", Volume 4, May 26-27, 1998, Florida Institute of Technology, Melbourne, Florida.

[2] Rand C. Chandler, Katherine A. Meiszer, A. Antonio Arroyo, "LawnShark: A New Platform for Autonomous Mowing and Navigation", Volume 5, April 29-30, 1999, University of Florida, Gainesville, Florida.

[3] Katherine A. Meiszer, "Using a Genetic Algorithm to Determine Optimal Beacon Placement for a Beacon Navigation System," UF Master's Thesis, 2001.

[4] Rand C. Chandler, A. Antonio Arroyo, Michael Nechyba, Eric M. Schwartz, "The Next Generation Autonomous Lawn Mower", Volume 4, May 4-5, 2000, Florida Atlantic University, Baca Raton, Florida.

[5] Rand C. Chandler. *Autonomous Agent Navigation Based On Textural Analysis*, UF Ph.D. Dissertation, Gainesville, Florida, 2003.

[6] Beer, Ferdinand P, Johnston Jr., E. Russell, DeWolf, John T. *Mechanics of Materials.* McGraw Hill. New York, NY 2002.

[7] Glenn Elert. *The Physics Hypertextbook: Friction.*
http://hypertextbook.com/physics/mechanics/friction/. Accessed March 24, 2006.

[8] Atmel Corporation. *Atmel 128 Manual.* Atmel Corporation, San Jose, CA, 2006.

**Code**

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.24.7d Evaluation
Automatic Program Generator
© Copyright 1998-2005 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
e-mail:office@hpinfotech.com

Project :
Version :
Date    : 4/20/2006
Author  : Freeware, for evaluation and non-commercial use only
Company :
Comments:


Chip type           : ATmega128
Program type        : Application
Clock frequency     : 14.745600 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 1024
*****************************************************/

#include <mega128.h>
#include <delay.h>


#include <stdlib.h>
// #include <stdio.h>

// I2C Bus functions
#asm
   .equ __i2c_port=0x12 ;PORTD
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm
#include <i2c.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x1B ;PORTA
#endasm
```

```
#include <lcd.h>

#define          _BV(bit)   (1 << (bit))
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))

#define ADC_VREF_TYPE 0x00

// Declare your global variables here


void motor(int, int, int, int);
// void obstacle(void);
// void turn(int);
void pingE0(void);
void compassC0(void);
void shock(void);
// void plow(void);
void obstacle(void);
void read_zapper(void);
// void bearing_avg(void);
// void strtwy(void);

int speed_val_l = 230;
int speed_val_r = 230;

int dir_val_l = 1;
int dir_val_r = 1;

unsigned int zapper_value=0;
unsigned int heading=0;
unsigned int desired_dir=0;

// int desired_dir;
// int heading;

// int i;
// int j;
// int t;
int rndm;

unsigned char rangeE0;
unsigned char bearingC0;

char str[10];
char stra[10];
```

```
char strb[10];
char speedle[10];
char speedri[10];


void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
//
//  LCD Display
//
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=Out Func5=Out Func4=In Func3=Out Func2=Out Func1=In
Func0=In
// State7=T State6=0 State5=0 State4=T State3=0 State2=0 State1=T State0=T
//
// Pins 5 and 6 are PWM output
// Pins 2 and 3 and Direction Outputs to motors
// Pin 1 is input of shock sensor
//
PORTB=0x00;
DDRB=0x6C;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=Out
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=0
PORTC=0x00;
DDRC=0x01;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
//
// Pin 1 is SDA
// Pin 0 is SCL
//
PORTD=0x00;
```

```
DDRD=0x00;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func4=In Func3=In Func2=In Func1=In Func0=In
// State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
ASSR=0x00;
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 1843.200 kHz   <-- Frequency for 7.2 Khz Output
// Mode: Fast PWM top=00FFh
// OC1A output: Non-Inv.
// OC1B output: Non-Inv.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
//
// PWM for Primary Drive Motors
```

```
//
TCCR1A=0xA1;
TCCR1B=0x0A;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
```

```
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;


// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
ETIMSK=0x00;

// ADC initialization
// ADC Clock frequency: 115.200 kHz
// ADC Voltage Reference: AREF pin
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=0x40;
ADCSRA=0xc7;


// I2C Bus initialization
i2c_init();


