Final Report

Kechen Tang

Gift- Servant robot

University of Florida
MAE Graduate student
EEL 4665/5666
Intelligent Machines Design Laboratory

Instructors:
Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

TAs :
Tim Martin
Josh Weaver

Table of Content

1. Abstract

The robot named Gift is an autonomous robot that could search the ball with special color and take it to the place ordered. Arduino2560 board is the microcontroller board for the robot. Hacked servo motor is the engine of the robot. Iphone is used as a webcamera to take picture of the environment around it and send the image data through wifi to the laptop .After recieving the date, the laptop will do the analysis and then use bluetooth to tell the robot whether there is special color ball around and the orientation of it. Sonar is used to detect the the distance between the ball and robot to help catch it procisely. After successfully finish the goal, the led will flare and the laptop will play a music.

2. Executive Summary

The purpose of the Gift is to help take the special color ball back to the place ordered. This is done through use webcamera and sonar. The webcamera for the robot is Iphone's backcamera, Iphone is placed in the front of the robot,I use a small wood stand to support it. The sonar is also put in the front of the robot, it could detect thing and the distance in front the robot.

The platform of the robot is made of plastic. The microcontroller board Arduino2560 is place on the back. Engine is four hacked servo moter. Two 7805 voltage regulator chips is used to make a power board to provide 5V electrical power to the board and moter seperately to reduce the disturb to the board and sensor from high current of motor. The platform is actually is not just including the robot, but also including the laptop, opencv technology to support the robot. So Gift is not just a robot, it is a robot system, which is one of the most important charactristics for this robot.

The behavior for the robot is that Gift turn around and used bluetooth to recieve data from laptop to know whether the ball is detected. If the ball is detected, the robot will decode the orientation from the bluetooth data and turn toward that direction. At the same time, sonar is detecting the distance in front of the robot, When the ball is wihin the range of access, Robot will catch the ball and start to go back. The place orderd is also specified with special color. Similar method with chasing ball could help robot to get there.

3. Introduction

The robot I will be designing in the Fall of 2012 is a servant robot named Gift. Nowadays, robot are used more and more widely. Having a robot in the house to help people do some housework is no longer just a dream. Gift is a simple example robot that could help you pick the ball with color you wanted and then take it back to the ordered place with special color to specify.

4. *Integrated System*

4.1 Organization of the system

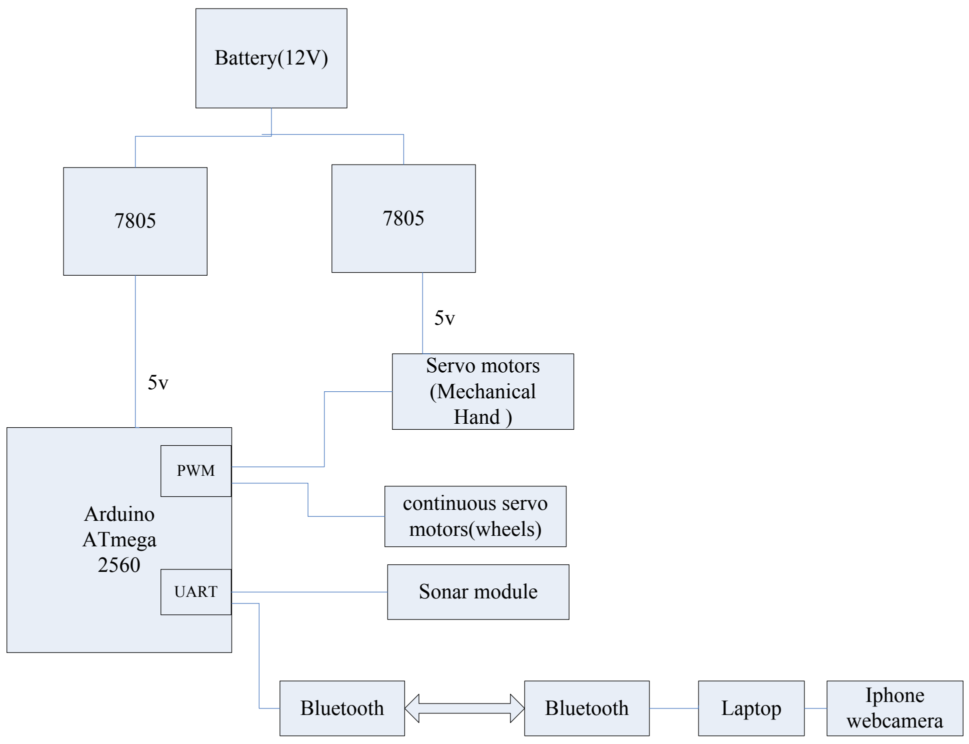


Figure 1: Flow chat of integrated system

Arduino ATmega2560 borad works as the head of the whole system. It use pwm to control four continuous servo motors to make the robot walking forward, backward, turning left and right. Also, Mega2560 controls the Servo motors to control the mechanical hand to catch the things. Sonor module is connected to the microcontroller through UART port. Another important sensor is iphone webcamera. I use bluetooth and laptop to build a bridge connecting the iphone webcamera to the microcontroller.

