

LC-113: Laser Chasing Robot

*By
Darsan Patel*



*University of Florida
EEL5666:
Intelligent Machine Design Lab*

*Instructors:
Dr. Arroyo
Dr. Schwartz*

*TA's:
Mike Pridgen
Thomas Vermeer*

Table of Contents

| | |
|------------------------------------------------|-------------------------------------|
| Abstract | 4 |
| Introduction | 5 |
| Flow Chart..... | 6 |
| Mobile Platform..... | 7 |
| Actuation..... | 8 |
| Servo Motor 1 | 8 |
| Servo Motor 2 | 8 |
| Motor Driver..... | 8 |
| Connectivity..... | 8 |
| Electric Motor | 8 |
| Sensors..... | 9 |
| Ultrasonic Range Finder – Maxbotix LV-EZ3..... | 9 |
| Analog Signal from the Sonar Sensor | 9 |
| Cleaner Data | Error! Bookmark not defined. |
| Infrared Proximity Sensor – Sharp..... | Error! Bookmark not defined. |
| Analog Signal from the IR Sensors..... | Error! Bookmark not defined. |
| Cleaner Data | Error! Bookmark not defined. |
| Special Sensor..... | 11 |
| Linksys WVC80N Wireless Camera..... | 11 |
| Xbee Module..... | 11 |
| Behaviors..... | 12 |
| Default On..... | 12 |
| Idle | 12 |
| Image Processing | 12 |
| Wireless Signal..... | 12 |
| Steering..... | 12 |
| Speed Control..... | 12 |
| Experimental Layout and Results..... | 13 |

| | |
|-------------------------|----|
| Digital Filter..... | 13 |
| Conclusion..... | 15 |
| Appendix..... | 16 |
| Obstacle Avoidance..... | 16 |
| Image Processing..... | 18 |
| Documentation..... | 25 |
| Resources..... | 25 |
| Special Thanks..... | 25 |

