University of Florida
EEL5666L
Department of Electrical and Computer Engineering
Intelligent Machine Design Laboratory

Ball Man

Final Report

Vivek Manoharan
April 17th, 2004

TAs: William Dubel, Steven Pickles
Instructors: Dr. A. Arroyo, Dr. E. Schwartz

TABLE OF CONTENTS

ABSTRACT

This paper will describe and explain the development of Ball Man. Ball Man is an intelligent and autonomous ball-boy robot. It is designed to retrieve stray tennis balls upon a tennis court and once it has collected its target amount, it locates its base station and deposits the tennis balls there. Feedback is provided through its LCD screen to notify the user of how many balls it has collected and what behavior it is currently exhibiting. Furthermore, this paper will explain the various sensors that were used to implement Ball Man's behaviors, and also explain the process of building this robot over the course of the semester.

EXECUTIVE SUMMARY

Ball Man is a robotic ball-boy that will roam around a tennis court and collect stray tennis balls. Once Ball Man has collected its target amount of tennis balls (currently set to 3, the amount in a typical can), it will search for its base station (an orange cone), and once it locates its base station, it deposits the tennis balls.

To start, the color of the tennis ball must me initialized unto Ball Man. This is to reduce risk of error due to different lighting conditions, and in case the players are using differently colored tennis balls (extra heavy duty pink tennis balls, etc.). Ball Man roams around a tennis court and avoids objects that don't match the properties of a tennis ball. Therefore, it can avoid hitting the net, the outside fence, and people walking around the court. Once the CMU camera has properly found the RGB color values of the initialized tennis ball, Ball Man will use the CMU camera's x-y coordinates to center the tennis ball under its mechanical arm. The mechanical arm drops down and collects the tennis ball

and checks what the current ball count is. If the target has been reached, Ball Man will seek out the orange cone to drop off the balls. Otherwise, it continues its behavior of avoiding people whilst searching for tennis balls. Appropriate feedback is given throughout Ball Man's functioning, and an enter key is used for confirmation and initialization in its startup.

INTRODUCTION

While you play a game of tennis, it can be a nuisance to pause your game to go and collect stray tennis balls. So to resolve this, I decided to build an autonomous robot that will retrieve these stray tennis balls. This autonomous robot will be able to detect a stray tennis ball on a typical tennis court, approach it at a high velocity, and use a mechanical arm to pick up the tennis ball and place it into a holder. Once three tennis balls have been retrieved, the robot will return to a home station and drop off the balls.

This paper explains both the electrical and physical components of the robot and will describe in detail the functions of the robots.

INTEGRATED SYSTEM

The processor used in Ball Man is an Atmega128. It was highly recommended by the TA's and many other students in the class had decided to use an Atmega128 as well. The chip contains many options, including over 50 I/O pins, 4 timers (two 8-bit, two 16-bit), two USARTS, 8 A/D's multiplexed, many external interrupts, and the chip has a very useful free C compiler. The board that I bought was a very basic header board from Sparkfun. The board's price is what appealed to me ($35), and the TA for Senior Design,

Aaron Chinault, suggested this board for me. Other than creating a breakout board for this board, it has been a great investment.

The programming was done in Programmers Notepad, which uses the AVR-GCC compiler. I initially began coding in CodeVisionAVR, however I began running into its evaluation limitations very soon, and so I had to switch to WinAVR's Programmers Notepad. The C programming was very straightforward and easy to manipulate as compared to doing it in Assembly or another language. There were many coding examples from previous semester's final reports to assist me in using AVR-GCC, and AvrFreaks.net was useful as well. I opted to use PonyProg to download my hex files to my Sparkfun board because it was very simplistic and easy to use. A basic ISP programming cable was also purchased from Sparkfun for this.

