Final Written Report

Pyrobot (Of The Swampibbean)

Charles M Groom
TAs : Ryan Stevens
Tim Martin
Instructors: Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

Table of Contents

1. Abstract:

After much thinking and many other ideas and the fact that it would be very difficult to work with sound, I've came up with "Pyrobot (Of The Swampibbean)." The theme of the bot will be to drive around (not on water) looking for treasure(colorful objects). When it has found the objects, it will turn on it's flame thrower to signal. when in a louder environment it will go faster and have a better chance of getting the enemy.

The bot will incorporate: an IP cam for image tracking, sonar for object distance and object avoidance, sound sensor for sound level sensing. The image processing aspect is the most complicated system on the robot. The robot is a very interesting demonstration because it brings people having to yell at it, and also a flame thrower

2. Executive summary:

PyroBot is a robot shaped like an old ship that tracks and seeks Pirate Treasure (colorful objects). He will not move unless it hears sound, music, or yelling pirates. Once PyroBot has found the Treasure, it will make a signal with its on board flame thrower. If the Treasure is up side down, PyroBot toggles the flame indicating Pirates are around.

Pyrobot's special sensor is an IP cam with image processing made available with open cv's C++ libraries. For safety and the fact that I didn't want any false alarms I created a more complicated way of target tracking then just simple blob tracking. I made a pattern of colors: purple, green, and blue. I set them up as a 45º offset of each other. So when I wrote the Image tracking code I would look for all the colors then I would check and see if they were in the correct angle and order, then I would make sure they were the correct size. If all was true I would send the x-y credence wirelessly to the robot.

The other sensors are: (2) sonars for rang finding and object tracking/avoiding, (1)microphone for sound sensing, and bump switches for last measure object avoiding. The (2) sonars would pan back and forth 100º to create an array of data

points according to angle. This would give a more accurate position of an object. I thought it would be slow but it worked fine for me. The sound sensing was used to detect the intensity of the sound around the environment. The way I implemented it was doing a full wave rectification then doing a 10 point moving average. The result was used to create a linear offset of motor speed.
The Platform was (6) 2” x 12” x 24” stacked on top of each other to create a solid block of wood. The center was hallowed out and the outside was shaped to look like a sailboat. I put mast and sails on it to further more look like a ship. The sound sensor and the flame thrower was mounted to the masts. On the deck I made a place for the battery and some of the electronics to go. It was driven by two drive wheels and a passive caster.
The Flame thrower was made up of:
- Propane heating torch (flame)
- Marine Grade 12v propane safety solenoid (turn on/off)
- Camping stove (for regulator)
- Backpacking propane (light weight, small)
- pipe fittings and brazing (very hard to get everything together)
- My car’s ignition system (for sparks)

In conclusion this was a very good project. It was very time intensive and did have many hurtles but with enough time, research, and thinking it was taken care of. I’ve learned countless things, and would recommend somebody else to do a project like this. One thing I cannot stress enough: start early it will take some time to get everything working correctly. And learn what Open CV’s libraries are doing.

3. Introduction:

Since I Have a fondness with sailing and we live here in the swamp at UF, this bot will most interest me.

The bot will be built to look like an old schooner. It will have two driving wheels and one passive swiveling wheel. The construction of the bot will be wood with fire retardant paint. A flame thrower will be used as a signel, and will have panning sonars for self-awareness. This will be a complete write up on how I implemented and created this robot.

4. Integrated System:

The integrated system will consist of: a Micro controller, IP Cam, my computer for video processing with open CV, and XBee for communication.

5. Mobile Platform:

(The Bot) the platform of the bot will be made out of wood. It will look like a classic Schooner fig1. Put turned out more like a pirate ship.