Abstract

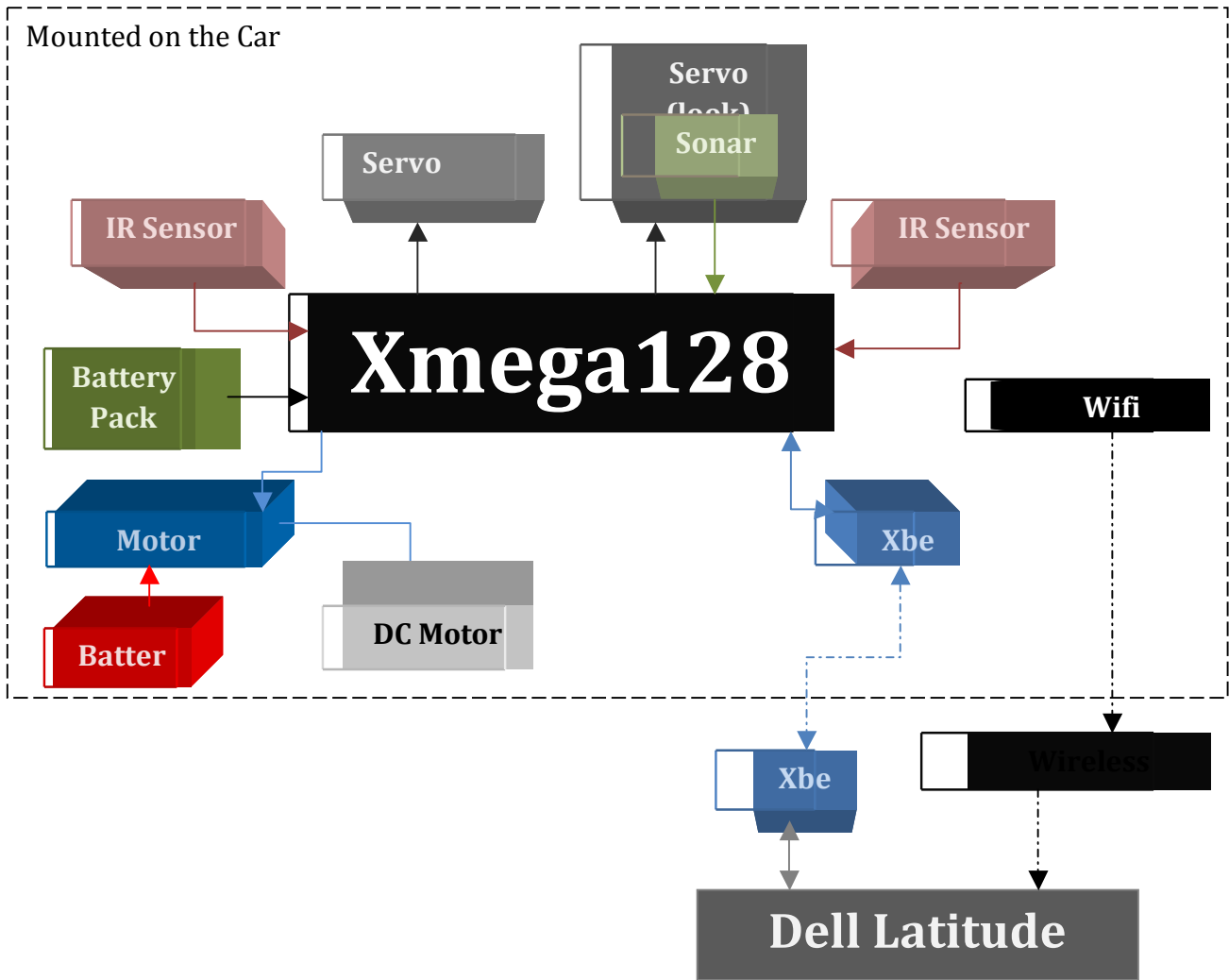
This paper will discuss the design and building process of LC 113, a laser chasing robot. The goal of this project is to have LC 113 be controlled by a laser pointer. This robot will use some computer vision to detect the laser pointing to the ground. LC 113 will also avoid obstacles along its way, using the data from two IR sensors and one sonar sensor.

Introduction

As most kids, my father bought me a RC car when I was young. Unlike most kid, my father bought me an unassembled RC car. As it took me couple days to assemble this car, it became my favorite toy as going up. It took me couple of weeks to get used to driving this car, since the orientation of the car changed as the car was driven. If I turned right, as the car was moving away from me, the car would go right from my point of view as mentally as expected. If I turned right, as the car was coming towards me, the car would go left from my point of view. The continuous change in orientation made the car difficult to control, that meant, I got in to a lot of crashes.

Controlling the car with the laser pointer is a good alternative to solve the changing orientation problem. To control the car, one would simply point the laser on to the ground. The car will automatically control the speed and the path it needs to take, according to where the laser is pointed. The sensors will feed knowledge about the environment around the car, so the car can avoid crashing into any obstacles around the car.

Flow Chart



Mobile Platform

The main platform of LC-113 is an RC car. The RC car is all-wheel drive with Ackermann steering system. The Ackermann steering system is controlled by a servo. The drive is controlled by the motor driver which controls an electronic motor. The platform also contains sensors to obtain information of the environment.

Actuation

Servo Motor 1

This servo motor controls the Ackermann steering system. The control for this servo is linearly proportionate to the x value sent by the computer, representing laser pointer horizontal position respect to the camera.

Servo Motor 2

This servo motor pans the sonar sensor. This servo motor is set to identically follow the steering. As the car is turning the servo will make the sonar point in the same direction so it knows what's up ahead.

Motor Driver

Motor driver receives signal from the board and sends the electric motor the appropriate power.

Connectivity

Motor driver came with three connecting wires for the board: input signal, Vcc, and ground. Motor driver is also connected to 7.2 volts battery pack that supplied power to the motor. This 7.2 volts battery when turned off, bumped power back in to the board from the Vcc wire connection to the board. Vcc was disconnected and everything worked smoothly from then on.

