Guai Shu Shu

EEL5666c

Instructors:
A. Antonio Arroyo
Eric M. Schwartz

Teaching Assistants:
Tim Martin
Ryan Stevens

Student Name:
shengkai Kong

Date:
2012/1/30

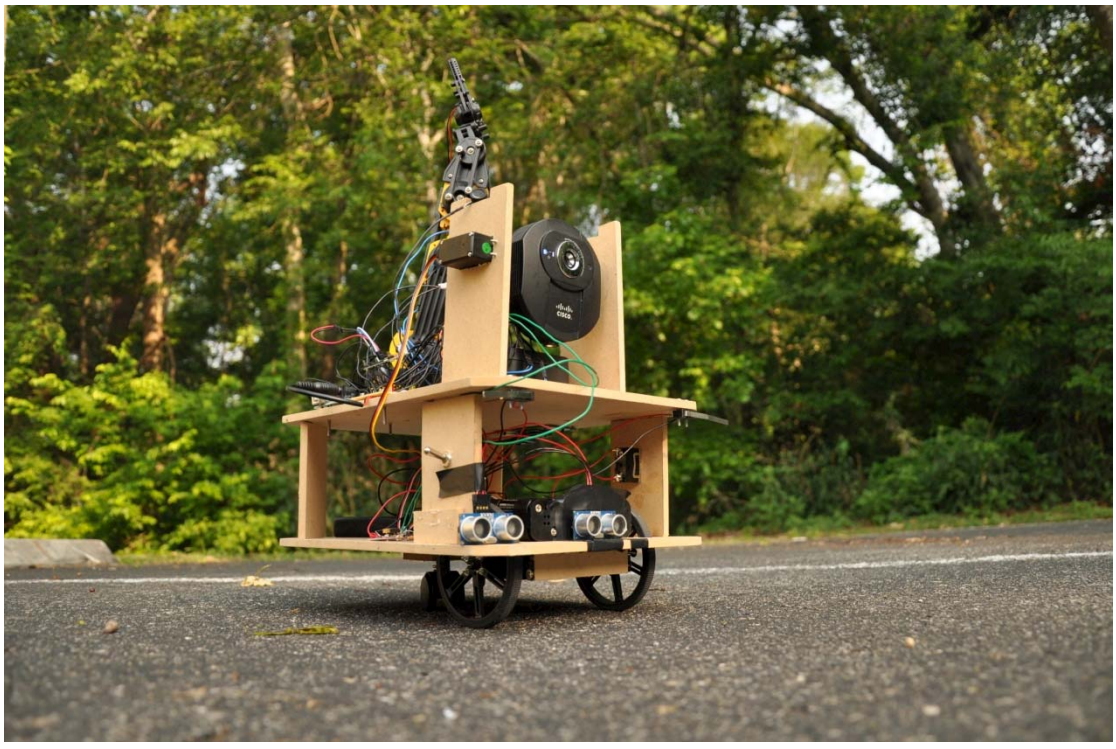


Table of Contents

Abstract

The purpose of Guai Shu Shu is to follow the kids based on their color, when get close to the kid, the Guai Shu Shu will ask them which candy they want, the robot can provide two kinds of candies, once the kid make the answer, Guai Shu Shu will use his arm to give them candies.

Executive Summary

The project is designed to provide kid candies, make them happy, encourage them to study the engineer.

The Guai Shu Shu has two levels, first level has wheels and servo motors, IP camera and two Ultrasonic sensors in the front, arduino board+ Xbee shield, and two batteries packs for arduino and IP camera respective. This level can make the robot follow the kid, and do obstacle avoidance.

The second level consists robot arm, the arm will be controlled by the arduino board+ voice recognition shield and motor driver circuit. This level can make the robot listen and talk. Move his arm to give kids candy.

Introduction

The Guai Shu Shu has two main targets:

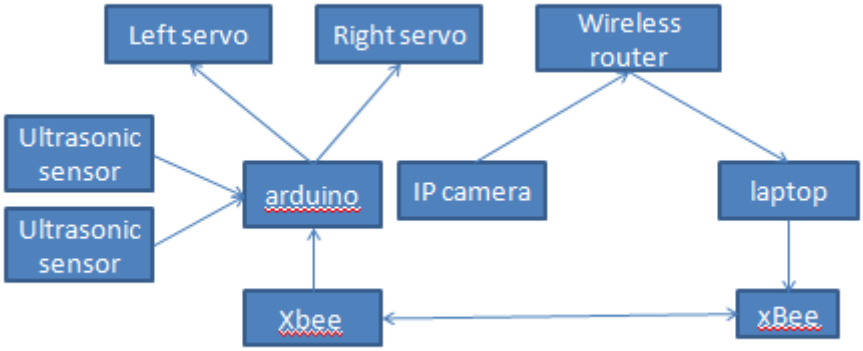
First is visual sense, the robot can see and do image processing. The process is follow.

The IP camera capture the images and send them to the laptop via wireless router. The laptop can do image process, filter the color and send instructions back to the Guai Shu Shu to make sure the robot face to the right direction.

Second is audio, the robot can record the voice, and can understand the man's speech.

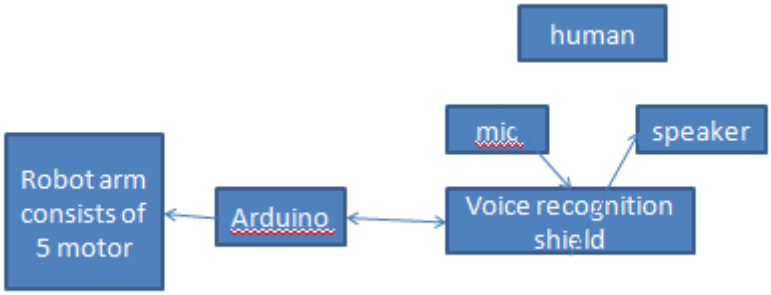
The voice recognition shield can achieve that goal. not only listen but also talk.

