Date: 12/06/2011
Student Name: Yue Bai
TAs: Ryan Stevens
Tim Martin
Instructors: Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

University of Florida
Department of Electrical and Computer Engineering
EEL 4665/5666
Intelligent Machines Design Laboratory

Final Report

Robot：Green helper

Table of Contents

Executive Summary

The Green Helper is an autonomous watering flower robot. The behaviors it exhibits includes obstacle avoidance, detecting a pot using IP cam and stop nearby it, actuating an arm which mounts a moisture sensor and grabbing the value and determining whether to water the plant based on the output of that moisture sensor.

In order to do obstacle avoidance, the Green Helper uses two sonar sensors mounted on front-left and front-right. If either the front-left or the front-right sonar sensor measure the distance between the robot and the obstacle less than twenty centimeters, it will turn left or right approximated ninety degrees angle. In addition, if both the front-left and the front-right sonar sensors feedback a distance less than twenty centimeters, the robot will go back a few length and than turn a one hundred and eighty degrees angle.

The IP camera in front of the robot is used to detect when a pot has been found. Through tracking a certain color, the robot can recognize where the pot is. Based on the feedback position of the pot, the robot controls the motor to move toward the pot and stop close to it. After that, the moisture sensor arm is dipped into the soil of the pot to determine whether the soil is moist or not.

Once moisture level has been determined, there are two possible behaviors. The first happens when the moisture sensor determines that the soil is dry/has not been watered. In this case, actuating another arm, which mounts a tube for a period of time, the water will flow into the soil through the tube. The other possible behavior is that the soil moisture sensor determines the plant to have already been watered. In this case, The Constant Gardener simply drives away.

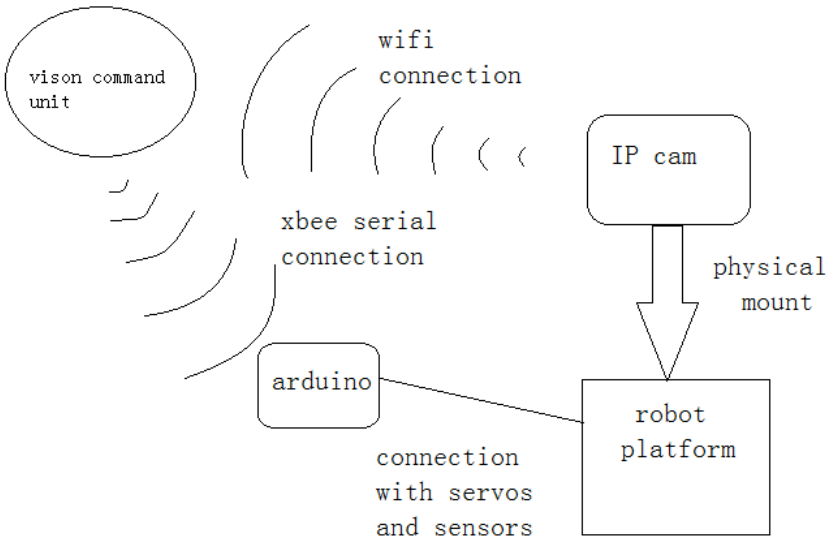
Abstract

This report covers the design of an autonomous robot “green helper” that can avoid obstacle, search flowerpot, detect the soil humidity of the flower and if the flower lack water, the robot will water the flower. Green helper will use popular arduino board along with IP camera to capture images for vision and image processing.

Introduction

Have you ever enjoy a long and happy trip but unfortunately when you come back home, you sadly find out that all your beautiful flowers have died because of lacking water? The purpose of the project is to create an autonomous robot, designed primarily for indoor use, that is capable of seeking out flowerpot and test the humidity of the soil. It then uses the water it carried to irrigate the flowers. The scope of the project primarily focuses on soil humidity detect, as well as the use of a IP camera for advanced image processing.

Integrated System



The diagram above shows a high level view of integration of the green helper autonomous watering flower system modules.