Electric Motor

Motor drives the car. The motor pulls in 2 amps at pick load. The motor driver was selected accordingly.

Sensors

The robot platform contains the sensors to obtain information about the environment in which the robot is in. There are two types of sensors being used in this robot platform: Sonar sensor (Ultrasonic Range Finder), and IR sensor (Infrared Proximity Sensor).

Ultrasonic Range Finder – Maxbotix LV-EZ3



Sonar sensor can detect obstacles by sending a pulse of sound waves and receiving the same signal once it's reflected from the obstacle. This sensor will detect obstacles in front of the car as much as 5 feet away from it. This particular sensor will be on a servo, so it can pan for obstacles while the car turns.

Analog Signal from the Sonar Sensor

The output of the sonar sensor was measured with different distances.

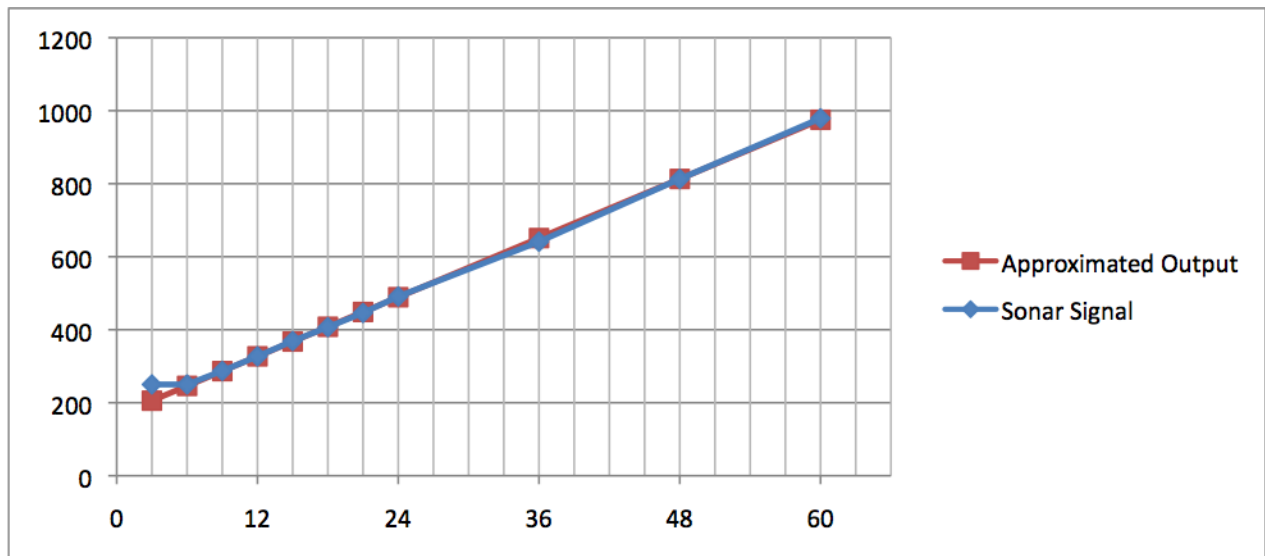


Figure 1: Compares the measured values from the sonar sensor with estimated values using the equation used in the program.

7a. This worked for the most part but still needed improvement in the data output.

Analog filter was used to clean up the data by simply connecting $.1\mu\text{F}$ capacitor from the signal input to ground. This slowed the reaction time of the sensor but gave more accurate

results. Analog filter was chosen over digital filter because analog filter would not affect the computer processing time.

Special Sensor

There are two different devices used to finish one process as the special Sensor: Wireless camera, wireless module.