Integrated System



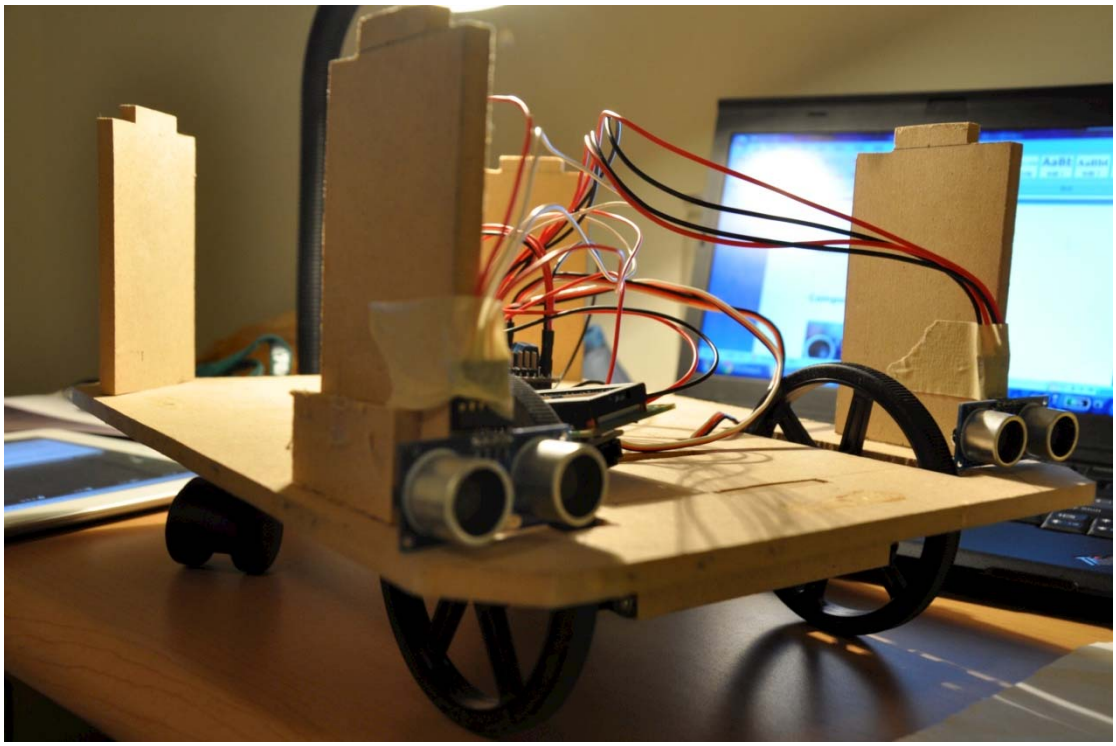
Two Ultrasonic Sensors can help the arduino know the distances in the front. The arduino can provide different plus wild to control the servo. Base on these components , the robot can do obstacle avoidance.

Ip camera send images to the laptop, laptop send instructions to the arduino via Xbee. When Xbee on the robot receive instruction, the arduino stop doing obstacle avoidance, and do color tracking.



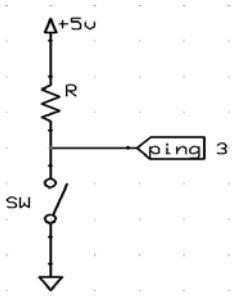
The voice recognition shield can interface with human, and use arduino to control the robot arm.

Mobile Platform



Components

touch sensor:



When touch sensor been touched, the arduino will receive a high signal, but to eliminate the noise, I use delay(50); to check the bump switch is actually been touch.

```
void Touch()
{
  buttonState=digitalRead(2);
  if(buttonState==HIGH)
  {
    delay(50);
    if(buttonState==HIGH)
    {
       move(45,90,500);
    }
  }
  buttonState=digitalRead(3);
  if(buttonState==HIGH)
  {
    delay(50);
    if(buttonState==HIGH)
    {
       move(90,45,500);
    }
  }}
```

ultrasonic sensor

HC-SR04 is cheap Ultrasonic sensor, the sensors are used to detect obstacles on the left and right sides of the robot. Tell the robot how far from the obstacles.


```
void avoidance()//forward and do abstacle avoidance
{
  lcd.setCursor(0, 0);
  lcd.print(uLeft.Ranging(CM));
  lcd.setCursor(10, 0);
  lcd.print(uRight.Ranging(CM));
  if(uLeft.Ranging(CM)<20 && uRight.Ranging(CM)<20)
  {
    move(180,180,500);
    move(90,0,500);//turn right
  }
  if(uLeft.Ranging(CM)<20)
  {
     move(0,90,500);
  }
  if(uRight.Ranging(CM)<20)
  {
      move(90,0,500);
  }
```

Xbee shield

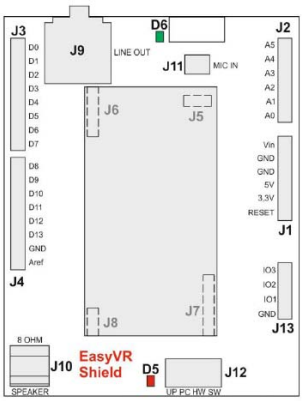
```
void Xbee()
{
    if (Serial.available()>0)
    {
              serialflag=1;
              com = Serial.read();
              if ('q' == com) {
                move(90, 0,100);}
              if ('w' == com) {
                move(60,0,100);}
              if ('e'==com)  {
                move(30,0,100);}
              if ('r'==com)  {
                move(0,0,100);}
```

Part of code in arduino, receive different character, give different responses.

Servo

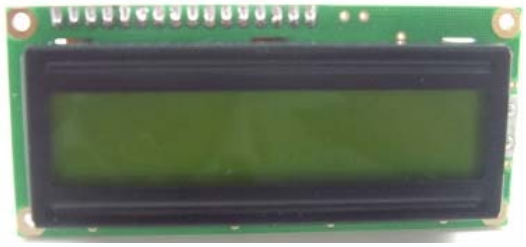
The servo control two wheels, the arduino give different plus wide, we can get different speed.