Mobile Platform

The mobile platform will be constructed of woods part designed in Solidworks software, cut and pieced in the lab. The body consists of two levels of wooden, rectangular disks. In the first level, the robot mounts to Arduino board and motor control board, two servos, two sonar sensors along with the IP camera. A tank contains water is placed on the top of the platform.

The structure of the mobile platform will be a 3-wheeled design. Two motor driven wheels are in the rear and one universal wheel is in the front. The platform will be simple but allow for enough space to mount for necessary sensors and equipment.

Actuation

The actuation on this robot consists primarily of its method of movement along with the motion and maneuverability of the arm. A Compact L298 Motor Controller controls two Pololu metal gearmotors with 90x10 mm wheels attached to the shafts. The arm is made out of wood fixed on the servos. It uses two MG995 servos, modified for 90-degree rotation.

Sensors

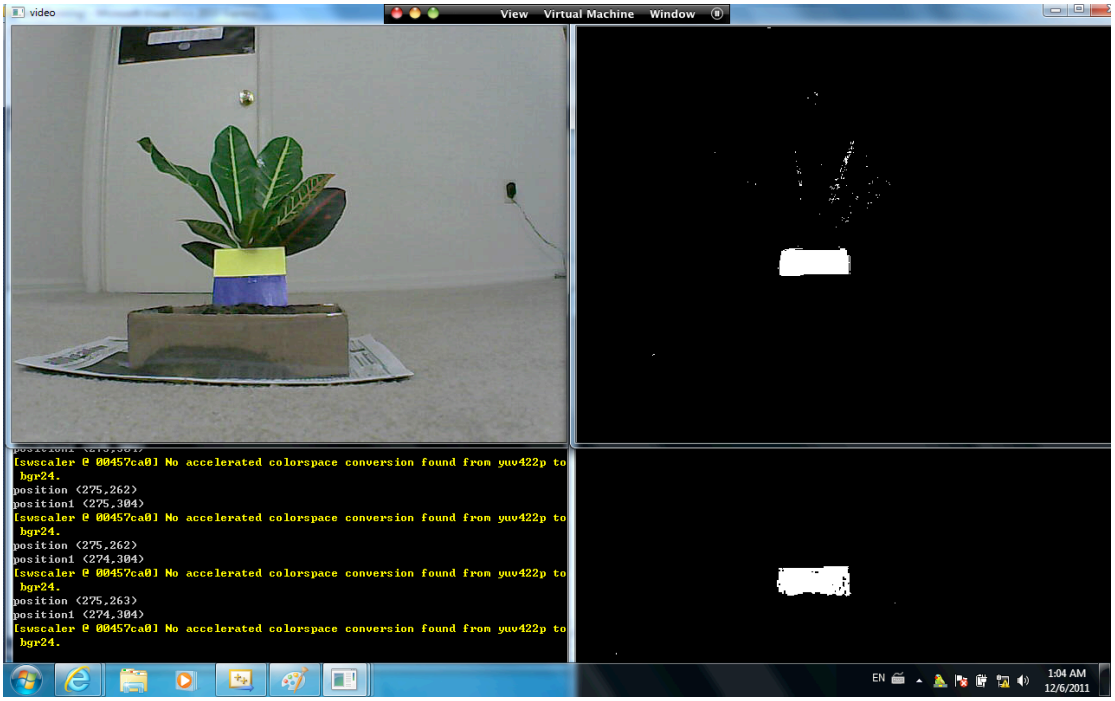
Bump Sensors
Bump sensors are utilized on the two front side of the robot. When using sonar sensors to do obstacle avoidance, if the robot moves too close along the wall, there are two blind areas that sonar sensors cannot work. Therefore, in order to prevent robot from hitting the wall, I use two bump sensors.

Sonar sensors
The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to micro controllers, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width the distance to target can easily be calculated.

