University of Florida
Department of Electrical and Computer Engineering
EEL 5666 Intelligent Machines Design Laboratory
Spring 2006

Helper

Final Written Report

By: Bradley Morin

Date: 25 April, 2006

TAs:
Adam Barnett
Sara Keen

Instructors:
A. A. Arroyo
E. M. Schwartz

Table of Contents

Abstract

Helper is a completely autonomous robot that picks up objects and sorts them into their correct container based on color. Golf balls of four different colors (red, green, blue, yellow) are placed inside of a white area bounded by black lines as well as four colored containers. Helper locates the objects, picks them up, takes the object to its correct container based on color, and drops the object into the box while avoiding obstacles and staying within the boundary line. This paper will describe how Helper is able to accomplish this task.

Executive Summary

Helper was designed and built entirely during the fall 2006 semester in the University of Florida's Intelligent Machine Design Laboratory Course. Helper is an autonomous robot capable of locating objects of four different colors and taking them to corresponding locations.

Helper uses four sensors to accomplish its task. Bump switches and IR sensors are used for obstacle avoidance and collision detection. Two Hamamatsu photoreflectors are used to detect a black boundary line which Helper will not cross. The special sensor used by Helper is the CMUCam. This is used for object recognition and color detection.

Helper is powered by an ATmega128 microcontroller which interfaces with all of the components that make up Helper. Helper uses dc motors for movement and servos to control its gripper used to pick up and drop off objects. Helper uses 12 AA batteries to power all of its components. This voltage is passed through regulators to create voltages of 5, 6, and 12 volts for the different components that make up Helper. An LCD screen on top of the robot is used for feedback.

This paper will go into greater detail of the components used to create Helper and the behaviors used to accomplish the task of sorting the objects by color.

Introduction

My robot, Helper, picks up objects and sorts them into their correct container based on color. The motivation for Helper is for users or families with multiple children or pets that may leave their toys out all over the room at the end of the day. Helper attempts to solve the problem of cleaning up the room. Helper would be able to locate the objects in the room and take them to their proper location for storage (i.e. one bin for the cat's toys, one for the dog's toys, and another for the children's toys).

For demonstration and prototyping purposes, the objects are plastic golf balls painted red, green, blue, and yellow. There are four containers painted in these same colors that Helper drops the golf balls off into. In the process of clearing the room, Helper stays within a boundary to keep out of certain rooms or going down the stairs for example. For the demonstration, the room is represented by a white poster board marked off with a black boundary line around the edges. In addition, Helper performs obstacle avoidance to avoid running into a table or one of the pets for example.

This paper will detail everything about Helper. The robot's integrated systems, mobile platform, actuation, sensors, and behaviors will all be detailed in this paper.

Integrated System

Helper is controlled by an ATmega128 microcontroller onboard of a Maveric IIB development board purchased from BDMicro. I choose to use the Maveric IIB board because of its many appealing features such as numerous I/O pins, memory size, onboard voltage regulators, analog to digital converters, and PWM channels for Servo control. The microcontroller board is interfaced with all the sensors and LCD screen, controls the servos, and interfaces with the motor driver. Figure 1 below shows a block diagram for the components making up Helper.

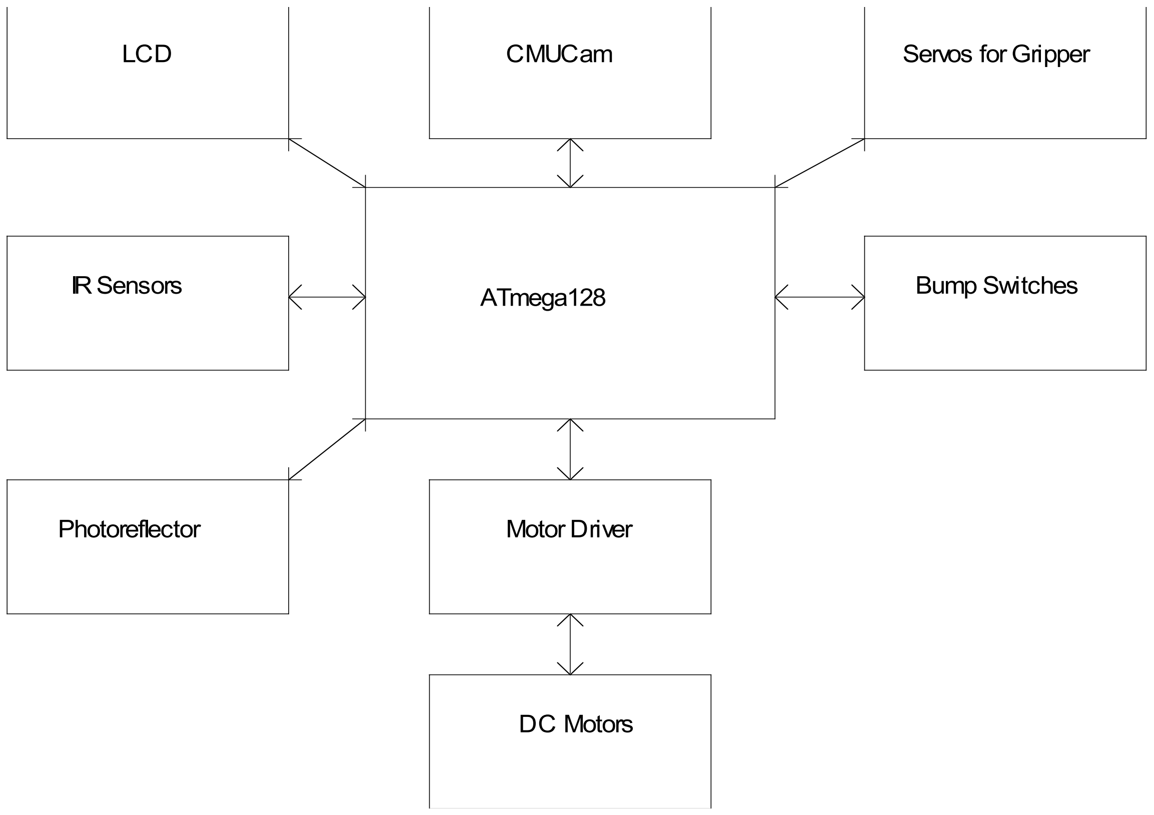

Figure 1: Block Diagram of Robot Components

Mobile Platform

The platform for Helper is a dual tier design that consists of two modified circles. The circles are 12 inches in diameter. A small portion of the front of the circles is cut off, creating a flat area where the camera, gripper, and IR sensors are mounted. The majority of the electronics will be mounted in between the top and bottom layers. The exceptions are the LCD and photoreflector circuits which will be mounted on the top and bottom of the robot respectively.