4.2 Some Parts' details

Microcontroller—Arduino mega2560

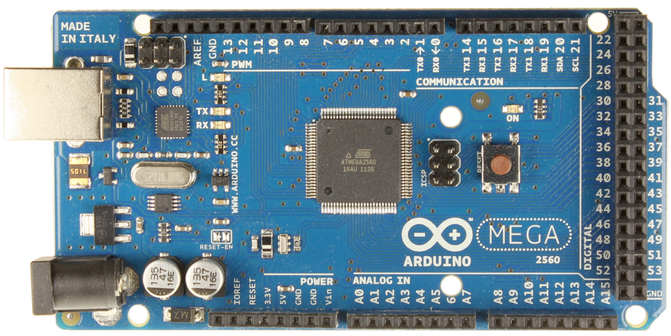


Figure 2: Arduino2560 board

The Cube Terminator will be using the Arduino2560 board as control center. The Arduino2560 board utilizes a ATmega2560 for its processing. This board peripherals that are enough to make Gift a functional robot such as 12 bit ADC, USARTs, UARTs, four 16 bit timers, and twelve PWM channels. These are enough to control 6 servo motors and could support to communicating with laptop and computer easily.

5. *Mobile Platform*

The Mobile platform is mainly consist of two part. One is the main body, using a plastic board and four motor including wheels, which could allow the robot to make the moving action smoothly. Another part is the mechanical hand, which is made of wood and fixed in front of the robot. Two servo motor is used to control the hand to catch and losen.

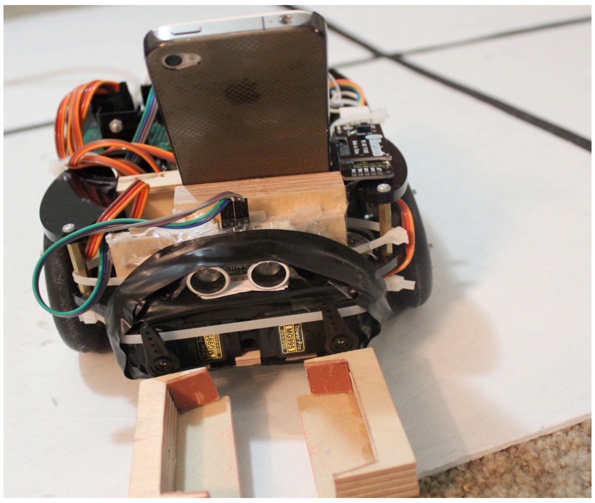


Figure 3: Mobile Platform

6. *Actuation*

The platform is moved by two continuous servo motors(DS04-KVC). Mega2560 microcontroller use PWM to control servo motors,the servo control command can be used to control the continuous servo motor. When the write.servo command setting the angles to 90 will stop the servos, setting them greater than 90 will move them clockwise (in reference to the wheel side), and setting them less than 90 will move them counterclockwise. Since the wheels were mounted opposite to each other, to move the platform forward one side is set to rotate clockwise and the other is set to counterclockwise. This method is used to turn the robot right, left, and back.

Figure 4: Iphone webcamera

7. *Sensors*

Webcamera

Main Sensor in the robot is webcamera, webcamera take pictures of the front and send them to the laptop. Then,laptop process the picture to decide where the black road is leading, and which color ball is in front and whether the robot need to pick it. Also webcamera could help

I will use my iphone as webcamera. Detail mathod is to crack iphone first, so that we can install softwares by ourself. Then install a software named Mobiola WebCamera in the phone. This solfware could send the pictures from iphone's camera to the certain ip address. In computer side, you need to install computer driver, which can be download on the Mobiala's website. This drive could let your computer recieve the picture from the iphone. In this way,I can use treat iphone webcamera just as a usb camera, which could make me programing much easier.

Color detection and black line follow will needs Opencv to do the image processing. First, using opencv to get the webcamera RGB image, then convert the RGB image to HSV image, which classify the color by color's saturation,lightness and value and could help us judge the color more accurately. Then try to find out the color range that you want to search. Then pick up the areas that have color that you want and calculate the center coordination of these areas. Then we get the position of the ball.

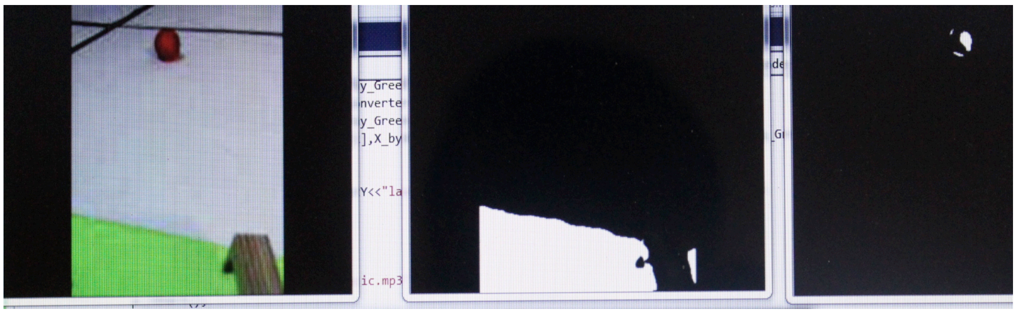
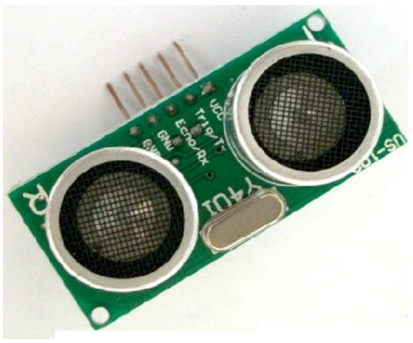