IP camera

The camera used in the project is a FI8918W IP Wireless camera. An Xbee 1mW Chip Antenna-Serials1 is integrated into the arduino board to allow for wireless communication with an external laptop via Zigbee technology. Images are repeatedly captured via http protocol and then analyzed and modified via OpenCV libraries that contain numerous image processing, matrix operation, and transformation algorithms. With the help of these functions, the Green helper is able to recognize, and locate, green objects with very specific hue and saturation values.



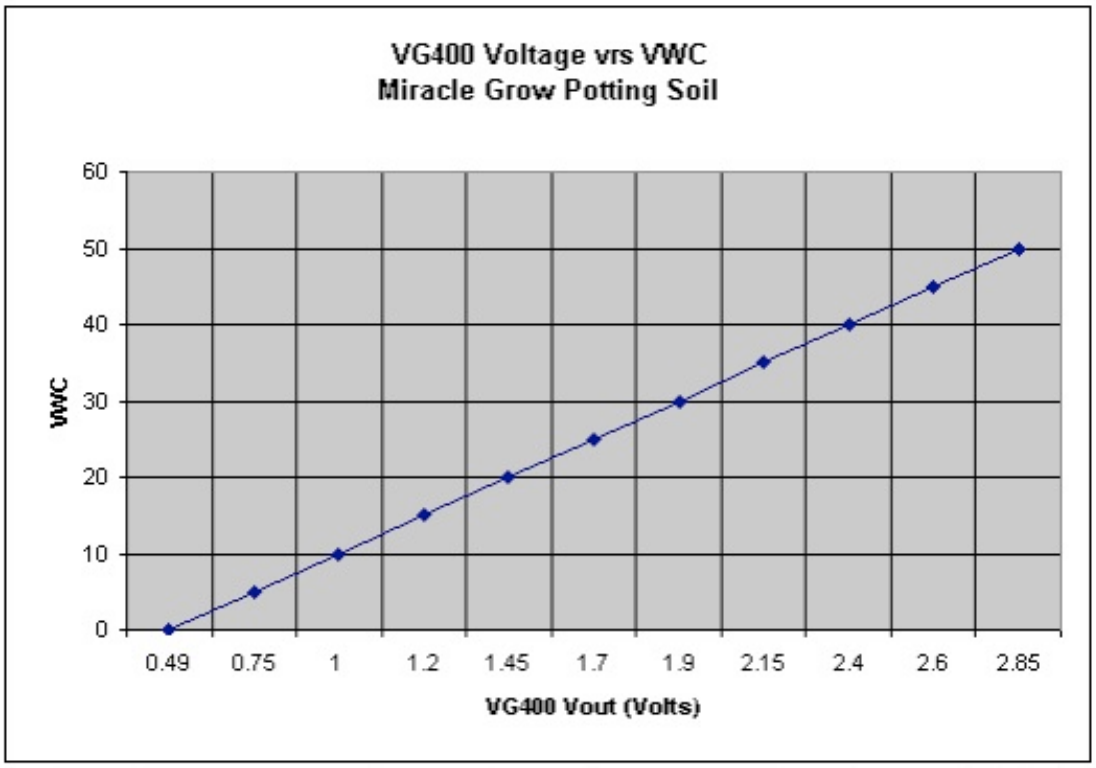
As the picture showed above, in order to correctly detect flowerpot, I place two color blocks on it. At first, I use only one color to do tracking, some other noise color may confuse the robot so that the robot cannot behave normal. Now, combing with two color to be recognized at the same time and have the same x position in the opencv screen, the robot can exactly know where the pot is.

Vegetronix Soil Moisture Sensor

The soil moisture sensor used is from a company called Vegetronix and the model number used is VG400. The sensor is shown in the figure below.

The Vegetronix VG400 Soil Moisture Sensor.

The intended purpose of the VG400 is to output a voltage that can be converted into Volumetric Water Content, which is the ratio of the volume of water to the volume of material (in this case, soil). Vegetronix provides an equation for converting the voltage output of the sensor to VWC, VWC= V*21.186 -10.381. Note that the equation is linear, which means that conversion to VWC is unnecessary if determination of "wetness" is all that is required. The following graph is provided by Vegetronix for the VG400.