This simple design was chosen in order to maximize the time spent on electronics and programming. In the future, I would like to redo the platform design. At the very least, I would like to make it smaller (probably around 9 or 10 inches in diameter.

Actuation

Helper has two forms of actuation in order to perform the desired task of moving an object from one location to another. For movement, Helper uses two 12 V, 200 rpm gear head motors. I choose to use motors for this application because they should be longer lasting than servos and more precise for positioning the robot for pick-up and drop-off. The motors are attached to a pair of 2.13" diameter wheels with a hub. The motors, wheels, hubs, and mounts were ordered from lynxmotion. I built my own motor controller board for Helper. The motor controller board uses a SN754410 Quadruple Half H Driver that is modified to be used as a dual Full H Bridge. The board takes in two PWM signals and two direction signals generated by the microcontroller for each motor.

The other form of actuation that is used is a gripper to pick up objects. The gripper requires two degrees of freedom. The jaws of the gripper open and close in order to grip and release objects. Additionally, the gripper lifts objects off the ground in order to drop them into containers. The gripper is controlled using two servos controlled by PWM signals from the microcontroller. Plastic golf balls were chosen to be lifted in order to ensure the wrist will be able to lift the objects without damaging or breaking the servo.

Sensors

- *Bump Switches* - The bump switches used on Helper are simple push button switches. They are placed periodically around Helper and are used for collision detection. By placing two switches in the front, two on each side, and two in the rear of the robot, a collision with an object can be detected. When a collision is detected, the robot stops and turns off in order to avoid damaging the motors or motor driver by attempting to move in a direction it can not.
- *IR Sensors* - The IR Range Finders being used on Helper are the Sharp GP2D12 IR sensors. Three IR sensors are placed on the front of the robot, one pointing straight ahead that is located in the middle and two pointed approximately 45 degrees outward from center that are located on the left and right sides.

The IR sensors are used for detecting objects in front of the robots path in order to avoid collisions. The IR sensors output a single analog signal that corresponds to the reflected light intensity, the higher the intensity the higher the value. This signal will be connected to one of the onboard A/D converters of the Maveric II board so that it can be used to allow Helper to detect nearby objects and move accordingly to avoid them. Table 1 below shows the digital IR signal value of the middle sensor with the robot is positioned different distances from an object (a wall in this experiment). The values fluctuate, so approximate ranges of the values are listed in the table.

Distance	**Digital Value**
2’	90-97
1’ 6”	123-127
1’	184-189
10”	218-223
8”	270-277
6”	365-370
5”	419-423

4”	497-502
3”	525-529

Table 1: IR Sensor Digital Values

At distances of less than approximately three inches, the digital values begin to decline, making them unable to be interpreted appropriately. Therefore, Helper attempts to ensure correct actions are taken within about six inches of an object so that it does not get too close for the sensor values to be incorrect.

- *Photoreflectors* - Two Hamamatsu Photoreflectors will be used to detect the black boundary line of the environment. They will be placed under the robot toward the front, one on each side. The photoreflectors are implemented in the circuit below in Figure 1. This circuit was obtained from the acroname website where the photoreflectors were purchased.

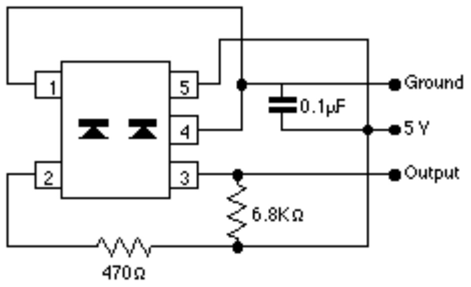

Figure 1: Typical circuit for photoreflectors

When used in the above circuit, the output of the photoreflectors is a logic “1” when the device is placed over a white surface and is a logic “0” when placed over a black surface. This allows Helper to sense when it has reached the black boundary line around the environment. Additionally, using two photoreflectors allows Helper to detect its orientation relative to the border line so that it can maneuver away form the border accordingly.

- *CMUCam* - A CMUCam is used by Helper for object detection and color recognition. Interfacing the CMUCam with the computer using the provided software is the first step necessary to understand the camera and its command set. Frames can be viewed on the computer using the software.

Using the PC software to capture frames, the pictures in the Figures below were taken of the four different color golf balls are shown under different conditions. The picture in Figure 2 was taken with no LED lighting and the camera's auto white balance off. The picture in Figure 3 was taken with no LED lighting and the camera's auto white balance turned off. The picture in Figure 4 was taken with LED lighting assistance and the auto white balance off. Finally, the picture in Figure 5 was taken with both LED lighting and the camera's auto white balance turned on. The LEDs refers to two circuit boards consisting of four bright white LEDs located placed on each side of the CMUCam. The auto white balance is a register setting that can be turned on/off in software.