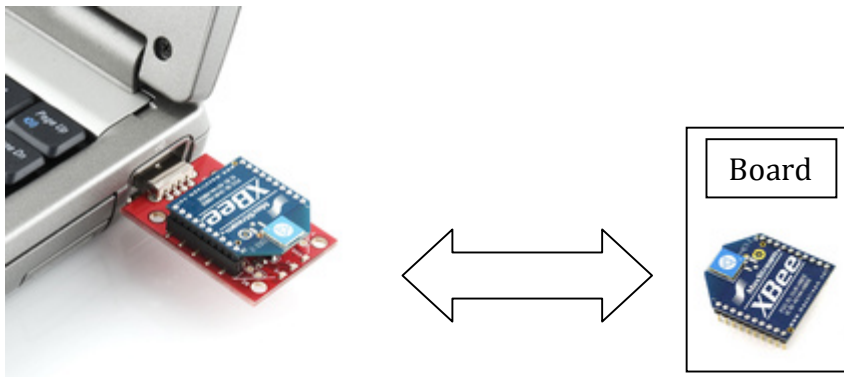
Linksys WVC80N Wireless Camera



This wireless camera sends images to a remote computer, where images can be processed, through wifi network.

This camera was chosen, because it had high resolution capability for spotting small red laser point, high frames per second for up to date data and wireless capability so the image can be processed on a remote computer.

Xbee Module



Xbee module is used to send information from computer to the robot wirelessly, by serial communication.

On the board: Power is pulled from the power bus and Tx and Rx is connected to the USART on port F (pin 2 and 3).

On computer: Com port 4 was used.

Behaviors

Default On

The IR and sonar sensor obtains data continuously.

Camera sends images to the computer continuously.

Computer sends information back to the robot, through xbee module, continuously.

If no laser is found the computer will send back 'x160.'

Idle

At idle, with no laser detection, the computer will transmit 'x160.' This will make the speed=0, steering angle=0, view angle=0 for the robot.

Image Processing

Using openCV library for image processing, different color were filtered out and color red was amplified. Everything was turned into black and the image of the laser point was turned in to white. The location of the center of laser was found.

Wireless Signal

Packet of information of 5 bits is made. First bit indicates the start 'x' and next four bit is the value of the x position calculated from the image processing.

Steering

The steering is linearly proportionate to the x position sent from the computer. The sonar sensor servo will follow the steering identically.

If the right IR sensor senses something close by, the robot will steer hard left and vice versa.

Speed Control

The speed of the robot is linearly proportionate to the data from the sonar sensor. Speed will decrease as obstacles get closer to the robot.

Experimental Layout and Results

Digital Filter

As the signal is obtained with an analog output from the sensors, it seems too noisy. To fix this problem couple of filter was kept in mind. Filtering the data digitally seemed to be the most practical plan for fast accurate results.

- Take a set of n data points
- Take the highest and the lowest values out
- Take the average of the rest of the data points
- Delete the oldest value of data and replace it with the new on

Pros

- Cleaner Data

Cons

- Slow Response
- Slowed down processing

Although Digital Filter was a good option, it was not used because it tends to slow down processing time.

Conclusion

LC113 can send image to the computer. The computer can process the image, find the laser point and come up with a coordinate system. The robot can control the speed with the use of sonar sensor. The robot can also steer away from obstacle using the IR sensors. In short the LC113 does have autonomous capabilities; however, LC113 fell short to make its goal. Major issue with the robot is that it couldn't send information back to the robot so steering can't be controlled.

Future work for this robot is to control it remotely from the computer. Also have complex behaviors such as going through a circuit as fast as possible.

In conclusion, after a lot of hard work and many hours of frustrations in programming this robot, LC113 fell just short of its goal.