Fig 1

I will be 24“ long and 12“ wide. It was (6) 2” x 12” x 24” stacked on top of each other, then carved out like a sailboat. It was going to be heavy so I made sure that everything was going to be strong, and used lightweight parts. it was made to be maneuverable. It will have two drive wheels and a passive caster hidden under the body. It will have sonar sensors mounted on it for navigation. The placement of the sonars is one on each side. On top of the deck a mic will be mounted to detect the sound level. I mounting the IP cam on top of the mast like the crow’s nest. I would like to have the fire jet mounted in the front of the boat. (Enemy ship) Optional with time give, but time was not given so I did make it. it will be an RC car with a metal ship on top of it. Otherwise a stationary ship will due. I’m going to have red LEDs on top of the mast so the other ship can find it. And will be painted red like our favorite Georgia team. It will have loose metal (“gold”) on top of it so the other ship can get it.

6. Actuation:

(The Bot) The propulsion will be done by two motors connected to two wheels with one passive caster for 360° movements. I would like to have a way to put the sails up when it starts and down when it's done. And I would like to have a Captain come on deck once it finds the enemy ship.

7. Sensors:

Sensors that I will be using are:

.

- IP cam (Main) with OpenCV will be used for finding the target.
- 2 Panning Sonars for obstacle avoidance and target ship proximity.
- Sound sensors to find the sound level of the room.
- Bump sensor for aft.

▪ IP cam (Main sensor)

The IP cam will be my main sensor (fig2). I'm going to use it to do blob detection of my target. The way I'm going to use it is stream a live video feed from my bot to my computer. On my computer it will do high level image processing, with the help of Open CV(open source video processing libraries). After my computer has done its video processing it will feed an x, y location of the blob. The information will be streamed back with an Xbee device. The video will be broadcasted over the network and the computer will connect to that.

The way my computer will do blob detection is it will convert the RGB color space to an HSV(*hue, saturation,* and *value)* color space (fig3). HSV space is easier then RGB space because each color is one hue value rather than a unique complicated combination of Red Green Blue. Then it will threshold the correct color (fig4 and fig5). Meaning only blue will show up, but it will show up as white. All else will be black. So the image will just be black and white, white being my threshold color. Fig4 is where I have blue tape on my head. Fig5 is my threshold value. The computer takes the largest blob and will send out an (x,y)location of the center.fig6. Algorithm in appendix.