Figure 6 :Sonar US-100 UART mode signal



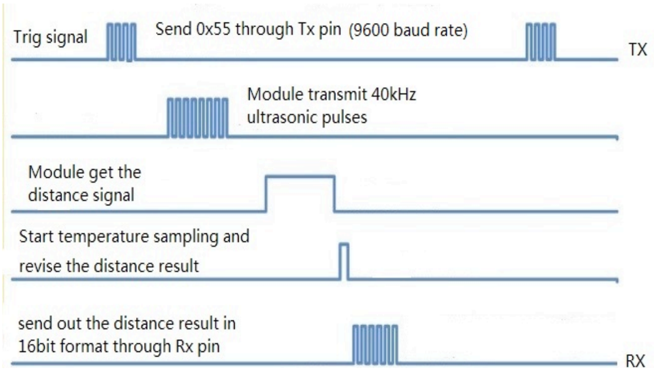
Sonar

Another important sensor for the robot is sonar, It could give the information about the distance between the thing procisely.

The sonar's model is US-100. The distance range for sensor is from 2cm to 450cm. The accuracy is 0.3cm+1% Sensor angle is less than 15 degree.

This sonar module have two working mode. One is High-Voltage-Trig-Mode, and the other one is UART mode. Here, I choose UART mode and connect the TX&RX pin of the sonar to the UART port2 of the microcontroller board.

When Sonar working in UART mode, microcontroller can get the distance data by just send out 0x55 to the UART port in 9600 baud rate; and then, the module would send back the 16bit distance data to the UART port.


Trig signal
Send 0x55 through Tx pin (9600 baud rate)
TX
Module transmit 40kHz ultrasonic pulses
Module get the distance signal
Start temperature sampling and revise the distance result
send out the distance result in 16bit format through Rx pin
RX

8. *Communication*

These two bluetooth modules can automatically reconize and connect with each other. This bluetooth module support UART in put.

Therefore, I could just connect one bluetooth module to the UART port of the Mega2560. Detail method is to connect the RXD pin of the bluetooth to the TX port of Mega2560, TXD pin to the RX port of 2560. Also, I need to connect the other bluetooth to the laptop. My laptop doesn't have serial port. So,I need a serial-USB transfer to connect the bluetooth and my laptop. The connection way is the same as connection between bluetooth and Mega2560 UART port(RXD-TXD,TXD-RXD).

9. Behaviors

Search for the ball

When Start the Gift will intiallize all the sensor and communication module. Then waiting for the connection with laptop. After getting connection with laptop. The robot start to turn around and tcheck the data sent from laptop through bluetooth to see whether there is the ball seen by webcamera.

Catch the ball

After determine that there is red ball in the webcamera. The Gift will trying to adjust it's direction to towards the ball, which is to put the center of ball in the middle of the image. Then run foward to get close to the ball.When the sonar's data indicate that the distance between the robot and ball is less than a certain number, The Gift will close the hand to catch the ball.

Going back

After catch the ball, The robot will go back to the ordered place. It start to turn around to search the color that stands for the home. After find such place in the webcamera image. The robot will move to that area. After letting the color center in the bottom of the webcamera, which means that the home is just right in the bottom of the robot. Then judge that mission accomplished, and led start flaring and send order to let laptop to play music.

10. Experimental Layout and Results

1. Test for sonar

Test method：Put the cube obstacle in front of the robot and use sonar to test the distance between them. Use serial port monitor in laptop to show the data of the distance. Then use rule to measure the distance. Compare these two values and calculate the error range. If the error is big, then try to find method to correct the error.(Sonar testing code is attached to Appendex)