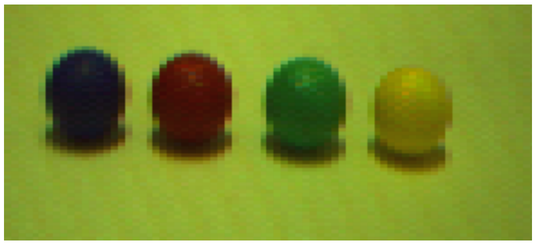
Figure 2: No LEDs and no auto balance

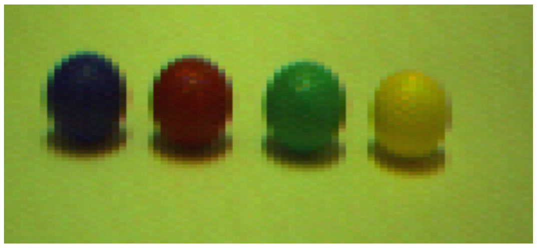
Figure 3: No LEDS and auto balance on

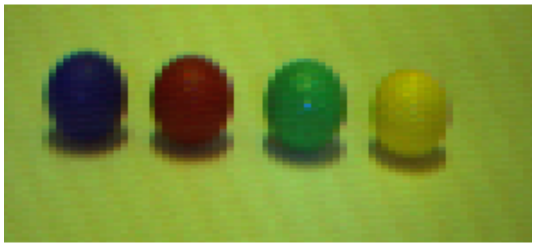
Figure 4: LEDs on and no auto balance

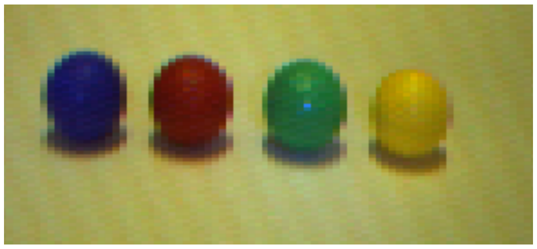
Figure 5: LEDs on and auto balance on

This experiment was done in order to determine the best condition in which to locate the golf balls. From the picture it can be seen that the golf balls are the brightest most distinguished in Figure 5 when both the LEDs and the camera's auto white balance were turned on.

Instructions that are used by Helper to locate objects and determine their color include the Get Mean function and the Track Color function. The Get Mean function returns the red, green, and blue mean color values in the current window. Table 2 shows the values returned by the Get Mean command for the four different colored golf balls Helper locates. In this test, the golf balls are placed approximately 3 inches from the camera. The LEDs and the camera's auto white balance were turned on. The values in the table are approximate averages of using the GM command three times.

Color	**Red Mean**	**Green Mean**	**Blue Mean**
Red	142	63	43
Green	94	111	53
Blue	92	64	128
Yellow	133	93	33

Table 2: CMUCam Get Mean Return Values

The Track Color command returns the bounding coordinates of an object that falls within a specified minimum and maximum red, green, and blue mean values. The values from the previous experiment (Table 2) can then be used with the track color command to determine if an object of that color resides in the current window. Information regarding the objects location in the window is returned such as the x and y coordinates of the middle of the object. With this information, the microcontroller can then position Helper to pick-up the object. The track color command will return the number of pixels that make up the object. This allows Helper to distinguish between a golf ball and the much larger drop-off containers of the same color. Finally, the track color command return a confidence level that helps from accidentally mistaking objects in the surrounding areas as the golf ball that it is looking for.

The CMUCam was easy to use and very useful with its built in command set. I would recommend the use of this sensor by anyone in the future in need of a camera to track a color or differentiate between different colors.

Behaviors

Helper performs the following behaviors in order to complete its tasks. The robot searches a room to locate an object using the CMUCam. If no object is in the current view, Helper rotates in a random direction looking for an object, if nothing is detected the robot moves to another portion of the room and looks again. Once located, Helper approaches the object until it reaches a calculated distance. Once in place, Helper lowers its gripper to the object by moving the wrist downward. The jaws of the gripper closes upon the object, grasping it firmly. Helper then lifts the object slightly off the ground for transportation.

With the object securely gripped, Helper begins searching the room for the container that corresponds to the current objects color. This is done in the same way Helper locates an object, only this time Helper only looks for one color (that of the object it is carrying). Helper distinguishes the container from the golf ball by the objects size. The container is a larger object occupying more than 40 pixels in the CMUCam window instead of the smaller golf ball which is typically less than 15 or 20 pixels depending on the lighting environment. Once the container is located, helper moves toward it to a calculated distance and releases the object by opening its jaws.

Upon completion, Helper backs up and repeats this process until all the objects in the area have been picked up. All of these behaviors are performed while staying within the boundaries of the room and avoiding collisions with other objects or walls that may be in the room.

Conclusion

I was able to design Helper to meet all of the specifications laid forth in the original project proposal at the beginning of the semester. I was pleasantly surprised with capabilities of the CMUCam and the ease with which I was able to use it. Without these qualities, I would not have been able to complete this project. The CMUCam does have its limitations. I had hoped to identify and sort object by shape in addition to color, but this unrealistic using the CMUCam.

As stated earlier, I would like to redesign the platform in the future. I would like to make it smaller and more compact. Additionally, using a servo to tilt the CMUCam back and forth would increase the range of view the robot currently has and make positioning the objects in front of the robot easier. This is something I would have incorporated into my design from the beginning if I was to start over.

Building my first robot this semester was an experience unlike any other I have had during my education process. It was extremely time consuming, but fun and well worth the time put into it.