Fig2

Fig3

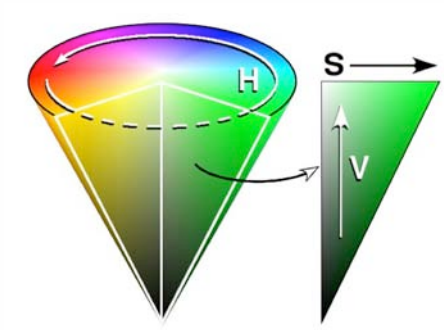


Fig4

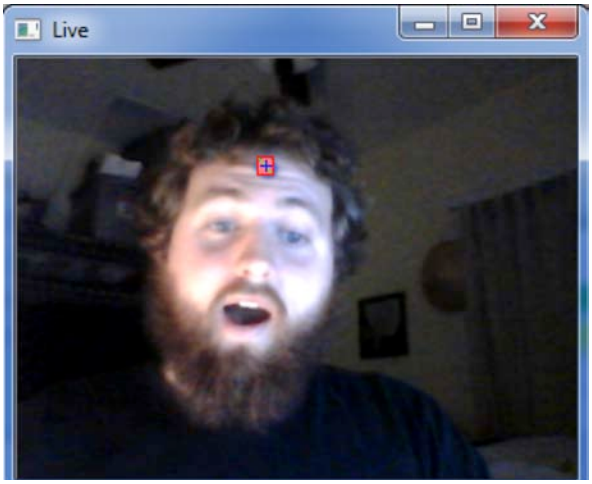


Fig5

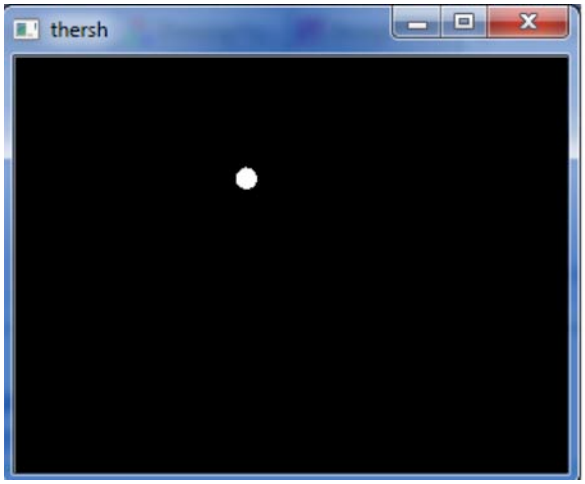
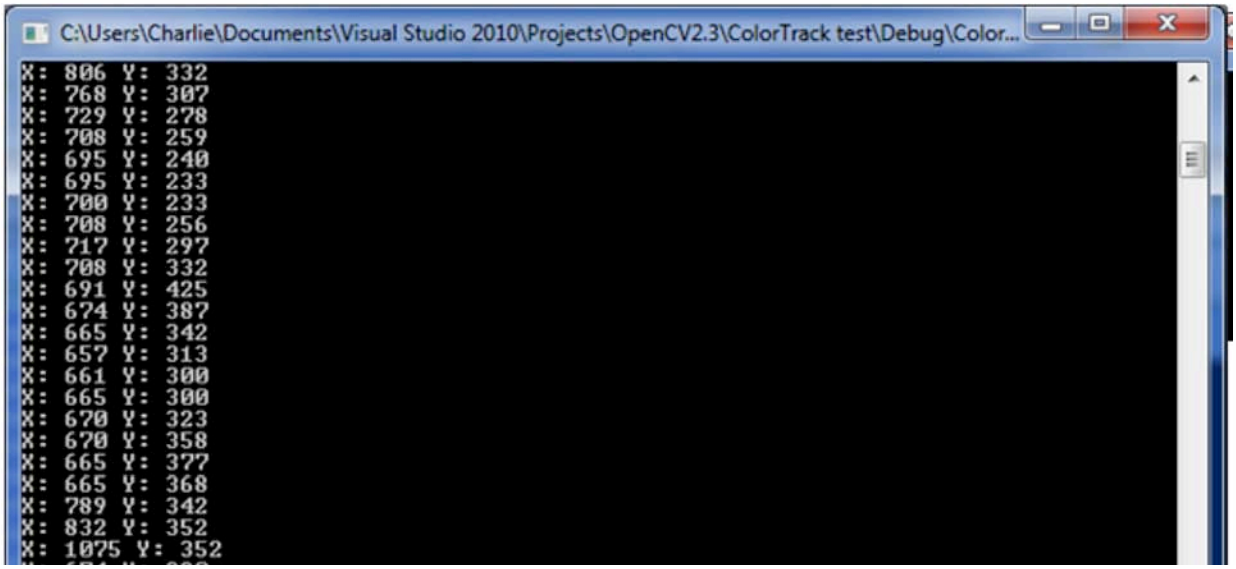