EasyVR shield



The voice recognition shield, it will give the robot ability listen and talk.
reference: http://www.veear.eu/Products/EasyVRShield.aspx

lcd1602



```
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Ultrasonic.h>

LiquidCrystal_I2C lcd(0x27,16,2);  //
```

add the library

```
 lcd.setCursor(0, 0);
 lcd.print(uLeft.Ranging(CM));
 lcd.setCursor(10, 0);
 lcd.print(uRight.Ranging(CM));
```

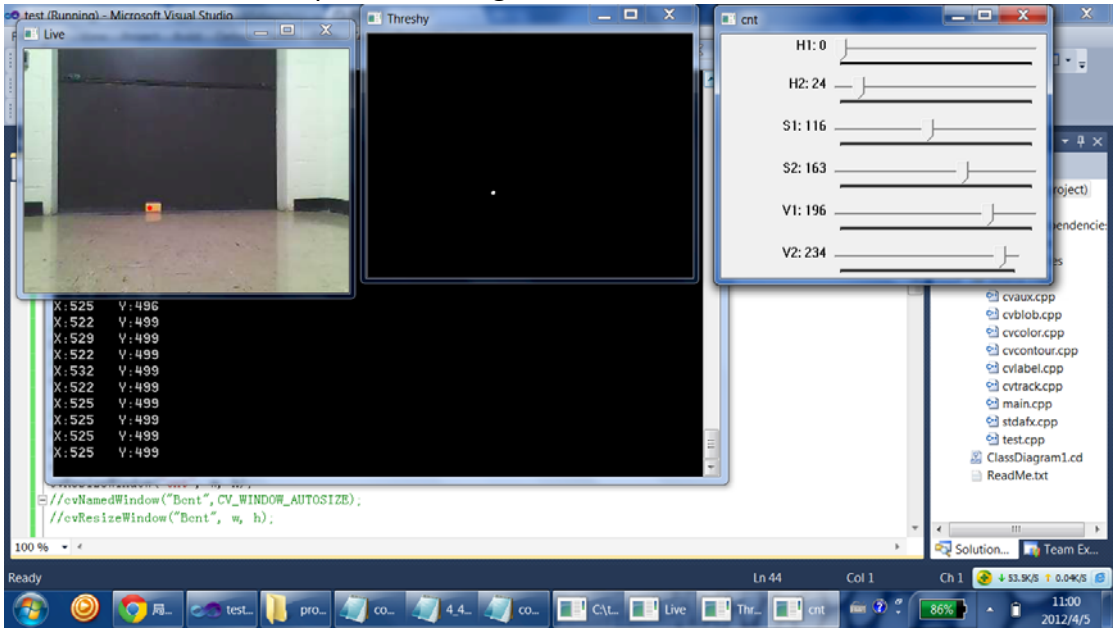
set the position of the cursor, and display the distance.