Table: test result

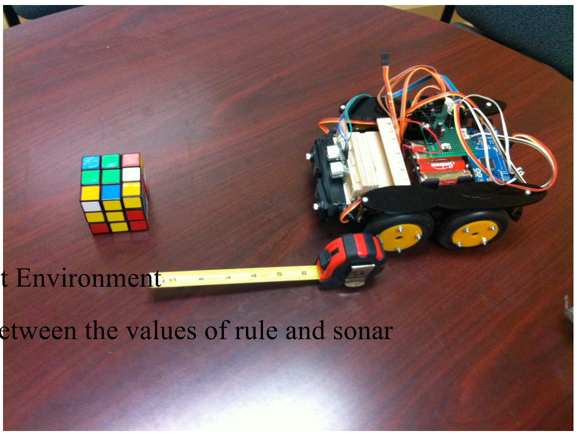
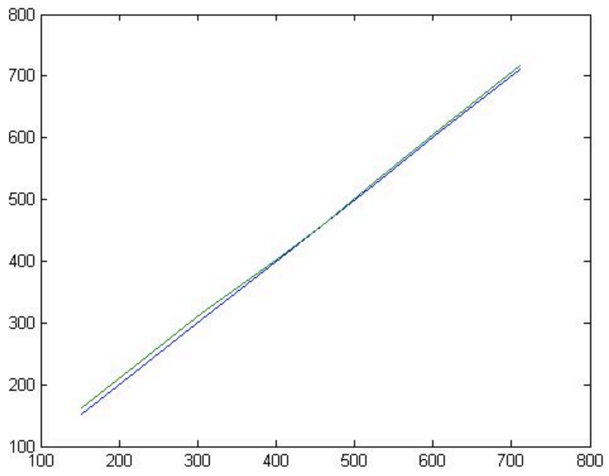


Figure: Test Environment

Figure: comparation between the values of rule and sonar

Test result:

Rule measure distance	Sonar measure distance	error
6inch(152.5mm)	163mm	10mm
9inch(228.6mm)	238mm	9.4mm
1 feet (305mm)	316mm	11mm
1feet 4 inch(406.4mm)	408mm	1.6mm
1feet 6inch (457.5mm)	458mm	0.5mm
1feet 9inch(533.4)	537mm	3.6mm
2 feet(610 mm)	615mm	5mm
2feet 4inch(711.2mm)	718mm	6.8mm



2. Test for webcamera function

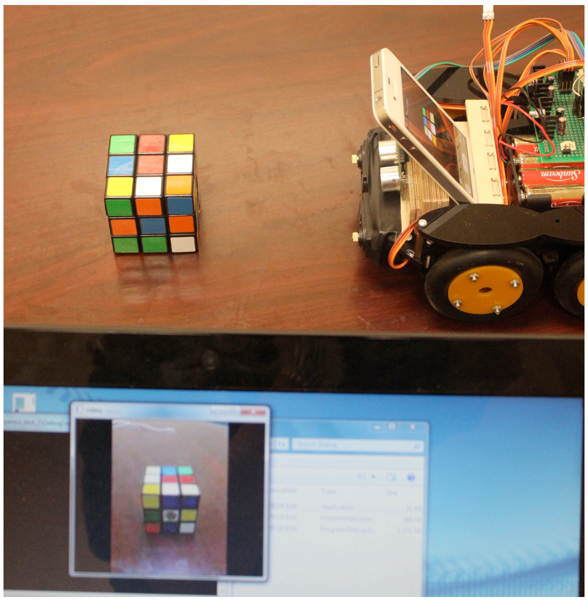
Test method:

1.Connect iphone and laptop to same router.

2.Run the Mobiola WebCamera in Iphone and set the iphone to connect with laptop.

3.Run the Opencv test project in the computer

Test result:



3. Test for webcamera color detection.
4. Test for webcamera black line reconization

11. *Conclusion*

The whole experience in making this robot gives me many problems and learn me a lot. A few important lessons learned:

1.Using Solidworks to design the platform
2.Using Opencv to processing image
3.Serial port communication
4.Laptop—Microcontroller System
5.Random strategy
6.Test your mechanical systems thoroughly to avoid surprises on demo day!

The robot is finished and a lot of experience learned. Also, in the future I could improve this robot to equip it with some other even more cool function.

12. *Documentation*

[1].Datasheet for US-100 sonar
[2].Datasheet of ATmega2560

[3]."Object Detection & Tracking using Color",
http://opencv-srf.blogspot.com/2010/09/object-detection-using-color-seperation.html

[4]."Detection of planar objects",

http://docs.opencv.org/doc/tutorials/features2d/detection_of_planar_objects/detection_of_planar_objects.html#detectionofplanarobjects

13. *Appendices A:*Sonar testing code

:

```
void setup(){
  // connect Arduino's RX2&TX2 pins to US-100's Echo/Rx&Trig/Tx pin
  Serial.begin(9600); // set baud rate to 9600bps.
  Serial2.begin(9600);
  Serial.print("system start");
}

void loop(){
  unsigned int lenHigh = 0; // high 8 bit for sonar data
  unsigned int lenLow = 0;  // low 8 bit for sonar data
  unsigned int dist_mm = 0; // distance

  Serial2.flush();     // clear serial data buffer
  Serial2.write(0x55); // send 0x55, trig US-100 to start the distance sampling
  delay(500);         // delay 500ms

  // if data in serial data buffer is more than 2 byte
  if(Serial2.available() >= 2){
    lenHigh = Serial2.read();        // high 8 bit for the distance data
    lenLow = Serial2.read();         // low 8 bit for the distance data
    dist_mm = lenHigh*256 + lenLow; // calculate the distance

    // effective distance range is from 1mm to 10m
    if((dist_mm > 1) && (dist_mm < 10000))
    {
      Serial.print("Distance is: ");// show the results in monitor.
      Serial.print(dist_mm, DEC);
      Serial.println("mm");
    }
  }
  delay(500); // delay 500mm
}
```