Appendix

Obstacle Avoidance

```
#include <avr/io.h>

#include "PVR.h"

#include "uart.h"

#include "print.h"

void main(void)
{
    xmegaInit();           //setup XMega
    delayInit();          //setup delay functions
    ServoCInit();         //setup PORTC Servos
    ServoDInit();         //setup PORTD Servos
    ADCAInit();           //setup PORTA analog readings
    lcdInit();            //setup LCD on PORTK
    lcdString("LC113"); //display "PV Robotics" on top line (Line 0) of LCD
    lcdGoto(1,0);         //move LCD cursor to the second line (Line 1) of
LCD
    lcdString("Obstacle Avoidance"); //display "Board Demo" on second line
    PORTQ_DIR |= 0x01;    //set Q0 (LED) as output
    int i = -30;
    PORTJ_DIR = 0b11111110; //set PORTJ bit 0 to input, 1-7 to output
    initUSARTF0(9600);    //setup Serial Communication
    while(1){
        int leftDis = ADCA1();
        int rightDis = ADCA0();
```



```

int frontDis = ADCA7();

ServoD5(0);           //sonar sensor
ServoD4(0);           //steering
if(frontDis>651){
    i=-10;             //Cap speed
}
else{
    i=-10*frontDis/324-40; // speed
}
ServoD0(i);           //motor driver
if (frontDis < 300){
    ServoD0(-50);
}
if (leftDis>2500){
    ServoD4(60);
    ServoD5(-60);
}
if (rightDis>2500){
    ServoD4(-60);
    ServoD5(60);
}
delay_ms(100);       //delay
100ms
}
}

```

Image Processing

```
//Standard Header Files
#include <iostream>
#include <string>

//OpenCV Header Files
/**
 * Environment Variable OPENCV_2_0 points to OpenCV directory e.g.
 C:\OpenCV\
 * Note: OpenCV version 2.0
 */

#include "highgui.h"           //Linker lib - highgui210.lib
#include "cv.h"                 //Linker lib - cv210.lib
#include "cvaux.h"             //Linker lib - cvaux210.lib
#include "cxcore.h"           //Linker lib - cxcore210.lib
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include "portLib.h"
//#include "portLib.c"

//#define PORT4 0x2E8

int main()
{

    cvNamedWindow("WEB_CAMERA",CV_WINDOW_AUTOSIZE);
    cvNamedWindow("Red - Green",CV_WINDOW_AUTOSIZE);
    cvNamedWindow("Red - Blue",CV_WINDOW_AUTOSIZE);

    CvCapture *webcam = 0;

    //webcam = cvCaptureFromCAM(0);
    webcam = cvCreateFileCapture("http://10.0.0.2/img/video.mjpeg");

    int frameDelay= 1;
    IplImage* testFrame = cvQueryFrame(webcam);
    int width = testFrame->width;
    int height = testFrame->height;
    int fps = 15;
    int numFrames = 10000000000;

    int noisex = 5;
    int noisey = 5;

    int sample = 4;

    int value2=0;

    int x[320];
    int xfilter[4];
    int c=0;
```

```

int y[240];
int yfilter[4];

int i, j;

int isColor = 1;

    IplImage *tempImg,
    *blurred,
    *redGray,
    *greenGray,
    *blueGray,
    *red,
    *green,
    *blue,
    *threshR,
    *threshR1,
    *threshG,
    *threshB;

    blurred = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 3 );
    tempImg = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 3 );
    red = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 3 );
    blue = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 3 );
    green = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 3 );

    redGray = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 1 );
    blueGray = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 1 );
    greenGray = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 1 );
    threshR = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 1 );
    threshR1 = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 1 );
    threshG = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 1 );
    threshB = cvCreateImage(
cvSize(width,height),IPL_DEPTH_8U, 1 );

    printf("LC-113: Laser Chasing Robot\n");
    printf("University of Florida\n");
    printf("EEL5666: Intelligent Machine Design Lab\n\n\n");

    printf("Width:                %d\n", width);
    printf("Height:                   %d\n", height);