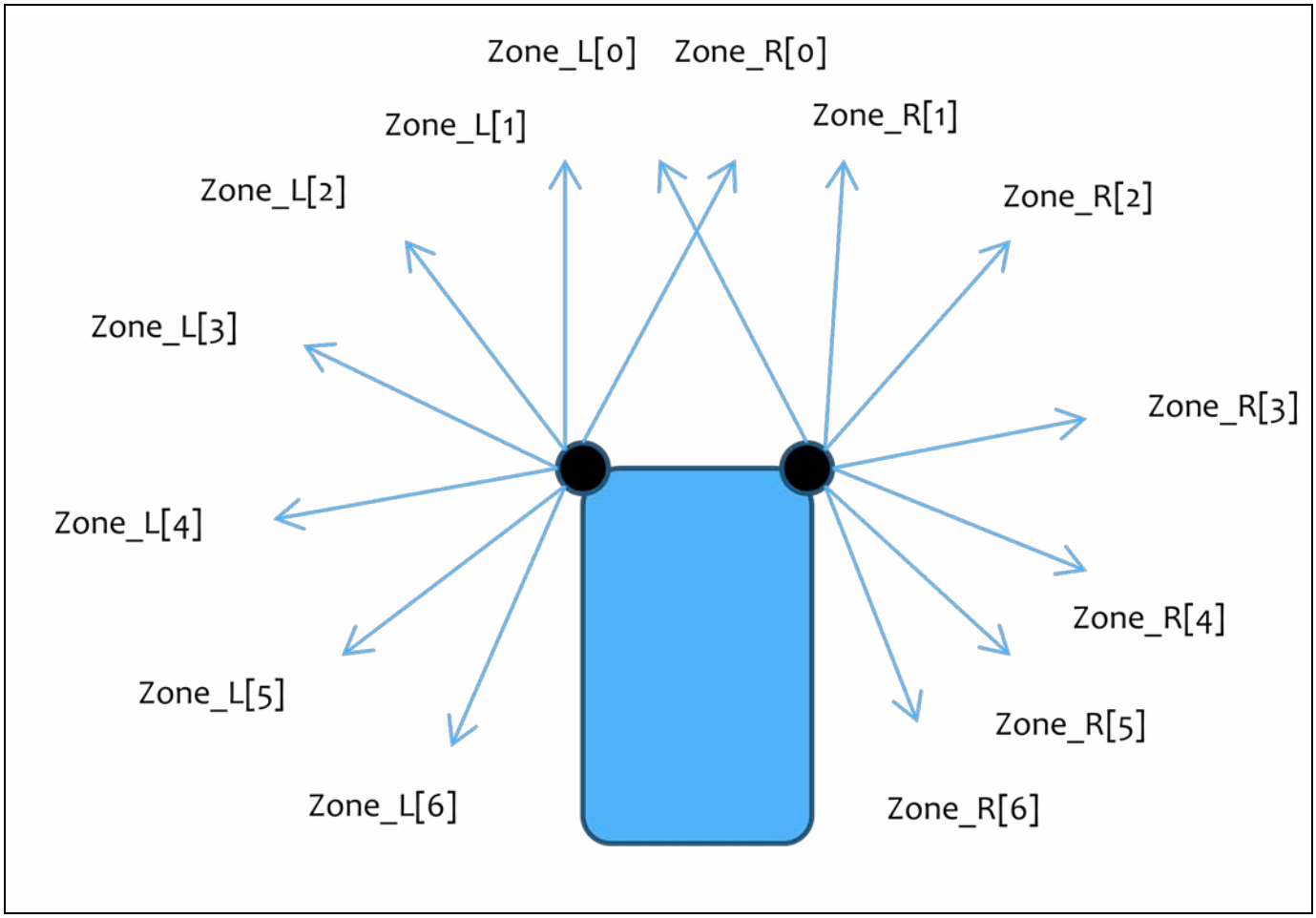
Fig6.



- **Sonars (panning)**

Another sensor will be sonars to do object avoid and seeking proximity. The location of them will be one on each side of the front, two total. The sonars will pan back and forth 180º with servos. Fig7 shows how I will implement the sonars.

I will have two different zones. Left_zone and Right_zone. When the sonars pans back and forth it will save the sonar distance value at the respected zone array value. It will take 50ms to get to each zone so I am driving the servos pretty fast. Each sonar will be 180º off each other so this system is pretty fast, if the servos don't explode. When doing object avoidance each zone will be update relatively fast, so this system should be almost real time values. It will take around 1sec to completely update all zones. Since they are 180º of each other the front will be checked twice a sec. It will be timer driven so I won't waste clock time on a delays meaning I can run other Algorithms in between.



- **Sound sensors**

I will be using a MEMs microphone to detect the sound levels. I'm implementing it by is doing a full wave rectification then a 10 point average. I use the result as a variable named "mic." It will be used to control the duty cycle of the PWM. I wanted a nice linear increase of PWM to mic level. So I fit a line, with our friend y-y1=m(x-x1) and m = $\frac{\Delta y}{\Delta x}$ and put it in the form y=mx+b.