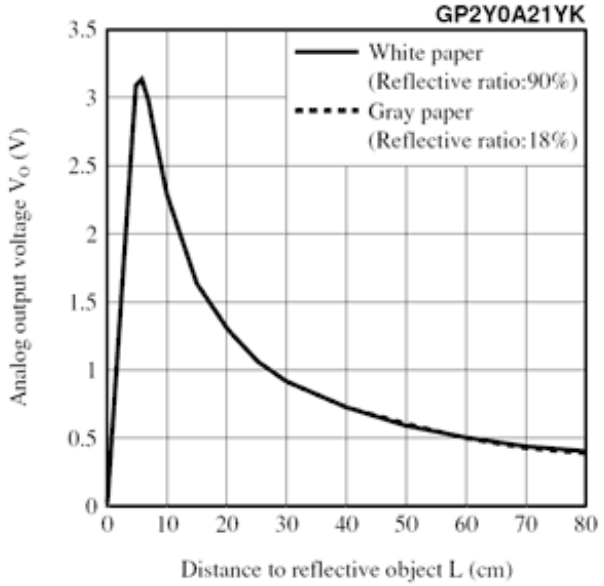
The test was performed to determine the output from the VG400 Soil Moisture Sensor. The output when the probe is placed at various depths in *very* moist soil, soil where the upper layer is moist and a cup of water is tabulated and graphed. The probe has markings for various depths, starting at point A at 5 mm to point R at 80 mm of depth.

Depth (mm)	Pure Water	Dry Soil	Less Dry Soil	Upper Layer Moist Soil
5	66	12	25	79
10	120	25	39	93
15	161	39	52	127
20	188	52	59	154
25	215	66	79	165
30	250	79	86	182
35	285	86	99	195
40	311	95	113	207
45	345	107	127	217
50	379	115	134	223
55	400	127	147	233
60	413	134	161	250
65	426	142	175	256
70	452	154	182	263
75	475	161	188	277
80	495	175	202	284

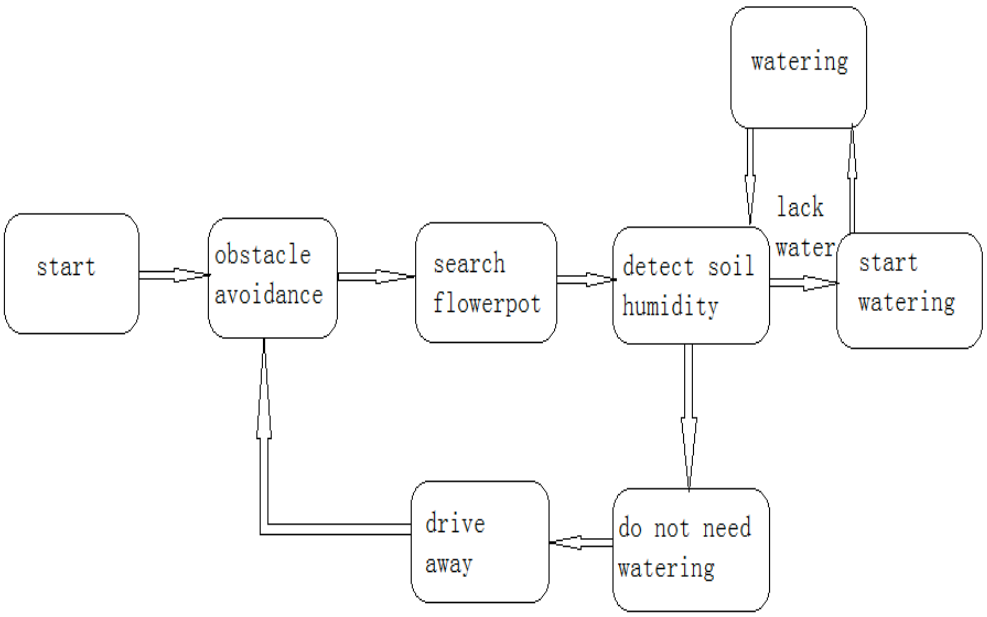
The outputs in three environments at various depths.