Appendices B: Laptop runing opencv code:

```
#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
#using<system.dll>
using namespace System;
using namespace System::IO::Ports;
using namespace System::Diagnostics;
using namespace System::ComponentModel;

#include "iostream"
using namespace std;

IplImage* imgTracking;
IplImage* imgTracking_Green;
IplImage* imgTracking_Blue;
int lastX = -1;
int lastY = -1;
int object =-1;
int lastX_Green = -1;
int lastY_Green = -1;
int lastX_Blue = -1;
int lastY_Blue = -1;
double area;
double area_Green;
double area_Blue;
//This function threshold the HSV image and create a binary image
IplImage* GetThresholdedImage(IplImage* imgHSV){
    IplImage* imgThresh=cvCreateImage(cvGetSize(imgHSV),IPL_DEPTH_8U, 1);
    cvInRangeS(imgHSV, cvScalar(170,160,60), cvScalar(180,256,256), imgThresh);
//red
    return imgThresh;
}
//for green
IplImage* GetThresholdedImage_Green(IplImage* imgHSV){
    IplImage* imgThresh=cvCreateImage(cvGetSize(imgHSV),IPL_DEPTH_8U, 1);
    cvInRangeS(imgHSV,        cvScalar(40,50,70),        cvScalar(90,256,256),
imgThresh);//green
    return imgThresh;
}
//for blue
IplImage* GetThresholdedImage_Blue(IplImage* imgHSV){
    IplImage* imgThresh=cvCreateImage(cvGetSize(imgHSV),IPL_DEPTH_8U, 1);
    cvInRangeS(imgHSV, cvScalar(110,100,50), cvScalar(130,256,256), imgThresh);
//blue
    return imgThresh;
}
void trackObject(IplImage* imgThresh){
// Calculate the moments of 'imgThresh'
CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(imgThresh, moments, 1);
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);
 area = cvGetCentralMoment(moments, 0, 0);
     // if the area<1000, I consider that the there are no object in the image and
it's because of the noise, the area is not zero
if(area>50)
{
int posX = moment10/area;
int posY = moment01/area;

lastX = posX;
lastY = posY;
}

free(moments);
}

void trackObject_Green(IplImage* imgThresh){
// Calculate the moments of 'imgThresh'
CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(imgThresh, moments, 1);
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);

 area_Green = cvGetCentralMoment(moments, 0, 0);
     // if the area<1000, I consider that the there are no object in the image and
it's because of the noise, the area is not zero
if(area_Green>1000)
{
int posX = moment10/area_Green;
int posY = moment01/area_Green;


lastX_Green = posX;
lastY_Green = posY;
}
```

```
free(moments);
}

void trackObject_Blue(IplImage* imgThresh){
// Calculate the moments of 'imgThresh'
CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(imgThresh, moments, 1);
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);

 area_Blue = cvGetCentralMoment(moments, 0, 0);
     // if the area<1000, I consider that the there are no object in the image and
it's because of the noise, the area is not zero
if(area_Blue>50)
{
int posX = moment10/area_Blue;
int posY = moment01/area_Blue;


lastX_Blue = posX;
lastY_Blue = posY;
}

free(moments);
}

    static void DataReceivedHandler(
                        Object^ sender,
                        SerialDataReceivedEventArgs^ e)
    {
        SerialPort^ sp = (SerialPort^)sender;
        int Data_number = sp->BytesToRead;
        int i;
        //sp->WriteLine("recieved");
        array<Byte>^Data_Recieved_Array={0x00};
            sp->Read(Data_Recieved_Array,0,1);
            if(Data_Recieved_Array[0]==0x55)
            {
                array<Byte>^X_byteArray = BitConverter::GetBytes(lastX);
                X_byteArray->Reverse(X_byteArray);
                array<Byte>^Y_byteArray = BitConverter::GetBytes(lastY);
                Y_byteArray->Reverse(Y_byteArray);
    array<Byte>^Data_Array={0x55,X_byteArray[2],X_byteArray[3],Y_byteArray[2],Y
_byteArray[3]};
                sp->Write(Data_Array,0,5);
                sp->ReadExisting();
                cout<<"lastX="<<lastX<<"lastY="<<lastY<<endl;
            }
            if(Data_Recieved_Array[0]==0xa5)
            {
                array<Byte>^X_byteArray_Green                                  =
BitConverter::GetBytes(lastX_Green);
                X_byteArray_Green->Reverse(X_byteArray_Green);
                array<Byte>^Y_byteArray_Green                                  =
BitConverter::GetBytes(lastY_Green);
                Y_byteArray_Green->Reverse(Y_byteArray_Green);

    array<Byte>^Data_Array={0x55,X_byteArray_Green[2],X_byteArray_Green[3],Y_by
teArray_Green[2],Y_byteArray_Green[3]};
                sp->Write(Data_Array,0,5);
                sp->ReadExisting();
                cout<<"lastX_G="<<lastX_Green<<"lastY_G="<<lastY_Green<<endl;
            }
            if(Data_Recieved_Array[0]==0x57)
            {
                array<Byte>^X_byteArray_Blue                                   =
BitConverter::GetBytes(lastX_Blue);
                X_byteArray_Blue->Reverse(X_byteArray_Blue);