Appendix: Program Code

```
************************************************************************
******
Delay Timer Functions:
************************************************************************
******

// code written referencing sample program (hw.c) on BDMicro.com

int msCount = 0;
volatile uint16_t delayCount;

//delay for specified number of milliseconds
void delay50us(uint16_t us)
{
  TCNT0  = 0;
  delayCount = 0;
  while (delayCount != us)
    ;
}

void msDelay(uint16_t ms)
{
        msCount = 0;
        while(msCount != ms)
        {
                delay50us(20);
                msCount++;
        }
}

//initialize timer 0 to generate an interrupt every millisecond.
void init_timer(void)
{
  /*
   * Initialize timer0 to generate an output compare interrupt, and
   * set the output compare register so that we get that interrupt
   * every millisecond.
   */
  TCCR0 = 0;
  TIFR  |= _BV(OCF0);
  TCCR0  = _BV(WGM01)|_BV(CS01);
  TCNT0  = 0;
  TIMSK |= _BV(OCIE0);     // enable output compare interrupt
  OCR0   = 98;             // match in 1 ms
```

```
}


**********************************************************************
******
ADC Functions:
**********************************************************************
******

//code written referencing sample program on BDMicro.com

//Initialize A/D converter to free running, start conversion,
//use internal 5.0V reference, pre-scale ADC clock to 125 kHz (assuming 16 MHz MCU
clock)
void adc_Init(void)
{
  /* configure ADC port (PORTF) as input */
  DDRF  = 0x00;
  PORTF = 0x00;

  ADMUX = _BV(REFS0);
  ADCSR = _BV(ADEN)|_BV(ADSC)|_BV(ADFR) |
_BV(ADPS2)|_BV(ADPS1)|_BV(ADPS0);
}

//Select the specified A/D channel for the next conversion
void adc_chsel(uint8_t channel)
{
  /* select channel */
  ADMUX = (ADMUX & 0xe0) | (channel & 0x07);
}

//Wait for conversion complete.
void adc_wait(void)
{
  /* wait for last conversion to complete */
  while ((ADCSR & _BV(ADIF)) == 0)
    ;
}

//Start an A/D conversion on the selected channel
void adc_start(void)
{
  /* clear conversion, start another conversion */
  ADCSR |= _BV(ADIF);
```

```
}

//Read the currently selected A/D Converter channel.
uint16_t adc_read(void)
{
  return ADC;
}

//Read the specified A/D channel 'n' times and return the average of the samples
uint16_t adc_readn(uint8_t channel, uint8_t n)
{
  uint16_t t;
  uint8_t i;

  adc_chsel(channel);
  adc_start();
  adc_wait();

  adc_start();

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

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

****
******
LCD Functions:
****
******

```
//code written referencing William Dubels sample code on course website

//Send command to LCD
//0xC0 = move cursor to second line
//0x94 = move cursor to third line
//0xD4 = move cursor to fourth line
```

```
void LCD_command(int db)
{
  LCD = (db >> 4) | E0;
  delay50us(1);
  LCD = db >> 4;
        delay50us(5);

  LCD = (db & 0x0F) | E0;
  delay50us(1);
  LCD = db & 0x0F;
        delay50us(100);
}

// Sends the command to clear the display, and resets the position variable (used to track
line wrap).
void LCD_clear()
{
  //Clear Home command
  LCD_command(0x01);
}

// Initialize a Hitachi 4x20 4 bit mode display
// E1 E0 RW RS D7 D6 D5 D4
void LCD_init()
{
        DDRC = 0xFF;

  delay50us(300);   //15
  //initialize to 4 bit mode:
  LCD_command(0x33);
  delay50us(100);
  LCD_command(0x32);

  //2 lines 5x8 font
  LCD_command(0x28);
  //Display, no Curser, no Blink
  LCD_command(0x0C);
  // F curser and blink, E just curser
}

// Allows you to print a string of data.  It requires null terminated strings.
void LCD_string(char db [])
{
        for (; *db!='\0';db++)
                LCD_data(*db);
}
```

```
//send data to LCD bus
void LCD_data(char db)
{
        //first nibble & strobe
        LCD = (db >> 4) | E0 | RS;
        delay50us(1);
        LCD = (db >> 4) | RS;
        delay50us(1);

        //second nibble & strobe
        LCD = (db & 0x0F) | E0 | RS;
        delay50us(1);
        LCD = (db & 0x0F) | RS;

  delay50us(1);

}

// used to write integer variable to the LCD
void LCD_integer(uint8_t num1)
{
  //first nibble & strobe
  LCD = (0x03) | E0 | RS;
        delay50us(1);
  LCD = (0x03) | RS;
        delay50us(1);
  //second nibble & strobe
  LCD = (num1 & 0x0F) | E0 | RS;
  delay50us(1);
  LCD = (num1 & 0x0F) | RS;

  delay50us(1);
}

// calls LCD_integer while printing number with mutiple digits to LCD
void LCD_writeInt(uint16_t integer)
{
        uint8_t thousands = integer / 1000;
        if (thousands != 0)
                LCD_integer(thousands);

        uint8_t hundreds = (integer - thousands*1000) / 100;
        if (thousands !=0 + hundreds != 0)
                LCD_integer(hundreds);
```

```
	uint8_t tens = (integer - thousands*1000 - hundreds*100) / 10;
	if (thousands !=0 + hundreds != 0 + tens != 0)
		LCD_integer(tens);

	uint8_t ones = (integer - thousands*1000 - hundreds*100 - tens*10);
	LCD_integer(ones);
}


***********************************************************************
******
CMUCam Functions:
***********************************************************************
******

volatile unsigned char RxBuffer[9];

//initialize desired parameters for UART1
void UART1_init(void)
{
	//set baud rate to 115.2k
	UBRR1H = 0x00;
	UBRR1L = 0x08;

	//enable reciever and transmitter
	UCSR1B = 0x18;

	//no parity bit, 1 stop bit, and 8 bit data
	UCSR1C = 0x06;
}

//transmit message
void UART1_Tx(char data[])
{
	int x = 0;
	while (data[x] != 0) {
		//wait for empty transmit buffer (code obtained from ATmega128 manual)
		while ( !( UCSR1A & (1<<UDRE1)) )
			;