IR sensor

The GP2Y0A21 Sharp distance sensor is a great way to add obstacle avoidance or motion sensing to your robot or any other project. With a detection range of 4" to 32" and an analog voltage indicating the distance, this sensor is very easy to use.



Behaviors



Obstacle avoidance is the basic behaviors. The behavior is implemented by the two sonar sensors and two bump sensors. When one of the sides of the robot is almost hit something, it will trig the bump sensor and then the robot will correspondingly turn left or right forty-five degrees angle. If either the front-left or the front-right sonar sensor measure the distance between the robot and the obstacle less than twenty centimeters, it will turn left or right approximated ninety degrees angle. In addition, if both the front-left and the front-right sonar sensors feedback a distance less than twenty centimeters, the robot will go back a few length and than turn a one hundred and eighty degrees angle.

The next behavior is to detect the pot with IP camera. Once the robot find the pot, it will stop avoiding obstacle and move toward the pot. At the same time, the sonar sensor still work and if both two sonar sensor feedback the distance between robot and pot less than twenty centimeters along with detected pot using camera, the robot will stop. Either the front-left or front-right IRs

Finally, Green Helper spawns one of the two following behaviors: either a) water the plant if the soil moisture sensor output senses low moisture or b) simply do nothing and turn and drive away.

Conclusion

For Demo Day my robot was capable of object avoidance, detecting the pot, and detects the soil humidity. However, there are some problems. I need to focus on improving my tracking, getting the water dispenser working, and most importantly, have two plants for my demo. One should be dry, and one should be wet. I should change my code so that I do not simply stop my robot once I have found the plant, but instead continue to look for more plants. When my robot gets to the wet plant, it will detect that it is wet, and not water it, and when it detects the dry plant, it should water it.

In order to enhance my tracking, I place two color blocks on the flowerpot. At first, I use only one color to do tracking, some other noise color may confuse the robot so that the robot cannot behave normal. Now, combing with two colors to be recognized at the same time and have the same x position in the opencv screen, the robot can exactly know where the pot is.

In addition, I add condition judgment statement to determine whether or not to water the flower based on the soil humidity. And after this step, instead of stopping near the pot, the robot will go back and drive away to find others pot.

About this project, I spent too much time to find a good regulator and battery to drive my IP camera, I bought two regulators and three six volt batteries. But both of the regulators cannot work. And the output volt of the batteries is a little low. Then I solder a linear regulator by my own, but because of the large current of the camera, the regulator is so hot and not reliable. Finally, I bought a DC-DC regulator and it works well.

I think there is a lot of issue about my project can be improved in the future. This project provided countless benefits for me. I learned a multitude of lessons in various engineering aspects including: electrical, mechanical, and computer science. So I can build many more robots by applying the knowledge I learned from this class. Building robot is very interesting and it makes my life happy.

Appendix:

The following is the code used to implement the behaviors of The Green Helper.

```
#include <LiquidCrystal.h>
#include <Wire.h>
#include <Servo.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
Servo soil;
Servo water;
int i;
int pos = 0;
int echo1=2;
int echo2=4;
int echo3=12;
int m1=3;
int m2=5;
int m3=6;
int m4=11;
int bump1=7;
int bump2=8;
int analogPin = 0;
int analogPin2=2;
int length;
int val=0;
int led=13;
int com=0;
int serialflag;
int p=240;
long duration1,cm1,duration2,cm2,duration3,cm3;
void setup()
{
   Serial.begin(9600);
   lcd.init();
   lcd.backlight();
   lcd.clear();
   soil.attach(9);
   water.attach(10);
   soil.write(0);
   water.write(180);
   pinMode(m1,OUTPUT);
   pinMode(m2,OUTPUT);
   pinMode(m3,OUTPUT);
   pinMode(m4,OUTPUT);
```

```
    pinMode(led,OUTPUT);
    pinMode(bump1,INPUT);
    pinMode(bump2,INPUT);
    analogWrite(m1,p);
    analogWrite(m2,0);
    analogWrite(m3,p);
    analogWrite(m4,0);
    digitalWrite(bump1, HIGH);
    digitalWrite(bump2, HIGH);
}
void loop()
{
    pinMode(echo1,OUTPUT);
    digitalWrite(echo1,LOW);
    delayMicroseconds(2);
    digitalWrite(echo1,HIGH);
    delayMicroseconds(5);
    digitalWrite(echo1,LOW);
    pinMode(echo1,INPUT);
    duration1=pulseIn(echo1,HIGH);

    pinMode(echo2,OUTPUT);
    digitalWrite(echo2,LOW);
    delayMicroseconds(2);
    digitalWrite(echo2,HIGH);
    delayMicroseconds(5);
    digitalWrite(echo2,LOW);
    pinMode(echo2,INPUT);
    duration2=pulseIn(echo2,HIGH);


    cm1=duration1/29/2;
    cm2=duration2/29/2;
    lcd.setCursor(0, 0);
    lcd.print(cm1);
    lcd.print("cm1");
    lcd.setCursor(0, 0);
    lcd.setCursor(10, 10);
    lcd.print(cm2);
    lcd.print("cm2");
    length = analogRead(analogPin2);
    lcd.setCursor(0, 0);
    lcd.setCursor(0, 1);
    lcd.print(length);
```

```
lcd.print("vv");
   delay(100);
   serial();
   bump();
   if(serialflag==0)    avoid();


}
void serial()
{
      if (Serial.available()>0)
      {
                  serialflag=1;
                  com = Serial.read();
                  if ('a' == com)
                        {
                              digitalWrite(led,HIGH);
                              delay(100);
                              digitalWrite(led,LOW);
                              analogWrite(m1,p);
                              analogWrite(m2,0);
                              analogWrite(m3,0);
                              analogWrite(m4,0);
                              delay(150);
                              analogWrite(m1,p);
                              analogWrite(m2,0);
                              analogWrite(m3,p);
                              analogWrite(m4,0);

                        }
                  if ('b' == com)
                        {
                              digitalWrite(led,HIGH);
                              delay(100);
                              digitalWrite(led,LOW);
                              analogWrite(m1,0);
                              analogWrite(m2,0);
                              analogWrite(m3,p);
                              analogWrite(m4,0);
                              delay(150);
                              analogWrite(m1,p);
                              analogWrite(m2,0);
                              analogWrite(m3,p);
                              analogWrite(m4,0);
```

```
                }

            if ('c'==com)
                {
                    digitalWrite(led,HIGH);
                    delay(100);
                    digitalWrite(led,LOW);
                    analogWrite(m1,p);
                    analogWrite(m2,0);
                    analogWrite(m3,p);
                    analogWrite(m4,0);

                }
             if(length>350) {
                                                    analogWrite(m1,0);
                                                    analogWrite(m2,0);