The motor_DrivePWM= (mic*1.9)-2490;

The equation gave around a 50% duty cycle with ambient sound and 100% duty cycle at full sound. The "mic" value updates around 15ms so it is almost instant. Code in appendix.

- **Bump sensors**

This sensor will be by far the simplest sensor I have. It is just a simple switch if the switch is open it has not been bumped. If it is closed it has been bumped. It will be on a port pin with a pull up resister. Simple.

8. Behaviors:

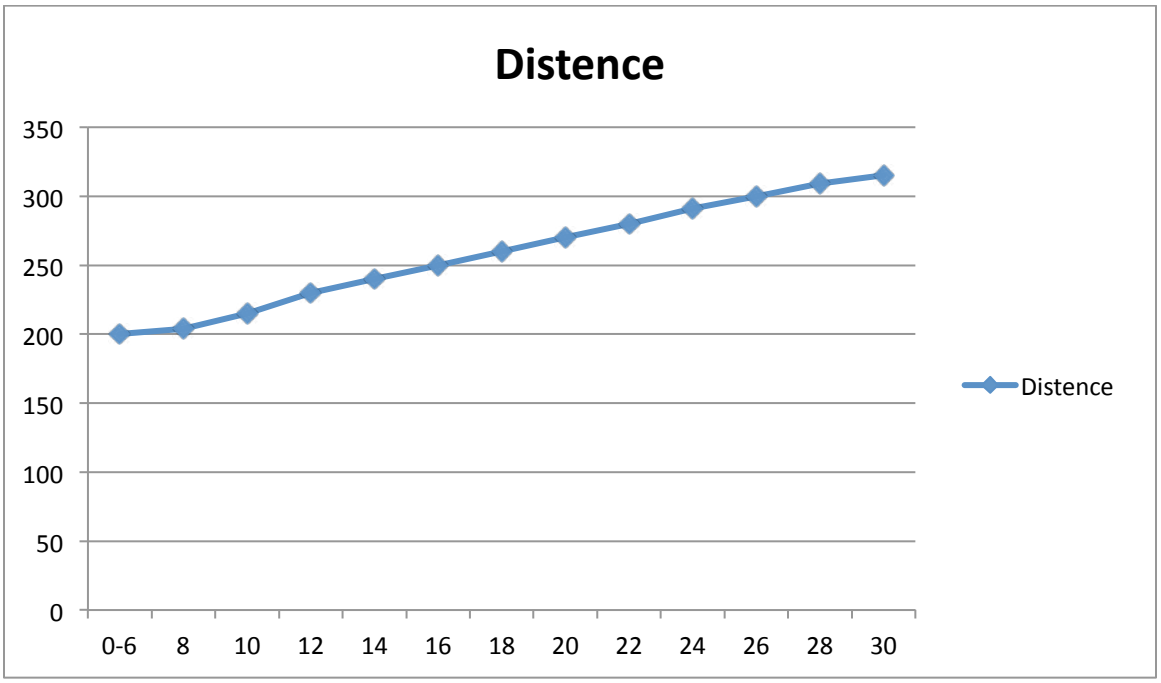
The Bot ship will drive around until it finds the target. It will turn on the flame thrower. The sails will go down and the Capt. will start dancing on deck. I would like a way to play a pirate song as well. It will disable the enemy ship with fire. It will steal it's gold by moving a magnet across the enemy ship's deck. The enemy ship if time is not limited will be driven around by someone else. They will be trying to out maneuver the bot. which hopefully will not be the case. If time does run out the enemy ship with be a stationary object that the bot has to find. Two machines might be pushing it. Time did run out so I used a color target This is going to be a lot of systems to put together and I hope I am not over doing myself. After the fact it was not so bad I would recommend anyone to do this