The code was written in sections and as more behaviors were developed, they were added onto a main piece of code that I kept appending. This provided useful because even though behaviors might work well in different pieces of code, once combined, many errors can occur. One recurring error was mixing up port functions with their alternate port functions. For example, the external timer pins are on Port D, however I was using these pins as digital inputs, and so this confused the Atmega128 and caused a variety of errors when I combined these behaviors.

Power was derived from 8 AA batteries. They were rechargeable Energizer batteries. The only downside is that they were 1.2V, so I wasn't putting my motors at full capacity when they were running. This voltage was used directly by the microprocessor (it had a voltage regulator onboard), the CMU camera, and the three motors (two for motion, one for the arm). A 5V 7805 regulator was also mounted onto the powerboard to

power the LCD screen, the logic for the motor drivers and direction boards, and for the servo that is used as the trapdoor for the ball canister. The only problems that occurred were when a small short would occur due to poor wiring. I would highly suggest to anyone that you use properly made cables the entire while, and not even bother with alligator clips for testing, because this will waste more time than it might seem to save.

The LCD screen was an EL backlit screen suggested by Will. Although the inverter was a pain to get to work properly and the screen could be iffy, once it was working, it looked extremely cool. Coding for the LCD screen was provided in a library file in CodeVisionAVR, however when I switched to WinAVR, I had to write my own code. This became easier than I realized, though it wasn't possible to output floating point variables to the LCD screen for precise sonar calculations. The LCD proved to be very useful for debugging while attempting all of the behaviors necessary by Ball Man.

The CMU camera and sonar integration will be further discussed in the SENSORS and BEHAVIORS sections.

MOBILE PLATFORM

The platform was cut out of 1/8" thick balsa wood that was provided from laboratory materials. The design was created in AutoCad 2005, but was transferred over to a AutoCad 2001 format to allow IsoPro to open the appropriate DXF files. Although I had no previous experience in any form of CAD whatsoever, the crash course and the homework assignment aided me greatly.

Sketches and random drawings were helpful, but in the end, things were redesigned on the fly in AutoCad. This was because there were limitations in what you

could design based on your devices. Furthermore, it was better to build a wide open platform to decide where you would want to mount your devices and sensors before you made your final platform.

My initial design resembled a typical TJ design. The Sparkfun board and the LCD screen would be the only boards on the surface, whilst everything else would be tucked away in a box underneath the robot. My sonar devices were mounted below the surface of the robot to allow for better detection of such items as rocks and tennis racquet covers.

The motor hubs were not the best of designs, though I found it to be very simple for adjustments and such. Since I wound up destroying two motors, replacing them in the design was quite easy.

Many of my mounting holes were not planned in my AutoCad design, but I cut them out with a drill as I deemed it necessary.

ACTUATION

DC motors allow the robot to travel at greater speeds than with servos. The motors used to actuate Ball Man are Jameco 300 RPM 12V motors. These motors are extremely speedy however have fairly low torque. The wheels used were Du-Bro model airplane wheels, 3.5" in diameter. These wheels are great, and were easy to attach to the Jameco motors (no hubs were used, I just slid them on). The motor driver used to control the motors was an NJM2670, a decent dual h-bridge driver. However, it requires two pwm signals per output, and that was just extraneous. Aaron Tucker and I decided to create a direction board in addition to our motor driver board. This allows us to use only pwm signal and one digital bit to control the signals going to the motor board. The

digital bit is AND'ed with the PWM signal and the inverse of the digital bit is also AND'ed with the PWM signal. This outputs two signals, and allows one to always be ground and one to be the pwm signal.

The final component of the actuation is the rear caster wheel. This isn't a typical caster wheel, it is an omni-wheel. It was found on Acroname, but Dan Huang from a pervious semester sold me the omni-wheel. It's very versatile; it can roll forward, and side to side. The only problem is that it is quite loud on all surfaces, so I would probably look to finding a "quieter" caster wheel if I was to do this again.