                                                    analogWrite(m3,0);
                                                    analogWrite(m4,0);
                                                    servo();
                                                    analogWrite(m1,0);
                                                    analogWrite(m2,p);
                                                    analogWrite(m3,0);
                                                    analogWrite(m4,p);
                                                    delay(1000);
                                                    analogWrite(m1,p);
                                                    analogWrite(m2,0);
                                                    analogWrite(m3,0);
                                                    analogWrite(m4,0);
                                                    delay(2200);
                                                    analogWrite(m1,p);
                                                    analogWrite(m2,0);
                                                    analogWrite(m3,p);
                                                    analogWrite(m4,0);
                                            }

            Serial.flush();

    }
    else    serialflag=0;
}
void bump()
{
    if(digitalRead(bump1)==LOW)
```

```
    {
        analogWrite(m1,0);
        analogWrite(m2,p);
        analogWrite(m3,0);
        analogWrite(m4,p);
        delay(500);
        analogWrite(m1,p);
        analogWrite(m2,0);
        analogWrite(m3,0);
        analogWrite(m4,0);
        delay(500);
        analogWrite(m1,p);
        analogWrite(m2,0);
        analogWrite(m3,p);
        analogWrite(m4,0);
    }
    if(digitalRead(bump2)==LOW)
    {
        analogWrite(m1,0);
        analogWrite(m2,p);
        analogWrite(m3,0);
        analogWrite(m4,p);
        delay(500);
        analogWrite(m1,0);
        analogWrite(m2,0);
        analogWrite(m3,p);
        analogWrite(m4,0);
        delay(500);
        analogWrite(m1,p);
        analogWrite(m2,0);
        analogWrite(m3,p);
        analogWrite(m4,0);
    }
}
void avoid()
{
      if(cm1<20&&cm2<20)
    {

        analogWrite(m1,0);
        analogWrite(m2,p);
        analogWrite(m3,0);
        analogWrite(m4,p);
        delay(1000);
```

```
        analogWrite(m1,p);
        analogWrite(m2,0);
        analogWrite(m3,0);
        analogWrite(m4,0);
        delay(2200);
        analogWrite(m1,p);
        analogWrite(m2,0);
        analogWrite(m3,p);
        analogWrite(m4,0);

  }
  else if(cm1<20)         //    left sonar
          {
            analogWrite(m1,p);
            analogWrite(m2,0);
            analogWrite(m3,0);
            analogWrite(m4,0);
            delay(1200);
            analogWrite(m1,p);
            analogWrite(m2,0);
            analogWrite(m3,p);
            analogWrite(m4,0);
          }
     else     if(cm2<20)
          {
            analogWrite(m1,0);
            analogWrite(m2,0);
            analogWrite(m3,p);
            analogWrite(m4,0);
            delay(1200);
            analogWrite(m1,p);
            analogWrite(m2,0);
            analogWrite(m3,p);
            analogWrite(m4,0);
          }
     }

void servo()
{
  for(pos = 0; pos < 75; pos += 1)
  {
    soil.write(pos);
    delay(15);
  }
```

```
   for(i=0;i<6;i++)
   {
      val = analogRead(analogPin);
      lcd.setCursor(0, 0);
      lcd.setCursor(10, 0);
      lcd.print(val);
      lcd.print("cc");
      delay(1000);
   }
  for(pos = 75; pos >=0; pos -= 1)
   {
      soil.write(pos);
      delay(15);
   }
     if(val<150)      waterpot();

}

void waterpot()
{

   for(pos = 180; pos>=90; pos-=1)
   {
      water.write(pos);
      delay(15);
   }
   delay(5000);
   for(pos = 90; pos < 180; pos += 1)
   {
      water.write(pos);
      delay(15);
   }

}
```