9. Experimental Layout and Results

a) Sonar sensor

Code for testing sonar is an appendix. It is basically read from ADC and print to LCD screen. I took values at different distances. I made a table and a graph of my results

Distance (inches)	value
0-6 same	200
8	204
10	215
12	230
14	240
16	250
18	260
20	270
22	280
24	291
26	300
28	309
30	315


Distence
350
300
250
200
150
100
50
0
0-6
8
10
12
14
16
18
20
22
24
26
28
30
Distence

b) sound sensor

Code for getting sound levels in appendix. Note: sound levels will vary with what your definition of low talk, High talk, scream is. Basically what this shows is the mic will be between 1600 and 2100. So I wrote my code with these results.

Sound level	value
ambient	1605
Low talk	1698
High talk	1809
scream	2054

10. Conclusion

This went rather smoothly but I still need to get the xbee working and send those x,y value to the bot. I tried for a while to get the blue tooth stuff working but the time that you put into the Bluetooth system you might of just gone with the xbee in the first place. I need to put the servos on for the panning of the sonars. I need to find a safe way to make a safe flame thrower. Then I will mostly be done. Now that I am done one thing that I would recommend is start early, I did get my robot to work but if I didn't start early there is no way that I would have gotten it done.

11. Documentation

http://8a52labs.wordpress.com/2011/05/24/detecting-blobs-using-cvblobs-library/

12. Appendices

a) OpenCV code

```
//OpenCV Headers
#include<cv.h>
#include<highgui.h>
//Input-Output
#include<stdio.h>
//Blob Library Headers
#include<cvblob.h>
//Definitions
#define h 240
#define w 320

//NameSpaces
using namespace cvb;
using namespace std;
int high;
int low;
int hue = 110 ;
int s=100;
int v=100;
char c;
int main()
{
//Structure to get feed from CAM
CvCapture* capture=cvCreateCameraCapture(0);
//Structure to hold blobs
CvBlobs blobs;
//Windows
cvNamedWindow("Live",CV_WINDOW_AUTOSIZE);
//Image Variables
IplImage *frame=cvCreateImage(cvSize(w,h),8,3); //Original Image
```

```
IplImage *hsvframe=cvCreateImage(cvSize(w,h),8,3);//Image in HSV color space
IplImage *labelImg=cvCreateImage(cvSize(w,h),IPL_DEPTH_LABEL,1);//Image Variable for
blobs
IplImage *threshy=cvCreateImage(cvSize(w,h),8,1); //Threshold image of yellow color

//Getting the screen information
int screenx = GetSystemMetrics(SM_CXSCREEN);
int screeny = GetSystemMetrics(SM_CYSCREEN);

while(1)
{
//Getting the current frame
IplImage *fram=cvQueryFrame(capture);
//If failed to get break the loop
if(!fram)
break;
//Resizing the capture
cvResize(fram,frame,CV_INTER_LINEAR );
//Flipping the frame
cvFlip(frame,frame,1);
//Changing the color space
cvCvtColor(frame,hsvframe,CV_BGR2HSV);




//Thresholding the frame for yellow
//hue=107;
high = hue+5;
low = hue-5;
cvInRangeS(hsvframe,cvScalar(low,s,v),cvScalar(high,255,255),threshy);
//cout<<"hue: "<<hue<<endl;
//cout<<"S: "<<s<<endl;
//cout<<"V: "<<v<<endl;

/*
opencv ranges seem to be
h 0 - 180
s 0 - 255
v 0 - 255

gimp ranges are:
h 0 - 360
s 0 - 100
v 0 - 100
*/


//Filtering the frame
cvSmooth(threshy,threshy,CV_MEDIAN,7,7);
//Finding the blobs
unsigned int result=cvLabel(threshy,labelImg,blobs);
//Rendering the blobs
cvRenderBlobs(labelImg,blobs,frame,frame);
//Filtering the blobs
cvFilterByArea(blobs,60,500);
for (CvBlobs::const_iterator it=blobs.begin(); it!=blobs.end(); ++it)
{
```

```
double moment10 = it->second->m10;
double moment01 = it->second->m01;
double area = it->second->area;
//Variable for holding position
int x1;
int y1;
//Calculating the current position
x1 = moment10/area;
y1 = moment01/area;
//Mapping to the screen coordinates
int x=(int)(x1*screenx/w);
int y=(int)(y1*screeny/h);
//Printing the position information
cout<<"X: "<<x<<" Y: "<<y<<endl;
}
//Showing the images
cvShowImage("Live",frame);
cvShowImage("thersh",threshy);
//Escape Sequence
c=cvWaitKey(33);
if(c=='q')
break;
else if(c=='v'){
       v--;
              }
else if(c=='V'){
       v++;
              }

else if(c=='s'){
       s--;
              }
else if(c=='S'){
       s++;
              }

else if(c=='h'){
       hue--;
              }
else if(c=='H'){
       hue++;
              }
else;
}
//Cleanup
cvReleaseCapture(&capture);
cvDestroyAllWindows();

}
```