		UDR1 = data[x];
		x++;
	}
}

//recieve data
```

```
unsigned char UART1_Rx(void)
{
	//wait for data to be received (code obtained from ATmega128 manual)
	while ( !(UCSR1A & (1<<RXC1)) )
		;

	return UDR1;
}

//flush UART RX buffer
void UART_flush(void)
{
	unsigned char dummy;

	while ( UCSR1A & (1<<RXC1) )
		dummy = UDR1;
}

//set up cmucam for desired functionality
void CMUCam_init(void)
{
	//reset camera
	UART1_Tx("RS\r");
	msDelay(50);

	//set to polling mode (i.e. wait for function call to send packet)
	UART1_Tx("PM 1\r");
	msDelay(50);

	//set to raw data mode (i.e. not to include spaces or readable ASCII text)
	UART1_Tx("RM 3\r");
	msDelay(50);
}

//call the camera's Get Mean Value fuction (returns 7 bytes of data)
void CMUCam_GM(void)
{
	int x = 0;
	unsigned char dummy;

	UART1_Tx("GM\r");

	//framing bit
	dummy = UART1_Rx();

	//receive 7 bytes of data, store in RxBuffer
	while (x < 7) {
		RxBuffer[x] = UART1_Rx();
		x++;
	}

	//framing bit
	while (dummy != ':')
		dummy = UART1_Rx();
}

//call the camera's track color command to track an object of red color
//recieve data in the form: TC[Rmin Rmax Gmin Gmax Bmin Bmax]\r
//returns  bytes of data in the form: M mx my x1 y1 x2 y2 pixel confidence\r
void trackRed(void)
{
	int x = 0;
	unsigned char dummy;

	UART_flush();

	UART1_Tx("TC 100 200 10 70 10 50\r");
	//these values experumentally obtained using GM function

	//framing bit
	dummy = UART1_Rx();

	//receive 9 bytes of data, store in RxBuffer
	while (x < 9) {
		RxBuffer[x] = UART1_Rx();
		x++;
	}

	//framing bit
	while (dummy != ':')
		dummy = UART1_Rx();
}

//call the camera's track color command to track an object of red color
//recieve data in the form: TC[Rmin Rmax Gmin Gmax Bmin Bmax]\r
//returns  bytes of data in the form: M mx my x1 y1 x2 y2 pixel confidence\r
void trackBlue(void)
{
	int x = 0;
	unsigned char dummy;

	UART1_Tx("TC 10 80 10 80 100 200\r");
	//these values experumentally obtained using GM function

	//framing bit
	dummy = UART1_Rx();

	//receive 7 bytes of data, store in RxBuffer
	while (x < 9) {
		RxBuffer[x] = UART1_Rx();
		x++;
	}

	//framing bit
	while (dummy != ':')
		dummy = UART1_Rx();

}

//call the camera's track color command to track an object of red color
//recieve data in the form: TC[Rmin Rmax Gmin Gmax Bmin Bmax]\r
//returns  bytes of data in the form: M mx my x1 y1 x2 y2 pixel confidence\r
void trackGreen(void)
{
	int x = 0;
	unsigned char dummy;

	UART1_Tx("TC 10 70 100 200 10 70\r");
	//these values experumentally obtained using GM function

	//framing bit
	dummy = UART1_Rx();

	//receive 7 bytes of data, store in RxBuffer
	while (x < 9) {
		RxBuffer[x] = UART1_Rx();
		x++;
	}

	//framing bit
	while (dummy != ':')
		dummy = UART1_Rx();
}

//call the camera's track color command to track an object of red color
//recieve data in the form: TC[Rmin Rmax Gmin Gmax Bmin Bmax]\r
//returns  bytes of data in the form: M mx my x1 y1 x2 y2 pixel confidence\r
void trackYellow(void)
```

```
{
        int x = 0;
        unsigned char dummy;

        UART1_Tx("TC 140 200 100 200 10 30\r");
        //these values experumentally obtained using GM function

        //framing bit
        dummy = UART1_Rx();

        //receive 7 bytes of data, store in RxBuffer
        while (x < 9) {
                RxBuffer[x] = UART1_Rx();
                x++;
        }

        //framing bit
        while (dummy != ':')
                dummy = UART1_Rx();

}
```

```
***********************************************************************
******
Motor/Servo Functions:
***********************************************************************
******
```

```
//Initialize setting for Timer 3 to control the PWM signals for the servos
void servo_Init(void)
{
        DDRE = 0x38;

        ICR3 = 25000;
        TCCR3A = 0xFC;
        TCCR3B = 0x12;
        TCNT3 = 0x0000;

        OCR3A = 23500;
        OCR3C = 23950;
        msDelay(1000);
        OCR3A = 23100;
}
```

```
//Initialize setting for Timer 1 to control the PWM signals for the motors
void motors_Init(void)
{
        ICR1 = 20000;
        TCNT1 = 0x0000;
        TCCR1A = 0xFC;
        TCCR1B = 0x12;
        OCR1A = 20000;
        OCR1B = 20000;

        DDRB = 0xFC;
        PORTB |= _BV(PORTB4) | _BV(PORTB3) | _BV(PORTB2);
}

//set direction and speed of left motor to specified values
//1 = forward ; 0 = backward
void left_Motor(int dir_L, int16_t speed_L)
{
        if (dir_L == 0)
                PORTB &= ~(_BV(PORTB4));

        else
                PORTB |= _BV(PORTB4);

        OCR1B = speed_L;
}

//set direction and speed of right motor to specified values
//1 = forward ; 0 = backward
void right_Motor(int dir_R, int16_t speed_R)
{
        if (dir_R == 0)
                PORTB &= ~(_BV(PORTB3));

        else
                PORTB |= _BV(PORTB3);

        OCR1A = speed_R;

}

//stop both motors
void stop_Motors(void)
{
        OCR1B = 20000;
        OCR1A = 20000;
        msDelay(10);
}

//adjust position of the gripper (servo 1)
//19000 = open fully ; 18000 = closed completely
void move_Gripper(int16_t pos1)
{
        OCR3C = pos1;
}