The following is the code used to implement the color tracking on the laptop.

```
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>
#using <System.dll>
#include "stdafx.h"

using namespace System;
using namespace System::IO::Ports;
using namespace System::Threading;

IplImage* GetThresholdedImage(IplImage* img)

{// Convert the image into an HSV image

	IplImage* imgHSV = cvCreateImage(cvGetSize(img), 8, 3);

	cvCvtColor(img, imgHSV, CV_BGR2HSV);

	IplImage* imgThreshed = cvCreateImage(cvGetSize(img), 8, 1);

	cvInRangeS(imgHSV, cvScalar(30, 100, 100), cvScalar(50, 255, 255), imgThreshed);

	cvReleaseImage(&imgHSV);

	return imgThreshed;

}

IplImage* GetThresholdedImage2(IplImage* img)

{// Convert the image into an HSV image

	IplImage* imgHSV = cvCreateImage(cvGetSize(img), 8, 3);

	cvCvtColor(img, imgHSV, CV_BGR2HSV);

	IplImage* imgThreshed = cvCreateImage(cvGetSize(img), 8, 1);

	cvInRangeS(imgHSV, cvScalar(115, 100, 100), cvScalar(135, 255, 255), imgThreshed);

	cvReleaseImage(&imgHSV);
```

```
	return imgThreshed;

}

int main()

{

SerialPort^ serialPort = gcnew SerialPort(
L"COM3",
9600,
Parity::None,
8,
StopBits::One);
serialPort->Open();

// Initialize capturing live feed from the camera

//CvCapture* capture = 0;

// capture = cvCaptureFromCAM(0);

//cvSetCaptureProperty(capture,CV_CAP_PROP_FRAME_WIDTH,480);

//cvSetCaptureProperty(capture,CV_CAP_PROP_FRAME_HEIGHT,480);

CvCapture*capture=
cvCreateFileCapture("http://192.168.1.5/videostream.cgi?user=admin&pwd=&resolution=32&rat
e=0&a=.mjpg"); // For IP cam

// Couldn't get a device? Throw an error and quit

	if(!capture)

	{

		printf("Could not initialize capturing...\n");

		return -1;

	}

	// The two windows we'll be using
```

```
cvNamedWindow("video");
cvNamedWindow("thresh");
cvNamedWindow("thresh2");
// This image holds the "scribble" data...
// the tracked positions of the ball
IplImage* imgScribble = NULL;
// An infinite loop
while(true)
{
	// Will hold a frame captured from the camera
	IplImage* frame = 0;
	frame = cvQueryFrame(capture);
	// If we couldn't grab a frame... quit
	if(!frame)
		break;
		// If this is the first frame, we need to initialize it
	if(imgScribble == NULL)
	{
		imgScribble = cvCreateImage(cvGetSize(frame), 8, 3);
	}
	// Holds the yellow thresholded image (yellow = white, rest = black)
	IplImage* imgYellowThresh = GetThresholdedImage(frame);
```

```
IplImage* imgblueThresh = GetThresholdedImage2(frame);

                // Calculate the moments to estimate the position of the ball

    CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));

    cvMoments(imgYellowThresh, moments, 1);

    // The actual moment values

    double moment10 = cvGetSpatialMoment(moments, 1, 0);

    double moment01 = cvGetSpatialMoment(moments, 0, 1);

    double area = cvGetCentralMoment(moments, 0, 0);

// Holding the last and current ball positions

    static int posX = 0;

    static int posY = 0;

    int lastX = posX;

    int lastY = posY;

    posX = moment10/area;

    posY = moment01/area;

    cvMoments(imgblueThresh, moments, 1);

    double momentb10 = cvGetSpatialMoment(moments, 1, 0);

    double momentb01 = cvGetSpatialMoment(moments, 0, 1);

    double areab = cvGetCentralMoment(moments, 0, 0);

// Holding the last and current ball positions

    static int posbX = 0;

    static int posbY = 0;
```

```
int lastbX = posbX;

int lastbY = posbY;

posbX = momentb10/areab;

posbY = momentb01/areab;

if(posbX>0 && posbY>0 && posX>0 && posY>0)

{

        printf("position (%d,%d)\n", posX, posY);

        printf("position1 (%d,%d)\n", posbX, posbY);

}

if(posbX>390&&posbX<640&&posX>390&&posX<640){

serialPort->WriteLine("a");}

if(posbX<250&&posbX>0&&posX<250&&posX>0){

      serialPort->WriteLine("b");}

if(posbX>250&&posbX<390&&posX>250&&posX<390){

      serialPort->WriteLine("c");}

//cvAdd(frame, imgScribble, frame);

cvShowImage("thresh", imgYellowThresh);

cvShowImage("thresh2", imgblueThresh);

cvShowImage("video", frame);

            // Wait for a keypress

int c = cvWaitKey(10);

if(c!=-1)
```

```
            {
                // If pressed, break out of the loop
                break;
            }
            // Release the thresholded image+moments... we need no memory leaks.. please
            cvReleaseImage(&imgYellowThresh);
            cvReleaseImage(&imgblueThresh);
            delete moments;
        }
        // We're done using the camera. Other applications can now use it
        cvReleaseCapture(&capture);
        serialPort->Close();
        return 0;
}
```