SENSORS

a) **CMU camera** – The CMU camera has the ability to detect various types of colors using both the RGB and YCrCb schemes; the latter adds in an illumination factor. In this particular application, the CMU camera initializes to the color of the tennis ball and will seek it out.

b) **Ultrasound** - Two Devantech SRF-04 Ultrasonic Rangers were used for obstacle avoidance. These were placed at approximately 45 degrees off of the vertical and horizontal and allowed each sonar sensor to let the motors know what obstacles lay ahead.

c) **Bump Sensor** - A simple bump sensor is used as an "enter-key" to initialize the color values and to allow the user to step through the beginning of the program.

d) **Photo-resistor** - The photoresistor is used to indicate to the CMU camera whether or not white balance should be turned on or off, based on the amount of light being received by the photoresistor.

More information about these individual sensors can be read in the Special Sensor Report.

BEHAVIORS

Ball Man follows a very distinct pattern and set of behaviors. Upon initialization of the appropriate tennis ball color, the robot immediately begins surveying the tennis court. Obstacle avoidance is enabled as long as Ball Man doesn't sense that there is a tennis ball nearby. The sonars are set to constantly fire until one gets a reading. Depending on which sonar gets the reading, its adjacent motor is signaled to speed up and the other to slow down, so as to "avoid" the object in realtime. If however both sonars get a close reading, it realizes that it will not be able to maneuver around such an object, and so must back up, pivot counter-clockwise, and then continue until it finds an open path.

Once it notices that there is a tennis ball close by, obstacle avoidance shuts off and the lining up of the tennis ball begins. Using CMU x-y coordinates based on the middle mass of the tennis ball, Ball Man will position itself until it feels that its mechanical arm is able to grab the tennis ball. Once that has been established, the mechanical arm lowers itself unto the tennis ball, retrieves it, and stores it in its container. The mechanical arm is run off of a 4-rpm Jameco Motor. This provides enough torque to lower the arm and to lift up the tennis ball once it has placed itself over the ball. After

Ball Man performs this behavior for the target number of tennis balls set (for now, three), it seeks out its home base and proceeds to drop off the balls at this location. Instead of trying to track the color of the tennis ball, it seeks out whatever has the largest amount of red in it. A brightly colored orange-red cone is used, and is perfect for any lighting condition. The drop-off mechanism is basically a "gate" that is servo controlled. Since a simple servo can be used for precise angular motion, I can open and close the "gate" that holds in the tennis balls.

EXPERIMENTAL LAYOUT AND RESULTS

The basic construction of the robot relied upon getting sensors and other behaviors working separately and then trying to integrate them all into one cohesive program. So "mini" experiments were conducted throughout to make sure that the A/D was working, the sonar values being read in were fairly accurate, and that the CMU RGB values were acceptable. Specific ranging and CMU tests were run and are covered in the Special Sensor Report. However, overall system experiments were conducted as well. Most problems occurred with the lining up of the ball and how the program resumed itself after an interrupt fired successfully. Once many tests were run to determine the appropriate stop-Y-distance needed to pickup the ball, everything went smoothly.

CONCLUSION

This has been by far my most time consuming and most challenging class ever. It has been my most educational as well; not just for its academic knowledge, but for its real-world knowledge as well. I wish I would've taken EEL4744 prior to this class,

however it IS possible to take them concurrently and still do successfully in both classes (although you have no life whatsoever for said semester). I was able to accomplish all of my goals that I set out for Ball Man, which was surprising to me. Two weeks until demo day, I assumed that I would get the basic behaviors to work, at most. However once the UART and CMU camera began working efficiently, things just fell into place. Thanks to the assistance of Will and Pickles, and the appropriate motivation from Drs. Arroyo and Schwartz, this class has been a success for me.

In the future, I would like to implement an IR mechanism on Ball Man to allow the user to stop and start the robot at anytime. Furthermore, I would like to be able to view the Ball Count from a remote station as well. During the course of Senior Design, I plan on writing a GUI program that will allow a user to submit information to Ball Man and allow Ball Man to relay that information back to it.