//adjust position of the wrist (servo 2)
//18500 = middle ; 18000 = compleltely down
//19000 = completely up
void move_Wrist(int16_t pos2)
{
        OCR3A = pos2;
}
```

****
Main Functions:
****

```
 #include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/pgmspace.h>

#include "delay.h"
#include "lcd.h"
#include "adc.h"
#include "cmucam.h"
#include "motors.h"

extern volatile uint16_t delayCount;
extern volatile unsigned char RxBuffer[9];
volatile uint16_t ir_m, ir_l, ir_r;
volatile int sensors;

//interrupt vector for timer 0
SIGNAL(SIG_OUTPUT_COMPARE0)
{
  delayCount++;
}

//This function handles positioning the robot and picking up the golf balls
int pickup(void)
{
	uint16_t count;
	count = 0;

	move_Wrist(23275);		//lower wrist
	msDelay(1000);

	ir_m = adc_readn(0,1);
	while(ir_m <= 500  && count < 15000) {
		left_Motor(1,16000);
		right_Motor(1,16400);
		count = count + 1;
		ir_m = adc_readn(0,1);			//move forward until ball detected by
IR sensor
	}

	if(count == 15000)
		return 0;		//if count times out, pick-up not successful

	stop_Motors();
	msDelay(1000);
	move_Gripper(23000);			//close gripper
	msDelay(1000);
	move_Wrist(23350);			//pick up ball slightly off ground
	msDelay(1000);

	ir_m = adc_readn(0,5);			//check pickup
	if(ir_m < 400)
		return 0;				//unsuccessful
	else
		return 1;				//successful
}

//This function is used to drop the golf ball into the container
//once the robot is properly positioned in front
int dropoff(void)
{
/*	ir_r = adc_readn(4,5);
	ir_l = adc_readn(2,5);
	msDelay(1);
	while(ir_l <= 180 && ir_r <= 170) {
			right_Motor(1,14500);
			left_Motor(1,15000);
			ir_r = adc_readn(4,5);
			ir_l = adc_readn(2,5);
		}
*/
		stop_Motors();
		move_Wrist(23800);			//lift golf ball
		msDelay(1000);
		right_Motor(1,16000);
		left_Motor(1,15500);
		msDelay(1500);				//move forward
		stop_Motors();
		msDelay(1000);
		move_Gripper(23950);			//open gripper to drop ball

		return 0;
}

//This function checks the bump switches and photoreflectors
//(1 = white surface ; 0 = black surface)
int check_sensors(void)
{
		sensors = PINA;

		//stop running if bump switch pressed (collision)
		if(!(sensors & 0x04) || !(sensors & 0x08) || !(sensors & 0x10) || !(sensors & 0x20)
/*|| !(sensors & 0x80)*/) {
			stop_Motors();
			LCD_clear();
			LCD_string("Collision Detected\0");
			while(1) ;
		}

		//if left sensor detects border line detected, backup and turn away from line
		if(!(sensors & 0x01)) {
			stop_Motors();
			LCD_clear();
			LCD_string("Line Detected\0");

			right_Motor(0,15000);
			left_Motor(0,15000);
			msDelay(1000);
			stop_Motors();

			right_Motor(0,15000);
```

```
                left_Motor(1,15000);
                msDelay(1500);
                stop_Motors();

                return 1;
        }

        //if right sensor detects border line detected, backup and turn away from line
        if(!(sensors & 0x02)) {
                stop_Motors();
                LCD_clear();
                LCD_string("Line Detected\0");

                right_Motor(0,15000);
                left_Motor(0,15000);
                msDelay(1000);
                stop_Motors();

                right_Motor(1,15000);
                left_Motor(0,15000);
                msDelay(1000);
                stop_Motors();

                return 1;
        }

        return 0;
}

//check for obstacle detection to avoid collisions
int check_ir(void)
{
        ir_m = adc_readn(0,5);
        ir_r = adc_readn(4,5);
        ir_l = adc_readn(2,5);

        //if obstacle detected (less than 6 inches away), stop and turn away from obstacle
        if(ir_l > 300 || ir_r > 300) {
                LCD_clear();
                LCD_string("Obstacle Detected\0");

                if(ir_l > (ir_r + 10)) {
                        right_Motor(1,15000);
                        left_Motor(0,15000);
                }
                else {
```

```
				right_Motor(0,15000);
				left_Motor(1,15000);
			}

			while(ir_r > 200 && ir_l > 200) {
				ir_r = adc_readn(4,5);
				ir_l = adc_readn(2,5);
				check_sensors();
			}

			stop_Motors();

			return 1;
		}

		return 0;
}

//Use this function to call see if one of the four golf ball colors is in the current view
//(return 1 for red, 2 for blue, 3 for green, 4 for yellow, and 0 for none)
int findBall(void)
{
		unsigned int Rmx, Rmy, Rpixels, Rconfidence;
		unsigned int Bmx, Bmy, Bpixels, Bconfidence;
		unsigned int Gmx, Gmy, Gpixels, Gconfidence;
		unsigned int Ymx, Ymy, Ypixels, Yconfidence;

		trackRed();
		msDelay(1);
		Rmx = RxBuffer[1];
		Rmy = RxBuffer[2];
		Rpixels = RxBuffer[7];
		Rconfidence = RxBuffer[8];

		if((Rmx !=0 || Rmy != 0) && Rpixels <= 25 && Rconfidence >= 75)
			return 1;


		trackBlue();
		msDelay(1);
		Bmx = RxBuffer[1];
		Bmy = RxBuffer[2];
		Bpixels = RxBuffer[7];
		Bconfidence = RxBuffer[8];

		if((Bmx !=0 || Bmy != 0) && Bpixels <= 25 && Bconfidence >= 75)
```

```
			return 2;