b) MCU code

```
/** Charlie Groom
 * IMDL
 * Piro Bot code
 *
```

```
 *
 *
 */

/*
 * Include header files for all drivers that have been imported from
 * AVR Software Framework (ASF).
 */
#include <asf.h>
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "picServo.h"
#include "ADC.h"
#include "sonar.h"

#define DbLedOn()           (PORTR.OUTCLR = 0x02)             //Turns the debug led on.
The led is connected with inverted logic
#define DbLedOff()          (PORTR.OUTSET = 0x02)             //Turns the debug led off.
The led is connected with inverted logic
#define DbLedToggle()       (PORTR.OUTTGL = 0x02)             //Toggles the debug led
off.  The led is connected with inverted logic


void O_avoid(void);
void get_adc(void);
void setupBlueToothConnection(void);


int sonar_L, sonar_R, sonar_B;     // sonar globels
int mic;                                          // mic global


int main (void)
{
       board_init(); /*This function originates in the file init.c, and is used to
initialize the Epiphany DIY
                                    motorInit() is declared within because by default you
the user should define what your
                                    motor setup is to prevent hurting the Epiphany.  You
can do this by
                                   */
       RTC_DelayInit();//initializes the Real time clock this seems to actually take an
appreciable amount of time
       DbLedOn();    //I like to do this by default to show the board is no longer
suspended in the bootloader.
       servoControlInit();//initializes the servo module ***including enabling global
interrupts*** required for the servo control module
       uartInit(&USARTC0,57600);//as can be seen in the schematic.  This uart is
connected to the USB port.  This function initializes this uart
       //stdout = &USB_str;//This function points stdout to the USB device.  So this
means printf will send data out the USB port.
```

```
        //stdout = &lcd_str;
        ADCsInits();//this function initializes the ADCs inside the Xmega
        LCDInit();
        //you should add any further initializations here
        sei();

        RTC_Delay_ms(100);


        uartInit(&USARTE1,38400);// xbee




        /*****************************
        *
        *STARTING MY CODE!!!!!!!!!
        *
        *****************************/


int i;
int p;



setupBlueToothConnection();







        while (1){

        //printf("\r\n+INQ=1\r\n"); // start inquiring
        for (i=0; i<4; i++ )
        {
        p=i*45;
        setServoAngle(p,1);
        _delay_ms(100);
        get_adc();

        O_avoid();

        _delay_ms(10);

        stdout = &lcd_str;
        printf("MIC = %d \n S = %d \r", mic, sonar_B );


        }//for

               for (i=4; i>1; i-- )
        {
        p=i*45;
```

```
        setServoAngle(p,1);
        _delay_ms(100);
        get_adc();

        O_avoid();

        _delay_ms(10);

        stdout = &lcd_str;