Many errors during the construction of Ball Man led me to understand much more about electricity than I could learn from the mundane lab we have in Circuits 1. Things can easily short with each other, and things fry easily. Voltage and current spikes are dangerous things and can cause the most stable machinery to go haywire at any given time. I was able to develop good debugging skills due to this class though, and it gave me the patience to sit down and test everything with a voltmeter (something I've always hated to do).

If I was able to do this project over, I would've spent more time deciding on an appropriate platform. Some of my biggest problems have been trying to figure out how to mount certain components onto the circular platform that I have. Furthermore, the ball

canister and deposit mechanism aren't too aesthetically pleasing, however do give it a

very Rube-Goldberg look.

DOCUMENTATION

Atmel ATMega128 Documentation
http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

AVRFreaks
http://www.avrfreaks.net/

Devantech SRF04 Ultrasonic Range Finder Reference Sheet
http://www.robot-electronics.co.uk/htm/srf04tech.htm

CodeVisionAVR
www.hpinfotech.ro/html/cvavr.htm

References and snippets of code were taken from Anthony Huerca's Sbob Report

APPENDIX

```
//final program!!!

//Copyrighted on April 13th, 20:01
//by Vivek Manoharan


#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/delay.h>

#include <ctype.h>
#include <inttypes.h>
#include <string.h>
#include <math.h>

#define motora          OCR0
#definemotorbOCR2
#defineservo   OCR3A
#define pickup_forward        OCR3B
```

```
#define pickup_backward      OCR3C
#define a 1
#define b 0
#define forward 1
#define backward 0

// Declare your global variables here
//   PIN SETUP:
//     PORTC7 = D7
//     PORTC6 = D6
//     PORTC5 = D5
//     PORTC4 = D4

//     PORTC3 = NC
//     PORTC2 = EN
//     PORTC1 = NC (grounded on the board), but PINC1 as input
//     PORTC0 = RS


const int delaylen = 50;
const int max_balls = 3;

int uart_data[1000];
int recpos = 0;
int readpos = 0;
char uart_mean_data[9]; //to account for the :
int uart_track_data[11];

int vala = 0xDD; //from DD, from D5
int valb = 0xDD;

float sonar_val1;
float sonar_val2;

int mx = 0;
int my = 0;

int mx_backup = 0;
int my_backup = 0;

int s = 0;
int found = 0;

int ball_count = 0;

int slow_val = 0xA0;
```

```
int nogo = 0x00;
//INTERRUPTS!
//UART recieve interrupt

//DELAY fuctions
void delay_ms(int time)
{
        for(int i=0; i < time; i++)
                _delay_ms(1);
}

void delay_us(int time)
{
        for(int i=0; i < time; i++)
            _delay_us(1);
}

void enter_key() //located on PC1
{
        while ((PINA&0x80) == 0x00)
        ;

        delay_ms(200); //software debounce
}

//LCD FUNCTIONS
void lcd_write(void)
{
        PORTC |= 0x04; //Set LCD enable high
        delay_us(100);
        PORTC &= ~(0x04); //set LCD enable low
        delay_us(100);
}


void lcd_init(void)
{
            delay_ms(1000);

    _delay_ms(delaylen);
    PORTC = 0x30;
    lcd_write();

    _delay_ms(delaylen);
    PORTC = 0x30;
    lcd_write();
```

```
        _delay_ms(delaylen);
        PORTC = 0x30;
        lcd_write();

        _delay_ms(delaylen);
        PORTC = 0x20;
        lcd_write();

//      ENTERING 4-BIT MODE      //

        _delay_ms(delaylen);

        PORTC = 0x20;
        lcd_write();

        PORTC = 0x00; //changed from 80 to 00
        lcd_write();


        _delay_ms(delaylen);
        PORTC = 0x00;
        lcd_write();

        PORTC = 0xF0;   //display off, cursor off, blink off (changed from F to 8, worked
with F)
        lcd_write();

        _delay_ms(delaylen);
        PORTC = 0x00;
        lcd_write();

        PORTC = 0x10;   //clear screen, cursor home