```

```

// printf("Width: %4d", width);
printf("Frames per sec:           %d\n", fps);

if(!webcam)
{
    printf("Cannot open webcam\n");
    return 0;
}
int key = 0;

int port = 4;
int baud = 9600;
HANDLE p1 = portOpen(port,baud);

while(key!='q')
{
    CvScalar    value1;

    tempImg = cvQueryFrame(webcam);

    //Lets blur the image using Gaussian Blurring
    cvSmooth(tempImg,blurred,CV_GAUSSIAN,3,3,3,0);

    //Lets split the image into its constituent channels
    cvSplit(tempImg,blueGray,greenGray,redGray,0);

    cvSub(redGray, greenGray, greenGray,0);
    cvSub(redGray, blueGray, blueGray, 0);

    //Lets Take a Threshold for each channel!
    cvThreshold(blueGray,threshB,40,255,0);
    cvThreshold(greenGray,threshG,30,255,0);
    cvThreshold(redGray,threshR,240,255,0);
    cvThreshold(blueGray,threshR1,240,255,0);

    i=0;
    j=0;
    value2 = 0;

    while (i<height){
        j=0;
        while (j<width){
            value1 = cvGet2D(threshG,i,j); // get the (i,j)
            pixel value

            if (value1.val[0] != 0){
                value2 = 1 + value2;
            }
            j=j+1;
        }
        y[i] = value2;
        i=i+1;
        value2=0;
    }
    i=0;
}

```

```

j=0;

//Total Y1
int y1 = 0;

while (i<height){
    y1 = y1 + y[i];
    i = i+1;
}
int y2 = 0;

i=0;

while (i<height){
    if (y1>noisy){
        y2 = y2 + i*y[i]*100/y1;
    }
    else{
        y2 = height*100;
    }
    i = i + 1;
}
y2 = y2/100;

//Detecting X
i = 0;
j = 0;
value2 = 0;

while (j<width){
    i = 0;
    while (i<height){
        value1 = cvGet2D(threshG,i,j); // get the (i,j)
        pixel value

        if (value1.val[0] != 0){
            value2 = 1 + value2;
        }
        i = i + 1;
    }
    x[j] = value2;
    j = j + 1;
    value2 = 0;
}
i=0;
j=0;

//Total X1
int x1 = 0;

while (j < width){
    x1 = x1 + x[j];
    j = j + 1;
}
int x2 = 0;

j = 0;
while (j < width){

```

```

        if (x1 > noisex){
            x2 = x2 + j * x[j] * 100/x1;
        }
        else{
            x2 = width*100/2;
        }
        j = j + 1;
    }
    x2 = x2/100; // x position of the laser

/* Serial Communication *
//      x2 = 688;
//      x2 = rand();
//      y2 = 11;
//      unsigned char xx2;

    unsigned char xx2[5]; //(unsigned char)x2;
    unsigned char yy2[5]; //(unsigned char)x2;
    unsigned char recv[10];

    xx2[0] = 'x';
    yy2[0] = 'y';
    intToBuffer(&xx2[1], x2);
    intToBuffer(&yy2[1], y2);
    unsigned char buf[10]; // = {0x00};
    memcpy(buf, xx2, 5); //sizeof(int));
    memcpy(buf + 5, yy2, 5);
    //      memcpy(buf + sizeof(int), yy2, sizeof(int));

//      int i=0;
//      while(i<10){
//          /*x2 = rand();
//          intToBuffer(&xx2[1], x2);*/

//      int sentBytes = portSend(p1, buf, sizeof(buf));
//          //xx2[1] = 'f';
//          //xx2[2] = 'c';
//          //xx2[3] = 'v';
//          //xx2[4] = 'h';

//      portSend(p1, buf, 10);

//for(int k = 0; k < sizeof (xx2); k++){
//      //      portSend(p1, &xx2[k], 1);

//}
//portSend(p1, yy2, 5);
//portRecv(p1, recv, 5);
//printf("%d : ", x2);
//for(int j = 0; j < sizeof(xx2); j++)
//{
//      int count = 0;