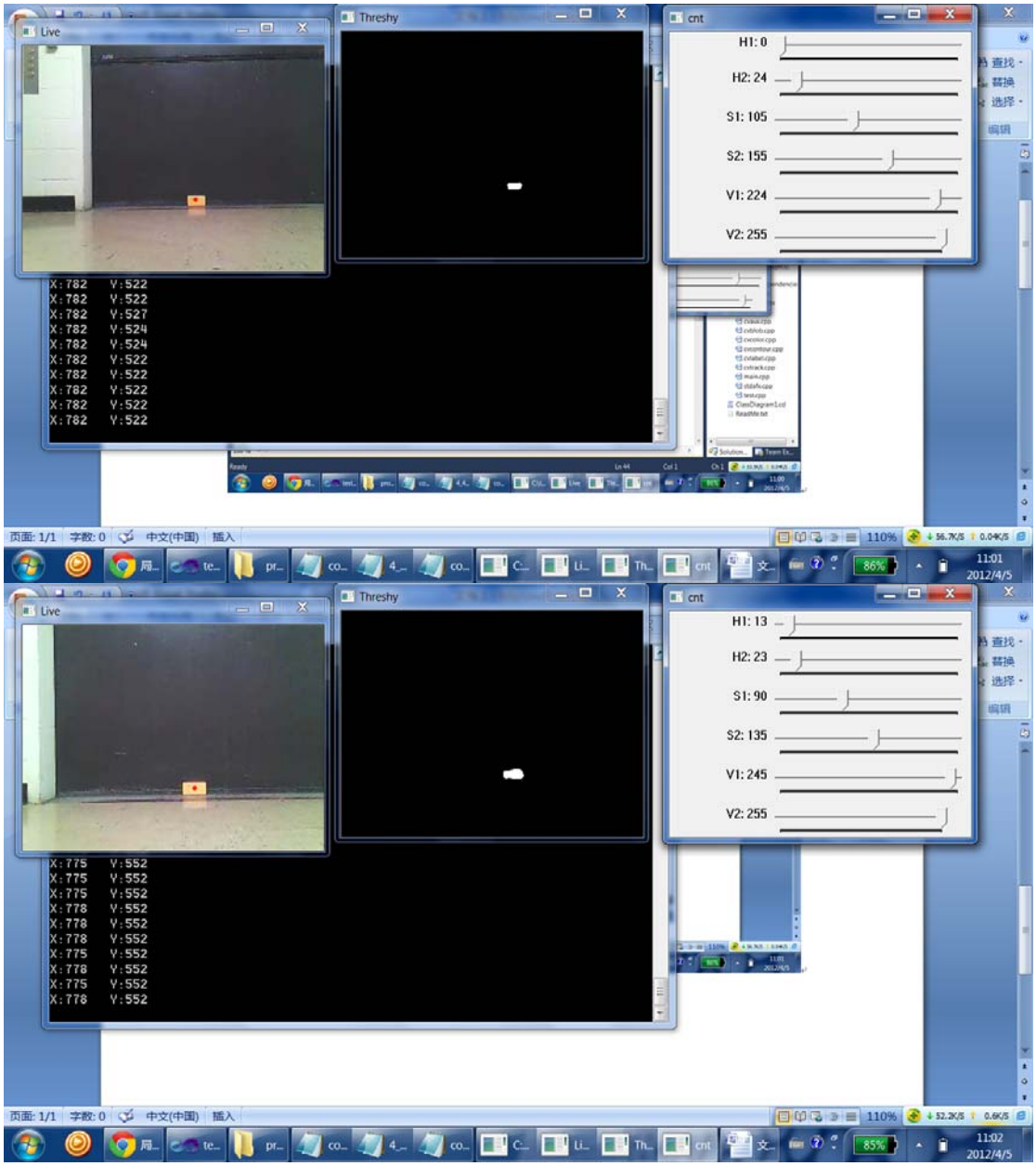
robot arm

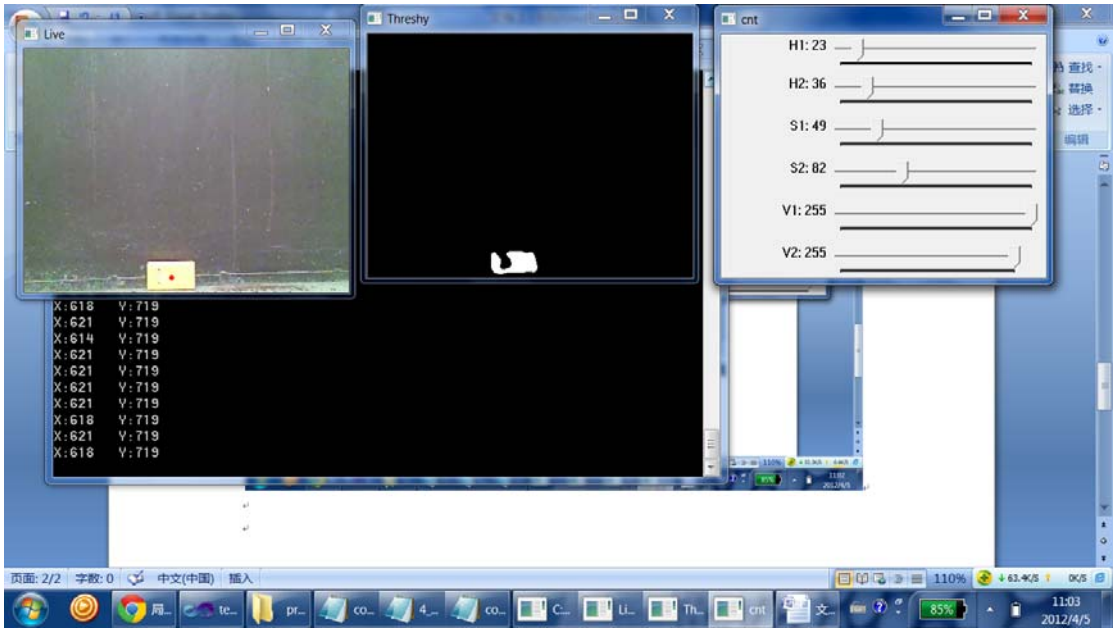
The robot arm, consists 5 motors, use arduino and motor driver circuit to control it.
Reference: http://luckylarry.co.uk/arduino-projects/arduino-modifying-a-robot-arm/

IP cam

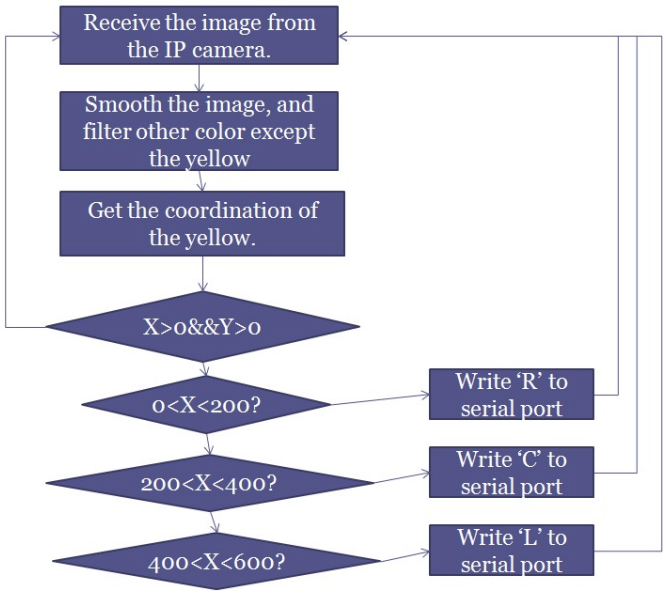
The CISCO IP camera, capture the image and send to the router.