        lcd_write();

        _delay_ms(delaylen);
        PORTC = 0x00;                   //was 00
        lcd_write();

        PORTC = 0x60;   //increment cursor to the right, dont shift screen (was 60
        lcd_write();

        _delay_ms(delaylen);

        PORTC = 0x00;
        lcd_write();
```

```
                PORTC = 0xF0;
                lcd_write();

                _delay_ms(delaylen);



//      INITIALIZATION DONE!    //



}

void lcd_put_ascii(int input)
{

        //set up the nibbles
        int input_high_nib = input & 0xF0;//high nibble
        int input_low_nib = input <<= 4;        //low nibble


        //make sure RS = 1, data mode
        input_high_nib |= 0x01;
        input_low_nib |= 0x01;



        PORTC = input_high_nib;
    lcd_write();
        PORTC = input_low_nib;
    lcd_write();
}


//puts a character onto the LCD display
void lcd_put_char(char data)
{
        lcd_put_ascii(toascii(data));
}

void lcd_puts(char string[])
{
        int i = 0;
    while(i<strlen(string))
        {
            lcd_put_char(string[i]);
            i++;
```

```
        }
}

void lcd_put_int(int data)
{
        int hundred, ten, one;
        int temp = data;

        hundred = temp / 100;
        temp = temp % 100;

        ten = temp / 10;
        temp = temp % 10;

        one = temp;

        hundred += 48;
        ten += 48;
        one += 48;

        if(hundred != 48)
            lcd_put_ascii(hundred);
        if(ten != 48)
            lcd_put_ascii(ten);
//      if(one != 48)
            lcd_put_ascii(one);

}

//clear LCD screen
void lcd_clear()
{
        PORTC &= 0xFE;              //set to command mode
        PORTC = 0x00;
   lcd_write();
        PORTC = 0x10;
   lcd_write();

        PORTC = 0x00;
   lcd_write();
        PORTC = 0xC0;
   lcd_write();

        PORTC |= 0x01;             //enable back to data write mode

        delay_ms(10);
```

```
        }

//sets the direction of whichever motor is chosen
//motor:
//            0 - motor A, d-bit is PA4
//            1 - motor B, d-bit is PA5
//direction:
//            0 - backwards
//            1 - forward

//CHANGING TO PORTA!!

void direction_set(int motor, int dir)
{
        if(motor == 0) //motor A
        {
            if(dir == 0)
                PORTA &= 0xDF;
            if(dir == 1)
                PORTA |= 0x20;
        }
        else
            if(motor == 1) //motor B
            {
                if(dir == 0)
                        PORTA &= 0xEF;
                if(dir == 1)
                        PORTA |= 0x10;
            }

}


void sonar_test()
{
    //Requires Two Pins!  PortB(0:1)
    //PORTB0     Pulse Enable (DDR Output)
    //PINB1     Echo        (DDR Input)

            //changing it to timer1


            TCCR1B = 0x00;  //stop timer
            TCNT1 = 0x00;
```

```
        PORTB|=0x01;        //Set Pulse Enable High
    delay_us(10);       //10 uSec Delay
    PORTB&=0xfe;        //Set Pulse Low


       while ((PINB&0x02) == 0x00)   //Poll Echo For Rising Edge
    {
    };

           TCCR1B = 0x05;  //start timer

   while ((PINB&0x02) == 0x02)   //Poll Echo For Falling Edge
    {
    };

    TCCR1B=0x00;        //Stop Timer
    sonar_val1=TCNT1*1.;       //Save Result  removed *1.

    delay_ms(10);       //Mandatory Delay
    sonar_val1 =  ((sonar_val1) * .432); //have to multiply it by 64/148

//SONAR TWO///////////////////////////////////////////

           TCCR1B = 0x00;  //stop timer
           TCNT1 = 0x00;

           PORTA|=0x01;        //Set Pulse Enable High
    delay_us(10);       //10 uSec Delay
    PORTA&=0xfe;        //Set Pulse Low


       while ((PINA&0x02) == 0x00)   //Poll Echo For Rising Edge
    {
    };

           TCCR1B = 0x05;  //start timer