                array<Byte>^Y_byteArray_Blue                                   =
BitConverter::GetBytes(lastY_Blue);
                Y_byteArray_Blue->Reverse(Y_byteArray_Blue);

    array<Byte>^Data_Array={0x55,X_byteArray_Blue[2],X_byteArray_Blue[3],Y_byte
Array_Blue[2],Y_byteArray_Blue[3]};
                sp->Write(Data_Array,0,5);
                sp->ReadExisting();
                cout<<"lastX_B="<<lastX_Blue<<"lastY_B="<<lastY_Blue<<endl;
            }
            else if(Data_Recieved_Array[0]==0xaa)
            {

                    Process^ p = gcnew Process;
                    p->StartInfo->FileName = "f:\\music.mp3";
                    p->Start();
                    cout<<"MUSIC"<<endl;
            }
            else if(Data_Recieved_Array[0]==0xff)
            {
                    cout<<Data_Recieved_Array<<endl;
            }



    };


int main(){
//========================Serial                port                configuration
=====================================
        SerialPort^ mySerialPort = gcnew SerialPort("COM6");
        mySerialPort->BaudRate = 9600;
        mySerialPort->Parity = Parity::None;
        mySerialPort->StopBits = StopBits::One;
        mySerialPort->DataBits = 8;
        mySerialPort->Handshake = Handshake::None;
        mySerialPort->DataReceived                   +=                  gcnew
SerialDataReceivedEventHandler(DataReceivedHandler);
        mySerialPort->Open();

//========================Serial                port                configuration
=====================================




      CvCapture* capture =0;
      capture = cvCaptureFromCAM(0);
      if(!capture){
printf("Capture failure\n");
return -1;
      }

      IplImage* frame=0;
      frame = cvQueryFrame(capture);
      if(!frame) return -1;

     //create a blank image and assigned to 'imgTracking' which has the same size
of original video
     imgTracking=cvCreateImage(cvGetSize(frame),IPL_DEPTH_8U, 3);
     imgTracking_Green=cvCreateImage(cvGetSize(frame),IPL_DEPTH_8U, 3);
     imgTracking_Blue=imgTracking_Green;
     cvZero(imgTracking); //covert the image, 'imgTracking' to black
     cvZero(imgTracking_Green); //covert the image, 'imgTracking' to black
     cvZero(imgTracking_Blue); //covert the image, 'imgTracking' to black
     cvNamedWindow("Video");
     cvNamedWindow("Ball");
     cvNamedWindow("Green");
     cvNamedWindow("Blue");

      //iterate through each frames of the video
      while(true){

            frame = cvQueryFrame(capture);
            if(!frame) break;
            frame=cvCloneImage(frame);

           cvSmooth(frame, frame, CV_GAUSSIAN,3,3); //smooth the original image
using Gaussian kernel

            IplImage* imgHSV = cvCreateImage(cvGetSize(frame), IPL_DEPTH_8U, 3);
            cvCvtColor(frame, imgHSV, CV_BGR2HSV); //Change the color format from
BGR to HSV
            IplImage* imgThresh = GetThresholdedImage(imgHSV);
            IplImage* imgThresh_green = GetThresholdedImage_Green(imgHSV);
            IplImage* imgThresh_Blue = GetThresholdedImage_Blue(imgHSV);
            cvSmooth(imgThresh, imgThresh, CV_GAUSSIAN,3,3); //smooth the binary
image using Gaussian kernel
            cvSmooth(imgThresh_green, imgThresh_green, CV_GAUSSIAN,3,3); //smooth
the binary image using Gaussian kernel
            cvSmooth(imgThresh_Blue, imgThresh_Blue, CV_GAUSSIAN,3,3); //smooth
the binary image using Gaussian kernel
          //track the possition of the ball
          trackObject(imgThresh);
          trackObject_Green(imgThresh_green);
          trackObject_Blue(imgThresh_Blue);
          if(area<50)
          {
            lastX=0;
            lastY=0;


          }
          if(area_Green<150)
          {
            lastX_Green=0;
            lastY_Green=0;

          }
          if(area_Blue<50)
          {
            lastX_Blue=0;
            lastY_Blue=0;

          }

          // Add the tracking image and the frame
          cvAdd(frame, imgTracking, frame);

          cvAdd(frame, imgTracking_Green, frame);
          cvAdd(frame, imgTracking_Blue, frame);
          cvShowImage("Ball", imgThresh);
          cvShowImage("Green", imgThresh_green);
          cvShowImage("Blue", imgThresh_Blue);
           cvShowImage("Video", frame);

           //Clean up used images
           cvReleaseImage(&imgHSV);
           cvReleaseImage(&imgThresh);
           cvReleaseImage(&imgThresh_green);
           cvReleaseImage(&imgThresh_Blue);
           cvReleaseImage(&frame);

            //Wait 10mS
            int c = cvWaitKey(10);
            //If 'ESC' is pressed, break the loop
            if((char)c==27 ) break;
      }

      cvDestroyAllWindows() ;
      cvReleaseImage(&imgTracking);
      cvReleaseImage(&imgTracking_Green);
      cvReleaseImage(&imgTracking_Blue);
      cvReleaseCapture(&capture);

      return 0;
}
```