// LCD module initialization
lcd_init(16);
lcd_clear();

delay_ms(500);

while (1)
      {
      // Place your code here

      lcd_clear();
```

```
    pingE0();

    delay_ms(50);

    pingE0();

    delay_ms(100);

    compassC0();

    delay_ms(50);

    read_zapper();
    if(zapper_value > 200)
       shock();
    }
}


// Read the zapper from the A to D
// This function takes approximately .3 ms to complete
void read_zapper(void)
{
unsigned int result;

//uses 2.5 V reference
ADMUX=0x40|0x00;
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;

result = ADCW;

//average the values (new value affects the result by 1/16 of the difference)
result = (result + zapper_value) >> 1;
result = (result + zapper_value) >> 1;
result = (result + zapper_value) >> 1;
zapper_value = (result + zapper_value) >> 1;
}


void motor(int dirl, int speedl, int dirr, int speedr){
```

```
    while(speed_val_l != speedl || speed_val_r != speedr || dir_val_l != dirl || dir_val_r != dirr){

        if(dirl < 0){
            //set output pin of left motor to 0 Volts
            PORTB.2 = 1;
            dir_val_l = -1;
        }

        if(dirl > 0)
            {
            //set output pin of left motor to 5 Volts
            PORTB.2 = 0;
            dir_val_l = 1;
        }

        if(dirr < 0){
            //set output pin  B 3 of right motor to 0 Volts
            PORTB.3 = 0;
            dir_val_r = -1;
        }

        if(dirr > 0)
            {
            //set output pin of left motor to 5 Volts
            PORTB.3 = 1;
            dir_val_r = 1;
        }

        // delay_ms(25);

        if(speed_val_l > speedl){
            OCR1BL = OCR1BL +1;
            speed_val_l--;
        }

        if(speed_val_l < speedl){
            OCR1BL = OCR1BL -1;
            speed_val_l++;
        }

        if(speed_val_r > speedr){
            OCR1AL = OCR1AL +1;
            speed_val_r--;
        }
```

```
        if(speed_val_r < speedr)
        {
            OCR1AL = OCR1AL -1;
            speed_val_r++;
        }

        lcd_clear();
        lcd_gotoxy(0,0);
        lcd_putsf("motr l:r");
        lcd_gotoxy(9,0);
        itoa(speed_val_l, speedle);
        itoa(speed_val_r, speedri);
        lcd_puts(speedle);
        lcd_putsf(":");


        lcd_gotoxy(13,0);
        lcd_puts(speedri);

        delay_ms(5); //delay on each increment to protect motors
    }

} //end motor()


void compassC0()
{
// PORTC.0=1;
delay_ms(50);
i2c_start();
i2c_write(0xC0);
i2c_write(0x01);

i2c_start();
i2c_write(0xC0 | 1);
bearingC0=i2c_read(0x01);
i2c_stop();

lcd_gotoxy(0,2);
lcd_putsf("Bearing: ");
lcd_gotoxy(10,2);
itoa(bearingC0, str);
lcd_puts(str);
// PORTC.0 = 0;
delay_ms(50);
}
```

```
void pingE0()
{
i2c_start();
i2c_write(0xE0);
i2c_write(0x00);
i2c_write(0x50);
delay_ms(70);

i2c_start();
i2c_write(0xE0);
i2c_write(0x03);

i2c_start();
i2c_write(0xE0 | 1);
rangeE0=i2c_read(0);
delay_ms(70);
i2c_stop();

lcd_gotoxy(0,1);
lcd_putsf("Range:   ");
lcd_gotoxy(10,1);
itoa(rangeE0, stra);
lcd_puts(stra);
}

void shock(void){
        lcd_clear();
        lcd_gotoxy(0,2);
        lcd_putsf("Shock!");

        motor(1, 240, 1, 240);
        delay_ms(50);

        motor(-1, 150, -1, 150);
        delay_ms(1000);

        rndm = TCNT1L + 1100;
        itoa(rndm, strb);
        lcd_gotoxy(0,3);
        lcd_puts(strb);

        motor(1, 175, -1, 175);
        delay_ms(rndm);

        motor(1, 125, 1, 125);
```

```
        }


// Obstacle Avoidance Function
void obstacle(void){
    if(rangeE0 <= 0x10){
        motor(1, 210, 1, 210);
        delay_ms(250);
        motor(-1, 150, -1, 150);
        delay_ms(250);
        rndm = TCNT1L + 1100;
        itoa(rndm, strb);
        lcd_gotoxy(0,3);
        lcd_puts(strb);

        motor(1, 175, -1, 175);
        delay_ms(rndm);

        motor(1, 125, 1, 125);
        }
     }
```