   while ((PINA&0x02) == 0x02)   //Poll Echo For Falling Edge
    {
    };

    TCCR1B=0x00;        //Stop Timer
    sonar_val2=TCNT1*1.;       //Save Result  removed *1.

    delay_ms(10);       //Mandatory Delay
    sonar_val2 =  ((sonar_val2) * .432); //have to multiply it by 64/148
```

```
}

void turn_around()
{
        int dbit1, dbit2;

        lcd_clear();
        lcd_puts("Found Wall!");
        motorb = 0x00;              //stop
        motora = motorb;

        delay_ms(1000);

        direction_set(a, 0);        //reverse direction
        direction_set(b, 0);

        motorb = 0xA0;
        motora = 0xA0;

        delay_ms(2000);

        dbit1 = 1;
        dbit2 = 0;

        direction_set(a, dbit1);    //pivot in place 0
        direction_set(b, dbit2);    //                              1

        sonar_test();

        while(! ((sonar_val1 > 25) && (sonar_val2 > 25)) )
            sonar_test();

        lcd_clear();
        lcd_puts("Full Speed Ahead!");
        direction_set(a, 1);        //forward
        direction_set(b, 1);

        motorb = valb;
        motora = vala;
}

//UART FUNCTIONS
void uart_write(unsigned char data)
{
        while ( !( UCSR0A & (1<<UDRE0)) )
        ;
```

```
        UDR0 = data;
}
unsigned char uart_recv( void )
{
        /* Wait for data to be received */
        while ( !(UCSR0A & (1<<RXC0)) )
            ;

        /* Get and return received data from buffer */
        return UDR0;
}

void uart_flush( void )
{
        unsigned char dummy;
        while ( UCSR0A & (1<<RXC0) )
            dummy = UDR0;

        recpos = 0;
        readpos = 0;
}

void uart_puts(char string[])
{
        int i = 0;
    while (i<strlen(string))
    {
      uart_write(string[i]);
            i++;
    };

}

void uart_print()
{
        for(readpos; readpos < recpos; readpos++)
        {
            //if(!((uart_data[readpos] < 48) || ((uart_data[readpos] > 57) &&
            //   (uart_data[readpos] < 64)) || (uart_data[readpos] > 122)))
                lcd_put_char(uart_data[readpos]);
        }

        readpos = 0;
        recpos = 0;
}
```

```
void uart_print_int()
{
        int i = 0;

        for(i; i < recpos; i++)
        {
            lcd_put_int(uart_data[i]);
            lcd_puts(" ");
        }

        readpos = 0;
        recpos = 0;
}

void uart_receive()
{
        while ((UCSR0A&0x80))
        {
            lcd_put_ascii(uart_recv());
        }
}


//CMU CAMERA FUNCTIONS
void camera_init()
{
        lcd_clear();

        //UART SETUP
        UBRR0H = 0x00;
        UBRR0L = 0x08; //changing speeds to 08 from 33
        UCSR0B = 0x98;
        UCSR0C = 0x06;

        uart_puts("RS \r"); //resets cmu
        delay_ms(10);
        uart_puts("RM 3\r");  //sets raw mode output and acks disabled
        delay_ms(10);
        uart_puts("CR 18 44\r");//white balance on
        delay_ms(10);
        uart_puts("MM 1\r"); //middle mass on
        delay_ms(10);

        delay_ms(10);
        uart_print();
```

```c
}

void drop_off()
{
        //make servo swing arm out

        TCCR3A = 0xA0;      //1010 | 0000
        TCCR3B = 0x12;      //0001 | 0010
        ICR3 = 20000;

        servo = 2500; //increase to adjust, from 500

        delay_ms(5000);

        servo = 1700; //decrease to adjust

        delay_ms(2000);

        TCCR3A = 0x00;
        TCCR3B = 0x00;

}

void find_cone()
{
        s = 0;
        cli();

        uart_puts("RS \r"); //resets cmu
        delay_ms(10);
        uart_puts("RM 3\r");  //sets raw mode output and acks disabled
        delay_ms(10);
        uart_puts("CR 18 44\r");//white balance on
        delay_ms(10);
        uart_puts("MM 1\r"); //middle mass on
        delay_ms(10);

        //set color for cone
        //
        uart_puts("TC 200 240 0 25 0 25\r");
        delay_ms(1000);