        printf("MIC = %d \n S = %d \r", mic, sonar_B );


        }//for

        //_delay_ms(150);


        }//while




        }//main









/************************
*
*Functions
*
************************/




/* Charlie Groom IMDL 10/15/2011
/ get_adc Ver 1.0
/
/ Code to get values from ADC
/ Global Ver: sonar_L , sonar_R , sonar_B , mic , bump
/ mic takes a 12 Pt. ave of mic does full wave rec.
*/

void get_adc(void)
{
        int i;
        int mic_temp1 = 0;   // to hold value while aveing
        int mic_temp2 = 0;

        // geting sonar val ** dont have all sonar yet **
        ADCA_request(1,1);
```

```
	sonar_B = ADCA_getVal(1);

	// geting Mic value 12 pt ave

	for (i=0; i<12; i++)
	{
		_delay_ms(1);		// need delay for some werid reason??

		ADCA_request(0,0);   // getting val
		mic_temp1 = ADCA_getVal(0);


		if(mic_temp1 > 1585)              // starting full wave!
		{
			mic_temp2=mic_temp2+mic_temp1; // high part of wave
		}


		else
		{
			mic_temp1=1585+(1585-mic_temp1); // low part of wave
			mic_temp2=mic_temp2+mic_temp1;
		}

	} // for

	//saveing into mic
	mic = mic_temp2/12;


}// END ADC GET STUFF




/* Charlie Groom IMDL 10/15/2011
/ O_avoid Ver 1.0
/
/ Code to do Object avoid
/ need to get more sonar
/ basic go strate if something in way turn till nothing then go strate againe
*/

void O_avoid(void)
{
	int pwm;

	if(sonar_B > 210){

		if (mic<1600){
			setMotorDuty(1,500,MOTOR_DIR_FORWARD_gc);
			asm("nop");

			setMotorDuty(3,500,MOTOR_DIR_FORWARD_gc);
			asm("nop");
		}
```

```
		else if(mic>1850){
			setMotorDuty(1,1023,MOTOR_DIR_FORWARD_gc);
			asm("nop");

			setMotorDuty(3,1023,MOTOR_DIR_FORWARD_gc);
			asm("nop");
		}

		else{
			pwm=(1.9*mic)-2490;
			setMotorDuty(1,pwm,MOTOR_DIR_FORWARD_gc);
			asm("nop");

			setMotorDuty(3,pwm,MOTOR_DIR_FORWARD_gc);
			asm("nop");
		}
	}// big if


	else{ // something in the way!!


		if (mic<1600){
			setMotorDuty(1,500,MOTOR_DIR_BACKWARD_gc);
			asm("nop");

			setMotorDuty(3,500,MOTOR_DIR_FORWARD_gc);
			asm("nop");
		}

		else if(mic>1850){
			setMotorDuty(1,1023,MOTOR_DIR_BACKWARD_gc);
			asm("nop");

			setMotorDuty(3,1023,MOTOR_DIR_FORWARD_gc);
			asm("nop");
		}

		else{
			pwm=(1.9*mic)-2490;
			setMotorDuty(1,pwm,MOTOR_DIR_BACKWARD_gc);
			asm("nop");

			setMotorDuty(3,pwm,MOTOR_DIR_FORWARD_gc);
			asm("nop");
		}
	}// Big else

}//END O_avoid



void setupBlueToothConnection(void)
{
```

```
      stdout = &uartE1_str;
    _delay_ms(1000);

      printf("\r\n+STWMOD=0\r\n");// set as slave
      _delay_ms(3000);
      printf("\r\n+STNA=modem\r\n");//name
      _delay_ms(3000);
      printf("\r\n+STAUTO=0\r\n");// auito connect
      _delay_ms(3000);
      printf("\r\n+STOAUT=1\r\n");// auto connect
      _delay_ms(3000);
      printf("\r\n+STPIN=0000\r\n");//pass word
      _delay_ms(2000);
      printf("\r\n+INQ=0\r\n"); // start inquiring
      _delay_ms(2000);

      //printf("\r\n+CONN=00,1f,81,00,08,30\r\n"); // connect to my computer


}//END bluetooth setup
```