Appendices C: Arduino2560 code

```
#include <Servo.h>

Servo lfservo;  // create servo object to control a servo
Servo lbservo;                 // a maximum of eight servo objects can be created
Servo rfservo;
Servo rbservo;
Servo Lcatch;
Servo Rcatch;
char  Goal_Color=1;
int x_Red=-1;
int y_Red=-1;
int x_Green=-1;
int y_Green=-1;
int x_Blue=-1;
int y_Blue=-1;
int predelay=1;
long randNumber;

  int distance=0;
void setup()
{

  Serial.begin(9600);//bluetooth
  Serial1.begin(9600);//bluetooth
  Serial2.begin(9600);//sonar\\

  Lcatch.attach(6);
  Rcatch.attach(7);
 // randomSeed(analogRead(0));//random generator.
  Losen_Catch();
  //Catch();
 Serial.write("start") ;
}

void loop()
{



            TurnToTheItem();
             if(x_Red!=0)
             {
}*/

               if(y_Red<180)
               {  /*
                     distance=DetectThedistance();
                     if(distance<130)
                     {
                           ob_avoidance();
                     }
                     */
                 }
                 else
                   {
                     distance=DetectThedistance();
                      Serial.println(distance, DEC);
                      if(distance<90)
                      {
                        Catch();
                        Go_Blue_Pad();
                        Serial.println("blue");
                        Go_Green_Pad();
                        while(1);
                      }
                    }

}
}

/*
void Randomturn()
{
  int randNumber;
  randNumber=random(600,3400);
  if(randNumber>=2000)
  {
      randNumber=4000-randNumber;
      TurnRight();
  }
  else
      turnleft();
   delay(randNumber);

}*/

void Bluetooth_Read_Red()
{
  byte sign,x_R_High,x_R_Low,y_R_High,y_R_Low;
    Serial1.flush();     // 清空串口接收缓冲区
    Serial1.write(0x55); // 发送 0x55，触发 US-100 开始测距
    delay(150);         // 延时 500 毫秒
    if(Serial1.available()>= 5)
    {

      sign= Serial1.read();
      if(sign==0x55)
      {
        x_R_High= Serial1.read();        // high-byte of the distance
        x_R_Low = Serial1.read();         // low-byte of the distance
        y_R_High = Serial1.read(); //calculate the distance
        y_R_Low = Serial1.read();
        x_Red=x_R_High*256+x_R_Low;
        y_Red=y_R_High*256+y_R_Low;
       }
       else
         {x_Red=-1;y_Red=-1;}
       Serial1.flush();
    }
}
void Bluetooth_Read_Green()
{
  byte sign,x_G_High,x_G_Low,y_G_High,y_G_Low;
    Serial1.flush();     // 清空串口接收缓冲区
    Serial1.write(0xa5); // 发送 0x55，触发 US-100 开始测距
    delay(150);         // 延时 500 毫秒

    if(Serial1.available()>= 5)
    {
      sign= Serial1.read();
      if(sign==0x55)
      {
        x_G_High = Serial1.read();        // high-byte of the distance
        x_G_Low = Serial1.read();         // low-byte of the distance
        y_G_High = Serial1.read(); //calculate the distance
        y_G_Low = Serial1.read();
        x_Green=x_G_High*256+x_G_Low;
        y_Green=y_G_High*256+y_G_Low;
      }
      else
      {x_Green=-1;y_Green=-1;
      }
      Serial1.flush();
  }
}
void Bluetooth_Read_Blue()
{
    byte sign,x_B_High,x_B_Low,y_B_High,y_B_Low;
    Serial1.flush();     // 清空串口接收缓冲区
    Serial1.write(0x57); // 发送 0x55，触发 US-100 开始测距
    delay(150
    );          // 延时 500 毫秒

     if(Serial1.available()>= 5)
    {
      sign= Serial1.read();
      if(sign==0x55)
      {
        x_B_High = Serial1.read();        // high-byte of the distance
        x_B_Low = Serial1.read();         // low-byte of the distance
        y_B_High = Serial1.read(); //calculate the distance
        y_B_Low = Serial1.read();
        x_Blue=x_B_High*256+x_B_Low;
        y_Blue=y_B_High*256+y_B_Low;
      }
      else
      {x_Blue=-1;y_Blue=-1;}
      Serial1.flush();
  }
}


int DetectThedistance()
{
  unsigned int lenHigh = 0; // 高位
  unsigned int lenLow = 0;  // 低位
  unsigned int dist_mm = 0; // 距离

   Serial2.flush();     // 清空串口接收缓冲区
    Serial2.write(0x55); // 发送 0x55，触发 US-100 开始测距
    delay(20);          // 延时 500 毫秒

    if(Serial2.available() >= 2)
    {
        lenHigh = Serial2.read();        // high-byte of the distance
        lenLow = Serial2.read();         // low-byte of the distance
        dist_mm = lenHigh*256 + lenLow; // calculate the distance
    }