        uart_flush();

        lcd_clear();
```

```
        lcd_puts("find cone!");

        sei();
        s = 1;
}

void arm_down()
{

        lcd_clear();
        lcd_puts("arm moving down!");

        pickup_forward = 0xffff;
        pickup_backward = 0x00;

        delay_ms(8300);

        pickup_forward = 0x00;
        pickup_backward = 0x00;

}

void arm_up()
{
        lcd_clear();
        lcd_puts("arm moving up!");

        pickup_forward = 0x00;
        pickup_backward = 0xffff;

        delay_ms(8000);

        pickup_forward = 0x00;
        pickup_backward = 0x00;


}

void pick_up()
{

        lcd_clear();
        lcd_puts("pickup mechanism");
        delay_ms(3000);

        //timer3, ocr3b/c used for two pins
```

```c
        TCNT3 = 0x00;

        TCCR3A = 0xAB;
        TCCR3B = 0x11;

        arm_down();
        arm_up();


        ball_count++;

        lcd_clear();
        lcd_puts("Picked up ");
        lcd_put_int(ball_count);
        lcd_puts(" balls!");

        delay_ms(5000);

        found = 0;

        mx = 0;
        my = 0;

        if(ball_count == max_balls)
            find_cone();


}

void line_up_cone()
{
        lcd_clear();
        lcd_puts("X: ");
        lcd_put_int(mx);
        lcd_puts(" Y: ");
        lcd_put_int(my);

        sonar_test();

        if((sonar_val1 < 20) && (sonar_val2 < 20))
        {
            motora = nogo;
            motorb = nogo;

            cli();
            ball_count++;
```

```
                drop_off();
            }

        if((mx > 20) && (mx < 58))
        {
            direction_set(a, 1);
            direction_set(b, 1);
            motora = slow_val;
            motorb = slow_val;
        }
        else
        {
            if(mx > 58)
            {
                direction_set(a, 0);
                direction_set(b, 1);
                motora = slow_val;
                motorb = slow_val;
            }

            if(mx < 50)
            {
                direction_set(a, 1);
                direction_set(b, 0);
                motora = slow_val;
                motorb = slow_val;
            }
        }

}


void line_up_ball()
{
        lcd_clear();
        lcd_puts("X: ");
        lcd_put_int(mx);
        lcd_puts(" Y: ");
        lcd_put_int(my);

        if(my > 50)
        {
            if((mx > 20) && (mx < 58))
            {
```

```
                    direction_set(a, 1);
                    direction_set(b, 1);
                    motora = slow_val;
                    motorb = slow_val;
                }
                else
                {
                    if(mx > 58)
                    {
                            direction_set(a, 0);
                            direction_set(b, 1);
                            motora = slow_val;
                            motorb = slow_val;
                    }

                    if(mx < 50)
                    {
                            direction_set(a, 1);
                            direction_set(b, 0);
                            motora = slow_val;
                            motorb = slow_val;
                    }
                }

        }
        else
        {
            if((my < 50) && (my > 20))
            {
                if((mx > 57) && (mx < 70))
                {
                            direction_set(a, 1);
                            direction_set(b, 1);
                            motora = slow_val;
                            motorb = slow_val;
                }
                else
                {
                            if(mx > 70)
                            {
                                    direction_set(a, 0);
                                    direction_set(b, 1);
                                    motora = slow_val;