		trackGreen();
		msDelay(1);
		Gmx = RxBuffer[1];
		Gmy = RxBuffer[2];
		Gpixels = RxBuffer[7];
		Gconfidence = RxBuffer[8];

		if((Gmx !=0 || Gmy != 0) && Gpixels <= 30 && Gconfidence >= 75)
			return 3;

		trackYellow();
		msDelay(1);
		Ymx = RxBuffer[1];
		Ymy = RxBuffer[2];
		Ypixels = RxBuffer[7];
		Yconfidence = RxBuffer[8];

		if((Ymx !=0 || Ymy != 0) && Ypixels <= 35 && Yconfidence >= 75)
			return 4;

		return 0;
}

//use this function to call trackColor()
void trackObject(int obj_color)
{
		if(obj_color == 1) {
			trackRed();
			msDelay(1);
		}
		else if(obj_color == 2) {
			trackBlue();
			msDelay(1);
		}
		else if(obj_color == 3) {
			trackGreen();
			msDelay(1);
		}
		else if(obj_color == 4) {
			trackYellow();
			msDelay(1);
		}
}
```

```
int main(void)
{
        unsigned char mx, my, pixels, confidence;
        unsigned int ball, color, pixelsize, boxconfidence, ballmy, boxmy;
        unsigned int count1, count2, temp1, temp2, dir, loop1;

        unsigned int rsize1, bsize1, gsize1, ysize1;
        unsigned int rconfidence2, bconfidence2, gconfidence2, yconfidence2;
        unsigned int rmy1, bmy1, gmy1, ymy1;
        unsigned int rmy2, bmy2, gmy2, ymy2;

        //initialize timers, motors, servos, camera, adc, and lcd

        sei();
        init_timer();
        LCD_init();
        motors_Init();
        servo_Init();
        adc_Init();

        DDRA = 0x00;
        PORTA = 0x00;

        UART1_init();
        CMUCam_init();
        UART_flush();

        msDelay(3000);

        //initialize variables

        ball = 0;

        //measured for calibration in NEB rotunda

        rsize1 = 25;
        bsize1 = 25;
        gsize1 = 30;
        ysize1 = 35;

        rconfidence2 = 75;
        bconfidence2 = 40;
        gconfidence2 = 40;
        yconfidence2 = 75;

        rmy1 = 120;
```

```
bmy1 = 120;
gmy1 = 120;
ymy1 = 120;

rmy2 = 85;
bmy2 = 90;
gmy2 = 85;
ymy2 = 70;

while(1) {
        while(ball == 0) {

                //determine if a ball is in the current view
                color = findBall();
                if(color == 1) {
                        pixelsize = rsize1;
                        boxconfidence = rconfidence2;
                        ballmy = rmy1;
                        boxmy = rmy2;
                }
                else if(color == 2) {
                        pixelsize = bsize1;
                        boxconfidence = bconfidence2;
                        ballmy = bmy1;
                        boxmy = bmy2;
                }
                else if(color == 3) {
                        pixelsize = gsize1;
                        boxconfidence = gconfidence2;
                        ballmy = gmy1;
                        boxmy = gmy2;
                }
                else if(color == 4) {
                        pixelsize = ysize1;
                        boxconfidence = yconfidence2;
                        ballmy = ymy1;
                        boxmy = ymy2;
                }
                else {
                        pixelsize = 0;
                        boxconfidence = 0;
                        ballmy = 0;
                        boxmy = 0;
                }

                //Display current tracked color on lcd
```

```
                        if(color != 0) {

                                if(color == 1) {
                                        LCD_clear();
                                        LCD_string("Tracking R Object\0");
                                }
                                else if(color == 2) {
                                        LCD_clear();
                                        LCD_string("Tracking B Object\0");
                                }
                                else if(color == 3) {
                                        LCD_clear();
                                        LCD_string("Tracking G Object\0");
                                }
                                else {
                                        LCD_clear();
                                        LCD_string("Tracking Y Object\0");
                                }

                                trackObject(color);
                                mx = RxBuffer[1];
                                my = RxBuffer[2];
                                pixels = RxBuffer[7];
                                confidence = RxBuffer[8];

                                //if ball detected enter loop
                                while((mx !=0 || my != 0) && pixels <= pixelsize &&
confidence >= 75) {

                                        trackObject(color);
                                        mx = RxBuffer[1];
                                        my = RxBuffer[2];
                                        pixels = RxBuffer[7];
                                        confidence = RxBuffer[8];

                                        //the following three loops are used to position the
golf ball
                                        //correctly for pick-up

                                        while(mx > 49 && pixels <= pixelsize &&
confidence >= 75) {
                                                left_Motor(0,16500);
                                                right_Motor(1,16500);

                                                trackObject(color);
                                                mx = RxBuffer[1];
                                                my = RxBuffer[2];
                                                pixels = RxBuffer[7];
                                                confidence = RxBuffer[8];
                                        }
                                        stop_Motors();

                                        while(mx < 47 && pixels <= pixelsize &&
confidence >= 75) {
                                                right_Motor(0,16500);
                                                left_Motor(1,16500);

                                                trackObject(color);
                                                mx = RxBuffer[1];
                                                my = RxBuffer[2];
                                                pixels = RxBuffer[7];
                                                confidence = RxBuffer[8];
                                        }
                                        stop_Motors();

                                        while(my < ballmy && mx >= 47 && mx <= 49
&& pixels <= pixelsize && confidence >= 75) {
                                                left_Motor(1,15500);
                                                right_Motor(1,15900);

                                                temp1 = check_sensors();
                                                temp2 = check_ir();

                                                trackObject(color);
                                                mx = RxBuffer[1];
                                                my = RxBuffer[2];
                                                pixels = RxBuffer[7];
                                                confidence = RxBuffer[8];
                                        }

                                        stop_Motors();

                                        //if the ball is positioned properly, enter this loop
                                        if(mx <= 49 && mx >= 47 && my >= ballmy) {
                                                temp2 = pickup();