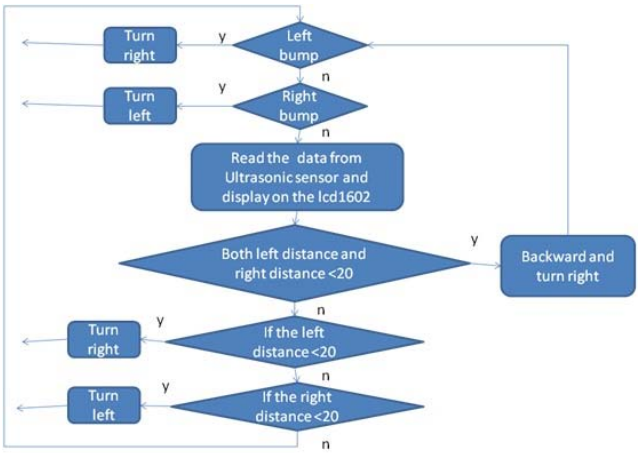


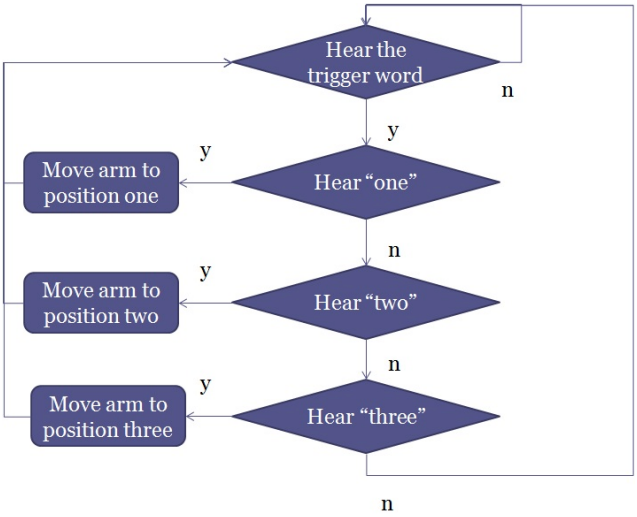

Live
Threshy
cnt
H1: 0
H2: 24
S1: 105
S2: 155
V1: 224
V2: 255
X:782 Y:522
X:782 Y:522
X:782 Y:527
X:782 Y:524
X:782 Y:524
X:782 Y:522
X:782 Y:522
X:782 Y:522
X:782 Y:522
X:782 Y:522
Live
Threshy
cnt
H1: 13
H2: 23
S1: 90
S2: 135
V1: 245
V2: 255
X:775 Y:552
X:775 Y:552
X:775 Y:552
X:778 Y:552
X:778 Y:552
X:778 Y:552
X:775 Y:552
X:778 Y:552
X:775 Y:552
X:778 Y:552



Behaviors flow chart







Conclusion

It's really pity that I drop this course last semester, I felt regret that I didn't work hard last semester. But I like this course, and I take it again this semester. Right now, I can watch my robot, maybe it's not beautify, there is still some problems make robot not perfect, but finally I make it out. That is most excited part of the course, do things to achieve our idea, face the all kind of problems, and overcome them. Not only mechanical, software, but also electrical knowledge we use for this course.

At last, thanks professor Arroyo and Schwartz, thank you for your patience and urger, also thanks to the Tim, Ryan and Josh, they provide great help on me.

Documentation

Reference:

1.Xbee---http://www.roboticfan.com/blog/user_2005/1229/2009512215944.shtml

2.opencv---http://8a52labs.wordpress.com/category/opencv/

3.easyVR---http://www.veear.eu/Demos/ARDUINODemos.aspx

4.arduino---http://www.arduino.cc/

Vendor:

1. Amazon-www.amazon.com

2.Sparkfun-www.sparkfun.com

3.Radioshack-www.radioshack.com

Appendices

OpenCV code:

```
#include<cv.h>
#include<highgui.h>
//Input-Output
#include<stdio.h>
#include <iostream>
#using <System.dll>
#include "stdafx.h"

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

//Definitions
#define h 300
#define w 400
//NameSpaces
using namespace std;

//Global variables
int fcount=0;//Counts the number of frames

//Main Function
int main()
{
	SerialPort^ serialPort = gcnew SerialPort(
		L"COM7",
		9600,
		Parity::None,
		8,
		StopBits::One);
```

```
    serialPort->Open();

//Function prototypes
void findxy(IplImage*,int*,int*);          //Function to find X and Y
double detectangle(IplImage*,IplImage*,double,IplImage*,IplImage*);//Angle measure
double findvelocity(IplImage*);//Velocity measure
void changepicture(IplImage*,IplImage*,IplImage*,IplImage*,double);

//Structure to get feed from CAM
CvCapture* capture = 0;
//capture = cvCaptureFromCAM(0);
capture = cvCreateFileCapture("http://192.168.2.5/img/video.mjpeg"); // For IP cam

//Windows
cvNamedWindow("Live",CV_WINDOW_AUTOSIZE);
cvNamedWindow("Threshy",CV_WINDOW_AUTOSIZE);
/*cvNamedWindow("cnt",CV_WINDOW_AUTOSIZE);
cvResizeWindow("cnt", w, h);
cvNamedWindow("Bcnt",CV_WINDOW_AUTOSIZE);
cvResizeWindow("Bcnt", w, h);*/

//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 *threshy=cvCreateImage(cvSize(w,h),8,1); //Threshold image of yellow color
//IplImage *threshb=cvCreateImage(cvSize(w,h),8,1);
  /*
//Variables for trackbars
int h1=23;int s1=41;int v1=133;
int h2=40;int s2=150;int v2=255;

int Bh1=23;int Bs1=41;int Bv1=133;
int Bh2=40;int Bs2=150;int Bv2=255;

//Creating the trackbars
cvCreateTrackbar("H1","cnt",&h1,255,0);
cvCreateTrackbar("H2","cnt",&h2,255,0);
cvCreateTrackbar("S1","cnt",&s1,255,0);
cvCreateTrackbar("S2","cnt",&s2,255,0);
cvCreateTrackbar("V1","cnt",&v1,255,0);
cvCreateTrackbar("V2","cnt",&v2,255,0);

cvCreateTrackbar("H1","Bcnt",&Bh1,255,0);
cvCreateTrackbar("H2","Bcnt",&Bh2,255,0);
```

```
cvCreateTrackbar("S1","Bcnt",&Bs1,255,0);
cvCreateTrackbar("S2","Bcnt",&Bs2,255,0);
cvCreateTrackbar("V1","Bcnt",&Bv1,255,0);
cvCreateTrackbar("V2","Bcnt",&Bv2,255,0);*/

//Infinate Loop
while(1)
{

//Getting the current frame
IplImage *fram=cvQueryFrame(capture);
//If failed to get break the loop
if(!fram)
break;

//1.PREPROCESSING OF FRAME
//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
cvInRangeS(hsvframe,cvScalar(113,156,85),cvScalar(130,188,185),threshy);
//cvInRangeS(hsvframe,cvScalar(Bh1,Bs1,Bv1),cvScalar(Bh2,Bs2,Bv2),threshb);
//Filtering the frame
cvSmooth(threshy,threshy,CV_MEDIAN,7,7);
//cvSmooth(threshb,threshb,CV_MEDIAN,7,7);
//Getting the screen information
int screenx = GetSystemMetrics(SM_CXSCREEN);
int screeny = GetSystemMetrics(SM_CYSCREEN);



//Calculating the moments
CvMoments *moments = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(threshy, moments, 1);
// The actual moment values
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);
double area = cvGetCentralMoment(moments, 0, 0);
//Position Variables
int x1;
int y1;
```

```
//Calculating the current position
x1 = moment10/area;
y1 = moment01/area;
//Fitting to the screen
int x=(int)(x1*screenx/w);
int y=(int)(y1*screeny/h);

/*
CvMoments *momentsb = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(threshb, momentsb, 1);
// The actual moment values
double moment10b = cvGetSpatialMoment(momentsb, 1, 0);
double moment01b = cvGetSpatialMoment(momentsb, 0, 1);
double areab = cvGetCentralMoment(momentsb, 0, 0);
//Position Variables
int bx1;
int by1;
//Calculating the current position
bx1 = moment10b/areab;
by1 = moment01b/areab;
//Fitting to the screen
int bx=(int)(bx1*screenx/w);
int by=(int)(by1*screeny/h);*/

//xy是º?yellow的Ì?坐Á?标À¨º，ê?bx,by是º?blue的Ì?坐Á?标À¨º
if(x>50 && y>0 )
{
        if(x>50&&x<193){
                serialPort->WriteLine("q");}
        if(x>193&&x<386){
                serialPort->WriteLine("w");}
       if(x>386&&x<579){
                serialPort->WriteLine("e");}
        if(x>579&&x<772){
                serialPort->WriteLine("r");}
       if(x>772&&x<965){
                serialPort->WriteLine("t");}
       if(x>965&&x<1158){
                serialPort->WriteLine("y");}
       if(x>1158 && x<1301){
                serialPort->WriteLine("u");}

cvLine(frame, cvPoint(x1,y1), cvPoint(x1,y1), cvScalar(0,25,255),5);
cout<<"X:"<<x<<"\tY:"<<y<<endl;
```

```
}
else{serialPort->WriteLine("p");}


//Showing the images
cvShowImage("Live",frame);
cvShowImage("Threshy",threshy);
//cvShowImage("Threshb",threshb);
//Escape Sequence
char c=cvWaitKey(33);
if(c==27)
break;
}

//Cleanup
cvReleaseCapture(&capture);
cvDestroyAllWindows();
return 0;
}
```