    return dist_mm;
}

void ob_avoidance()
{
    Run_Back();
    delay(1000);
    TurnRight();
    delay(2000);
    RunForward();
    delay(1500);
}


void Catch()
{
  Lcatch.write(150);
  Rcatch.write(50);

}

void Losen_Catch()
{
  Lcatch.write(31);
  Rcatch.write(160);
}

void Go_Green_Pad()
{
  int i=1;
  while(i)
  {
    int turn;
     Bluetooth_Read_Green();

     Serial.print("xGreen=");// 输出结果至串口监视器
      Serial.print(x_Green, DEC);
      Serial.println("");
     turn=x_Green-180;
       if(abs(turn)>40)
       {
         if(turn>0)
```

```
                TurnRight();
          else
                TurnLeft();
        }
        else
        {
          if(y_Green<220)
          RunForward();
          else
          i++;//
        }
        if(i==3)
        i=0;
      }
  Robot_Stop();
}


void Go_Blue_Pad()
{
  int i=1;
  while(i)
  {
    int turn;
     Bluetooth_Read_Blue();

      turn=x_Blue-180;
       if(abs(turn)>40)
       {
         if(turn>0)
               TurnRight();
         else
               TurnLeft();
       }
       else
       {
         if(y_Blue<220)
         RunForward();
         else
         i++;//
       }
       if(i==3)
       i=0;
     }
  Robot_Stop();
}

void TurnToTheItem()
  {
    Bluetooth_Read_Red();
    if(x_Red==-1)
    {
      Robot_Stop();
    }
    else
     {
       int turn=x_Red-180;
       if(abs(turn)>20)
       {
         if(turn>0)
               TurnRight();
         else
               TurnLeft();
       }
       else
       {
               RunForward();
       }
    }
  }

void Robot_Stop()
{
 lbservo.detach();
 lfservo.detach();
 rbservo.detach();
 rfservo.detach();
}
void TurnLeft()
{

  lfservo.attach(2);  // attaches the servo on pin 9 to the servo object
  lbservo.attach(3);  // attaches the servo on pin 10 to the servo object
  rfservo.attach(4);  // attaches the servo on pin 11 to the servo object
  rbservo.attach(5);  // attaches the servo on pin 12 to the servo object
  lfservo.write(105);              // tell servo to go to position in variable 'pos'
  lbservo.write(105);              // tell servo to go to position in variable 'pos'
  rfservo.write(105);              // tell servo to go to position in variable 'pos'
  rbservo.write(105);              // tell servo to go to position in variable 'pos'

  }

void TurnRight()
{
  lfservo.attach(2);  // attaches the servo on pin 9 to the servo object
  lbservo.attach(3);  // attaches the servo on pin 10 to the servo object
  rfservo.attach(4);  // attaches the servo on pin 11 to the servo object
  rbservo.attach(5);  // attaches the servo on pin 12 to the servo object
  lfservo.write(94);              // tell servo to go to position in variable 'pos'
  lbservo.write(94);              // tell servo to go to position in variable 'pos'
  rfservo.write(94);              // tell servo to go to position in variable 'pos'
  rbservo.write(94);              // tell servo to go to position in variable 'pos'
}


void RunForward()
{
  lfservo.attach(2);  // attaches the servo on pin 9 to the servo object
  lbservo.attach(3);  // attaches the servo on pin 10 to the servo object
  rfservo.attach(4);  // attaches the servo on pin 11 to the servo object
  rbservo.attach(5);  // attaches the servo on pin 12 to the servo object
  rfservo.write(105);              // tell servo to go to position in variable 'pos'
  rbservo.write(105);              // tell servo to go to position in variable 'pos'
  lfservo.write(92);              // tell servo to go to position in variable 'pos'
  lbservo.write(92);              // tell servo to go to position in variable 'pos'
}
void Run_Back()
{
  lfservo.attach(2);  // attaches the servo on pin 9 to the servo object
  lbservo.attach(3);  // attaches the servo on pin 10 to the servo object
  rfservo.attach(4);  // attaches the servo on pin 11 to the servo object
  rbservo.attach(5);  // attaches the servo on pin 12 to the servo object
  rfservo.write(92);              // tell servo to go to position in variable 'pos'
  rbservo.write(92);              // tell servo to go to position in variable 'pos'
  lfservo.write(105);              // tell servo to go to position in variable 'pos'
  lbservo.write(105);              // tell servo to go to position in variable 'pos'


}
```