                                                LCD_clear();
                                                LCD_string("Object Picked-Up\0");

                                                if(temp2 == 0) {
                                                    LCD_clear();
                                                    LCD_string("Pickup failed\0");
```

```
                                        left_Motor(0,16000);
                                        right_Motor(0,16000);
                                        move_Gripper(23950);
                                        msDelay(2000);
                                        stop_Motors();
                                        move_Wrist(23100);
                                        msDelay(3000);          //wait
for ball to settle
                                    }
                                    else
                                        ball = 1;
    //successful pick-up
                                }

                                //if ball is lost call trackColor() again to ensure a
false read was
                                //not the cause
                                if((mx ==0 && my == 0) || pixels > pixelsize ||
confidence < 75) {
                                    trackObject(color);
                                    mx = RxBuffer[1];
                                    my = RxBuffer[2];
                                    pixels = RxBuffer[7];
                                    confidence = RxBuffer[8];
                                }
                            }

                        }
                        //No ball detected
                        else {
                            LCD_clear();
                            LCD_string("Searching for Ball\0");

                            //turn in random direction for complete turn
                            if(TCNT0 & 0x01) {
                                right_Motor(0,16000);
                                left_Motor(1,16000);
                                dir = 0;
                            }
                            else {
                                right_Motor(1,16000);
                                left_Motor(0,16000);
                                dir= 1;
                            }
```

```
count1 = 0;
loop1 = 0;

while(loop1 == 0) {
        while(!(findBall()) && count1 < 26) {
                count1 = count1 + 1;
                temp1 = check_sensors();
        }

        stop_Motors();
        count2 = 0;

        //after turn, move forward for 2 feet if no ball found
        if(count1 == 26) {
                left_Motor(1,14600);
                right_Motor(1,15000);

                while((!findBall()) && count2 < 5) {
                        count2 = count2 + 1;

                        if(check_sensors() || check_ir())
                                count2 = 5;
                }

                loop1 = 1;
        }
        //if ball found, turn backward to confirm before
tracking
        else {
                temp1 = findBall();
                if(temp1 == 0) {
                        if(dir == 0) {
                                right_Motor(0,16000);
                                left_Motor(1,16000);
                        }
                        else {
                                right_Motor(1,16000);
                                left_Motor(0,16000);
                        }
                }
                else
                        loop1 = 1;
        }
}

stop_Motors();
```

```
				}
			}

			//find container to place golf ball the same way golf ball was found
			while(ball == 1) {
				trackObject(color);
				mx = RxBuffer[1];
				my = RxBuffer[2];
				pixels = RxBuffer[7];
				confidence = RxBuffer[8];

				//if container found enter this loop
				if((mx !=0 || my != 0) && pixels > pixelsize && confidence >=
boxconfidence) {

					LCD_clear();
					LCD_string("Tracking Box\0");

					//position container for drop-off

					while(mx > 49 && pixels > pixelsize && confidence >=
boxconfidence) {
						left_Motor(0,16000);
						right_Motor(1,16000);

						trackObject(color);
						mx = RxBuffer[1];
						my = RxBuffer[2];
						pixels = RxBuffer[7];
						confidence = RxBuffer[8];
					}
					stop_Motors();

					while(mx < 45 && pixels > pixelsize && confidence >=
boxconfidence) {
						right_Motor(0,16000);
						left_Motor(1,16000);

						trackObject(color);
						mx = RxBuffer[1];
						my = RxBuffer[2];
						pixels = RxBuffer[7];
						confidence = RxBuffer[8];
					}
					stop_Motors();
				while(my < boxmy && mx >= 45 && mx <= 49 &&
pixels > pixelsize && confidence >= boxconfidence) {
					left_Motor(1,15500);
					right_Motor(1,16000);

					temp1 = check_sensors();
					temp2 = check_ir();

					trackObject(color);
					mx = RxBuffer[1];
					my = RxBuffer[2];
					pixels = RxBuffer[7];
					confidence = RxBuffer[8];
				}

				stop_Motors();

				//container positioned correctly, call drop-off function
				if(mx <= 49 && mx >= 45 && my >= boxmy) {
					msDelay(1000);

					temp2 = dropoff();

					LCD_clear();
					LCD_string("Object Dropped\0");

					msDelay(1000);
					right_Motor(0,16000);
					left_Motor(0,16000);
					msDelay(1000);
					stop_Motors();
					move_Wrist(23100);

					ball = 0;
				}
			}
			//search for container
			else {
				LCD_clear();
				LCD_string("Searching for Box\0");

				//turn in random direction
				if(TCNT0 & 0x01) {
					right_Motor(0,15500);
					left_Motor(1,15500);
					dir = 0;
```

```
                }
                else {
                    right_Motor(1,15500);
                    left_Motor(0,15500);
                    dir = 1;
                }

                count1 = 0;

                while(((mx ==0 && my == 0) || pixels <= pixelsize ||
confidence < boxconfidence) && count1 < 60) {
                    trackObject(color);
                    mx = RxBuffer[1];
                    my = RxBuffer[2];
                    pixels = RxBuffer[7];
                    confidence = RxBuffer[8];
                    count1 = count1 + 1;

                    temp1 = check_sensors();
                }

                stop_Motors();
                count2 = 0;
                if(count1 == 60) {
                    left_Motor(1,14600);
                    right_Motor(1,15000);

                    while(((mx ==0 && my == 0) || pixels <= pixelsize
|| confidence < boxconfidence) && count2 < 20) {
                        trackObject(color);
                        mx = RxBuffer[1];
                        my = RxBuffer[2];
                        pixels = RxBuffer[7];
                        confidence = RxBuffer[8];
                        count2 = count2 + 1;

                        if(check_sensors() || check_ir())
                            count2 = 20;
                    }
                }
                stop_Motors();
            }
        }
    }
}
```