arduino code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>
#include <Ultrasonic.h>

LiquidCrystal_I2C lcd(0x27,16,2);    // set the LCD address to 0x27 for a 16 chars and 2 line display

Ultrasonic uLeft(7,6);
Ultrasonic uRight(9,8);

Servo Left;
Servo Right;
int com;// data receive froom Xbee(can be char?)
int com1;
int serialflag=0;
int xbeeflag=0;
int ledPin = 13;//check if the xbee get data
volatile int buttonState;

//const int TouchLeft=2;//http://coopermaa2nd.blogspot.com/2011/04/attachinterrupt.html
//const int TouchRight=3;
//int val;
```

```
void setup()
{
   lcd.init();                                  // initialize the lcd
   lcd.backlight();
   lcd.clear();
   Left.attach(10);
   Right.attach(11);
   Serial.begin(9600);
   pinMode(ledPin, OUTPUT);

   pinMode(2, INPUT);
   pinMode(3, INPUT);
}

void loop()
{

   if(serialflag==0)
   {
      lcd.setCursor(0, 1);
      lcd.print("avoid");
      Touch();
      avoidance();
      //digitalWrite(ledPin, HIGH);
      //delay(500);
   }
   Xbee();

  // lcd.print("b");
   /*lcd.setCursor(0, 0);
   lcd.print(uLeft.Ranging(CM));
   lcd.setCursor(10, 0);
   lcd.print(uRight.Ranging(CM));
   delay(500); */
}


void Touch()
{
   buttonState=digitalRead(2);
   if(buttonState==HIGH)
   {
      delay(20);
```

```
        if(buttonState==HIGH)
        {
            move(45,90,500);
        }
    }
    buttonState=digitalRead(3);
    if(buttonState==HIGH)
    {
        delay(20);
        if(buttonState==HIGH)
        {
            move(90,45,500);
        }
    }
     // move(90,90,100);
}

void Xbee()
{
        if (Serial.available()>0)
        {
                        //lcd.setCursor(0, 1);
                        //lcd.print("Xbee ");
                        serialflag=1;
                        //for(int i=0; i<4; i++)
                        //{
                           com = Serial.read();
                        //}
                        //delay(50);
                        //com1 =Serial.read();
                        //if(com1==com){
                           if ('q' == com) {
                              move(80,90,100);
                            }
                           if ('w' == com) {
                              move(80,85,100);
                           }

                           if ('e'==com)    {
                              move(75,80,100);
                           }

                           if ('r'==com)
                           {
```

```
                    if(uLeft.Ranging(CM)<20 && uRight.Ranging(CM)<20)
                    {
                       while(true){
                          move(90,90,100);}
                     }
                       move(75,75,70);
                   }

                  if ('t'==com)    {
                     move(80,75,100);
                  //move(90,90,100);
                     }

                  if ('y'==com)    {
                     move(85,80,100);
                  //move(90,90,100);
                     }

                  if ('u'==com)    {
                        move(90,80,100);
                  //move(90,90,100);
                }
                if('p'==com)
                {
                          //serialflag=0;
                          move(80,90,50);
                          //Touch();
                         // avoidance();
                          //delay(5000);
                }
                if('a'==com)
                {
                  serialflag=0;
                }
                //}
               // Serial.flush();
      }
}


void avoidance()//forward and do abstacle avoidance
{
   lcd.setCursor(0, 0);
   lcd.print(uLeft.Ranging(CM));
```

```
   lcd.setCursor(10, 0);
   lcd.print(uRight.Ranging(CM));
/*if(uLeft.Ranging(CM)<10 && uRight.Ranging(CM)<10)
   {
      move(180,180,500);
      move(90,0,500);//turn right
   }*/
   if(uLeft.Ranging(CM)<10)
   {
      move(180,180,500);
       move(0,90,500);
   }
   /*if(uRight.Ranging(CM)<10)
   {
        move(90,0,500);
   }*/
  move(70,70,100);
}

void move(int left, int right, int t)// 0 forward max, 90 stop, 180 backward
{
   Left.write(left);
   Right.write(180-right);
   delay(t);
}
```

arduino code for easyVR:

```
#include <SoftwareSerial.h>
#include "protocol.h"

int MotorEnablePin = 2;//m1
int Motor1Pin1 = 3;
int Motor1Pin2 = 4;
int Motor2Pin1 = 5;//m2
int Motor2Pin2 = 6;
int Motor3Pin1 = 7;//m3
int Motor3Pin2 = 8;
int Motor4Pin1 = 9;//m4
int Motor4Pin2 = 10;
int Motor5Pin1 = 14;//m5
int Motor5Pin2 = 15;
```

```
int LED1=16;
int LED2=17;
int LED3=18;

uint8_t _receivePin;
uint8_t _transmitPin;
long _baudRate;
int _bitPeriod;

void setup(){

   pinMode(Motor1Pin1, OUTPUT);
   pinMode(Motor1Pin2, OUTPUT);
   pinMode(MotorEnablePin, OUTPUT);
   digitalWrite(MotorEnablePin, HIGH);

   pinMode(Motor2Pin1, OUTPUT);
   pinMode(Motor2Pin2, OUTPUT);

   pinMode(Motor3Pin1, OUTPUT);
   pinMode(Motor3Pin2, OUTPUT);

   pinMode(Motor4Pin1, OUTPUT);
   pinMode(Motor4Pin2, OUTPUT);

   pinMode(Motor5Pin1, OUTPUT);
   pinMode(Motor5Pin2, OUTPUT);

   // VRbot UART Init
   pinMode(12, INPUT);              // sets the digital pin as input
   pinMode(13, OUTPUT);              // sets the digital pin as output

   pinMode(LED1,OUTPUT);
   pinMode(LED2,OUTPUT);
   pinMode(LED3,OUTPUT);

   // if Di2 is HIGH enter Normal mode
   Serial.begin(9600);
   delay(200);
   Serial.println("Arduino 2009 VRBot control program");
   Serial.println("Normal Mode");

   // Set up and detect VRbot
```

```
  VRbot_setup();

  if (!VRbot_Detect())
      Serial.println("VRbot NA");
  else {
        Serial.println("VRbot detected");

        // Set VRbot timeout to 5 seconds
        Serial.println("Setting timeout to: 5 seconds");
        VRbot_SetTimeout(5);

        // Set VRbot language to English
        Serial.println("Setting Language to: English");
        VRbot_SetLanguage(0);
  }
}

void loop(){

  // implement speaker independent recognition
  SI_Recognition();
  // or speaker dependent recognition
  // SD_Recognition();
  // according to your need
}

/*************************************************************************
*
* Speaker Independent recognition function:
*
* Wait for trigger word 'Robot'
* Set wordset 1
* Wait for command MOVE or command TURN, errors or other commands are ignored
* Set wordset 2
* Wait for command FOWRWARD or command BACWARD for MOVE, errors or other commands
are ignored
* or
* Wait for command LEFT or command RIGHT for TURN, errors or other commands are ignored
*
*************************************************************************
*/

void SI_Recognition()
```

```
{
  int cmd;
  ResetLed();
  Serial.println("Say Trigger Word!");
  digitalWrite(LED1,HIGH);
  VRbot_RecognizeSI(0);              // start SI trigger word recognition and wait for trigger
  cmd = VRbot_CheckResult();    // check result

  if( cmd == -1) // timeout
  {
    Serial.println("Timeout");
    return;
  }

  if( cmd == -2) // error
  {
    Serial.println("Error Trigger");
    return;
  }

  Serial.println("Wordset 3");
  Serial.println("Say a command!");
  digitalWrite(LED2,HIGH);    // LED ON

  VRbot_RecognizeSI(3);               // start SI recognition of wordset 1
  cmd = VRbot_CheckResult();      // check recognition result

  switch (cmd){

    case -2: // Error
      Serial.println("Error");
      break;

    case -1: // Timoeut
      Serial.println("Timeout");
      break;

    case 1: // ONE
      Serial.println("One");
      //digitalWrite(LED2,HIGH);
      Motor1Right(3900);
      Motor3F(5300);
      Motor2F(1200);
      Motor5C(1500);
```

```
            Motor2B(1300);
            Motor3B(5600);
            Motor1Left(4100);
            delay(500);
            Motor5O(1000);
            //delay (200);
            break;

        case 2:    // TURN
            Serial.println("TWO");
            //digitalWrite(LED2,HIGH);
            Motor3F(6000);
            Motor2F(1000);
            Motor5C(1500);
            Motor2B(1200);
            Motor3B(6300);
            delay(500);
            Motor5O(1000);
            //delay (200);
            break;

        case 3:
            Serial.println("Three");
            //digitalWrite(LED2,HIGH);
            Motor1Left(4000);
            Motor3F(4700);
            Motor2F(2000);
            Motor5C(1500);
            Motor2B(2100);
            Motor3B(5000);
            Motor1Right(4100);
            delay(500);
            Motor5O(1000);
            //delay(200);
            break;

        default: // Other command
            Serial.println("Commmand: other!");
            delay(200);
            break;
    }
}

// Speaker Dependent recognition function
```

```
// Wait for user trigger word set by user with VRBot GUI
// Set wordset Group 1
// Wait for command 0 or command 1 set by user with VRBot GUI, other commands are ignored

void SD_Recognition()
{
   int cmd;

   Serial.println("Say Trigger!");

   VRbot_RecognizeSD(0);            // start SD trigger word recognition and wait for trigger
   cmd = VRbot_CheckResult(); // check recognition result

   if( cmd == -1) // timeout
   {
      Serial.println("Timeout");
      return;
   }

   if( cmd == -2) // error
   {
      Serial.println("Error Trigger");
      return;
   }

   Serial.println("Group 1");
   Serial.println("Say a command!");

   VRbot_RecognizeSD(1);               // start SD recognition group 1 and wait for a command
   cmd = VRbot_CheckResult();      // check recognition result

   switch (cmd){

      case -2: // Error
         Serial.println("Error");
         break;

      case -1: // Timoeut
         Serial.println("Timeout");
         break;

      case 0: // USER SD WORD 0
         Serial.println("Commmand: word 0");
         break;
```