                                    motorb = slow_val;
                            }
```

```c
                        if(mx < 50)
                        {
                                direction_set(a, 1);
                                direction_set(b, 0);
                                motora = slow_val;
                                motorb = slow_val;
                        }
                }
        }
        else //safe to assume its right underneath the arm mechanism
        {
            cli();
            lcd_clear();
            lcd_puts("arm time");

            motora = nogo;
            motorb = nogo;

            delay_ms(2000);

            pick_up();

            found = 0;
        }
    }
//      delay_ms(2000);

        mx_backup = mx;
        my_backup = my;

}

void avoid_obstacle()
{
        motora = vala;
        motorb = valb;

        sonar_test();

        if(sonar_val1 < 35)
        {
            motora = 0xFF;
            while(sonar_val1 < 20)
            {
                motorb = 0x50;
                sonar_test();
```

```c
            if((sonar_val1 < 20) && (sonar_val2 < 20))
                    turn_around();
        }
    }

    sonar_test();

    if(sonar_val2 < 25)
    {
        motorb = 0xFF;
        while(sonar_val2 < 20)
        {
            motora = 0x50;
            sonar_test();

            if((sonar_val1 < 20) && (sonar_val2 < 20))
                    turn_around();
        }
    }

}


SIGNAL(SIG_UART0_RECV)
{
    //get data
    if(recpos == 900)
    {
        recpos = 0;
        readpos = 0;
    }
    unsigned int foo=UDR0;

    if(uart_data[(recpos-2)] == 77)
    {
        mx = uart_data[(recpos-1)];
        my = foo;
        if(s)
        {
            if(mx || found)
            {
                    if(!found)
                    {
                            //cli();
                            motora = 0x00;
```

```c
                            motorb = 0x00;
                            lcd_clear();
                            lcd_puts("found object!");
                            delay_ms(2000);
                            found = 1;

                    }
                    else
                    {
                            if(mx == 0)
                                    mx = mx_backup;
                            if(my == 0)
                                    my = my_backup;

                            if(ball_count == max_balls)
                                    line_up_cone();
                            else
                                    line_up_ball();
                    }

            }
        }
        else
            lcd_clear();
    }

    uart_data[recpos++]=foo;
}

//MAIN FUNCTION
int main(void)
{
    delay_ms(500);

    DDRC = 0xFF;  //bits: O O O O | O O I O
    DDRB = 0x91;
    DDRE = 0x3A; //bits: I I O O | O I O I
    DDRA = 0x31;

    sei();

    direction_set(0, forward);
    direction_set(1, forward);
```

```
lcd_init();
lcd_puts("BallMan v1.0");
delay_ms(2000);
lcd_clear();
lcd_puts("by Vivek Manoharan");

enter_key();

lcd_clear();

/////////////////////////////
//set up the motor pwms
TCNT2=0x00;
TCCR2=0x61; //0x61 for 00 being off, FF being full speed

TCNT0=0x00;
TCCR0=0x61;
/////////////////////////////

camera_init();
enter_key();
uart_flush();

lcd_clear();
lcd_puts("place object in window!");
cli();
enter_key();

lcd_clear();
uart_puts("TW\r");
delay_ms(5000);
sei();

motora = nogo; //vala
motorb = nogo; //valb

delay_ms(1000);

uart_flush();

s = 1;

while(ball_count <= max_balls) //adding the = to for testing
{
    if(!found)  //commented out to test ranges of x and y allowable
    {
```

```
            avoid_obstacle();
        }

    };


    cli();
    lcd_clear();
    lcd_puts("done!");

}
```