```

```

//      //portSend(p1, &xx2[j], 1);
//
//      printf("[%x : ", xx2[j]);//%2hhx
//      portRecv(p1, &recv[j], 1);
//      printf("%x]  ", recv[j]);
//}

//      unsigned char buff;
//      portRecv(p1,&buff,15);

/*      unsigned char front[5];
      unsigned char left[5];
      unsigned char right[5];

      if (buff[0] = 'f'){
          for(int j=0; j<5; j++)
          {
              front[j] = buff[j+1];
          }
      }
      if (buff[5] = 'l'){
          for(int j=0; j<5; j++)
          {
              left[j] = buff[j+6];
          }
      }
      if (buff[0] = 'r'){
          for(int j=0; j<5; j++)
          {
              front[j] = buff[j+11];
          }
      }
*/      //      printf("\n");
//      printf("%d : ", y2);
//for(int j = 0; j < sizeof(xx2); j++)
//{
//      int count = 0;
//      //portSend(p1, &xx2[j], 1);
//
//      printf("[%x : ", yy2[j]);//%2hhx
//      portRecv(p1, &recv[j], 1);
//      printf("%x]  ", recv[j]);
//}

//      printf("\n");
//printf("set %d bytes\n", sentBytes);
//      i=1+1;
//      }

///  

//xfilter[c] = x2;
//yfilter[c] = y2;
//if (c = sample - 1){
//      c = 0;
//}

```

```

        //else{
        //    c++;
        //}
        //int minx = xfilter[0];
        //int maxx = xfilter[0];
        //int maxy = yfilter[0];
        //int miny = yfilter[0];
        //int s = 0;

        //while (s < sample){
        //    if (xfilter[s] < minx){
        //        minx = xfilter[s];}
        //    if (xfilter[s] > maxx){
        //        maxx = xfilter[s];}
        //    if (yfilter[s] < miny){
        //        miny = yfilter[s];}
        //    if (yfilter[s] > maxy){
        //        maxy = xfilter[s];}
        //    s = s+1;
        //}
        //s=0;
        //int totalx = 0;
        //int totaly = 0;
        //while (s < sample){
        //    totalx = totalx + xfilter[s];
        //    totaly = totaly + xfilter[s];
        //    s++;
        //}
        //x2 = totalx - minx - maxx/(sample - 2);
        //y2 = totaly - miny - maxy/(sample - 2);

        printf("y2      %d      ", y2);
        printf("y1      %d\n", y1);
        printf("x2      %d      ", x2);
        printf("x1      %d\n", x1);

        cvShowImage("WEB_CAMERA", tempImg);
        cvShowImage("Red", threshG);

        key = cvWaitKey(1);
    }
    cvReleaseCapture(&webcam);
/*
    cvReleaseCapture(&tempImg);
    cvReleaseCapture(&red);
    cvReleaseCapture(&blue);
    cvReleaseCapture(&green);
    cvReleaseCapture(&blurred);*/
    cvDestroyWindow("WEB_CAMERA");
    cvDestroyWindow("Red - Blue");
    cvDestroyWindow("Red - Green");
    cvDestroyWindow("Blurred");
    cvDestroyWindow("Red");
    return 1;
}

```


Documentation

| | | |
|---------------------------------------------|------------------------------------------------|-------|
| XMEGA A1 Microcontroller | ATxmega128A1 Preliminary | |
| XMEGA A Microcontroller Microcontrollers | XMEGA A manual Preliminary Application Note | XMEGA |
| USART GP2Y0A21YK.pdf | For IR sensor | From |
| SparkFun LV-MaxSonar-EZ3-Datasheet.pdf | For Sonar sensor | From |
| SparkFun XBee-Datasheet.pdf | For XBee module | From |
| SparkFun | | |

Resources

- Mike and Thomas's PVR board documentation and C libraries
- Code for USART serial communication from Tim
- James' read and write to port code
- Vishesh Vikas' help with setting up openCV

Special Thanks

Dr. Arroyo,
Dr. Schwartz,
Mike Pridgen,
Thomas Vermeer,
Ryan Stevens,
Tim Martin,
Devin Hughes,
Vishesh Vikas,
Jihyun Yoon (James)