```
      case 1: // USER SD WORD 1
         Serial.println("Commmand: word 1");
         break;

      default: // Other command
         Serial.println("Commmand: error!");
         break;

   }

}

/*********************************************************************************
*/

void VRbot_setup()
{
   _baudRate = 9600;
   _bitPeriod = 1000000 / _baudRate;
   _receivePin = 12;
   _transmitPin = 13;
   digitalWrite(_transmitPin, HIGH);
   delayMicroseconds( _bitPeriod);
}

unsigned char VRbot_read(void)
{
   uint8_t val = 0;
   int bitDelay = _bitPeriod - clockCyclesToMicroseconds(100);

   // one byte of serial data (LSB first)
   // ...--\       /--\/--\/--\/--\/--\/--\/--...
   //   \--/\--/\--/\--/\--/\--/\--/\--/
   // start    0      1      2      3      4      5      6      7 stop

   while (digitalRead(_receivePin));

   // confirm that this is a real start bit, not line noise
   if (digitalRead(_receivePin) == LOW) {
      // frame start indicated by a falling edge and low start bit
      // jump to the middle of the low start bit
      delayMicroseconds(bitDelay / 2 - clockCyclesToMicroseconds(50));
```

```
      // offset of the bit in the byte: from 0 (LSB) to 7 (MSB)
      for (int offset = 0; offset < 8; offset++) {
      // jump to middle of next bit
      delayMicroseconds(bitDelay);

      // read bit
      val |= digitalRead(_receivePin) << offset;
      }

      delayMicroseconds(_bitPeriod);

      return val;
   }

   return -1;
}

void VRbot_write(uint8_t b)
{
   if (_baudRate == 0)
      return;

   int bitDelay = _bitPeriod - clockCyclesToMicroseconds(50); // a digitalWrite is about 50 cycles
   byte mask;

   digitalWrite(_transmitPin, LOW);
   delayMicroseconds(bitDelay);

   for (mask = 0x01; mask; mask <<= 1) {
      if (b & mask){ // choose bit
         digitalWrite(_transmitPin,HIGH); // send 1
      }
      else{
         digitalWrite(_transmitPin,LOW); // send 1
      }
      delayMicroseconds(bitDelay);
   }

   digitalWrite(_transmitPin, HIGH);
   delayMicroseconds(bitDelay);
}

/**************************************************************************
*/
```

```
unsigned char VRbot_Detect(void) {
   unsigned char i;
   for (i = 0; i < 5; ++i) {
      delay(100);
      VRbot_write(CMD_BREAK);
      if ( VRbot_read() == STS_SUCCESS)
         return 255;
   }
   return 0;
}

unsigned char VRbot_SetLanguage(unsigned char lang) {

   VRbot_write(CMD_LANGUAGE);
   delay(5);
   VRbot_write(ARG_ZERO + lang);

   if (VRbot_read() == STS_SUCCESS)
      return 255;
   return 0;
}

void VRbot_RecognizeSD(unsigned char group) {
   VRbot_write(CMD_RECOG_SD);
   delay(5);
   VRbot_write(ARG_ZERO + group);
}

void VRbot_RecognizeSI(unsigned char ws) {
   VRbot_write(CMD_RECOG_SI);
   delay(5);
   VRbot_write(ARG_ZERO + ws);
}

void VRbot_SetTimeout(unsigned char s) {
   VRbot_write(CMD_TIMEOUT);
   delay(5);
   VRbot_write(ARG_ZERO + s);
   delay(5);
}

signed char VRbot_CheckResult(void) {
```

```
    unsigned char rx;
    rx = VRbot_read();
    if (rx == STS_SIMILAR || rx == STS_RESULT) {
        delay(5);
        VRbot_write(ARG_ACK);
        return (VRbot_read() - ARG_ZERO);
    }
    if (rx == STS_TIMEOUT)
        return -1;

    return -2;
}

void Motor1Left(int t)
{
    digitalWrite(MotorEnablePin, HIGH);
    digitalWrite(Motor1Pin1, HIGH);
    digitalWrite(Motor1Pin2, LOW);
    delay(t);
    digitalWrite(Motor1Pin1, LOW);
    digitalWrite(Motor1Pin2, LOW);
}

void Motor1Right(int t)
{
    digitalWrite(MotorEnablePin, HIGH);
    digitalWrite(Motor1Pin1, LOW);
    digitalWrite(Motor1Pin2, HIGH);
    delay(t);
    digitalWrite(Motor1Pin1, LOW);
    digitalWrite(Motor1Pin2, LOW);
}

void Motor2F(int t)
{
    digitalWrite(MotorEnablePin, HIGH);
    digitalWrite(Motor2Pin1, HIGH);
    digitalWrite(Motor2Pin2, LOW);
    delay(t);
    digitalWrite(Motor2Pin1, LOW);
    digitalWrite(Motor2Pin2, LOW);
}

void Motor2B(int t)
```

```
{
   digitalWrite(MotorEnablePin, HIGH);
   digitalWrite(Motor2Pin1, LOW);
   digitalWrite(Motor2Pin2, HIGH);
   delay(t);
   digitalWrite(Motor2Pin1, LOW);
   digitalWrite(Motor2Pin2, LOW);
}

void Motor3F(int t)
{
   digitalWrite(MotorEnablePin, HIGH);
   digitalWrite(Motor3Pin1, HIGH);
   digitalWrite(Motor3Pin2, LOW);
   delay(t);
   digitalWrite(Motor3Pin1, LOW);
   digitalWrite(Motor3Pin2, LOW);
}

void Motor3B(int t)
{
   digitalWrite(MotorEnablePin, HIGH);
   digitalWrite(Motor3Pin1, LOW);
   digitalWrite(Motor3Pin2, HIGH);
   delay(t);
   digitalWrite(Motor3Pin1, LOW);
   digitalWrite(Motor3Pin2, LOW);
}

void Motor4U(int t)
{
   digitalWrite(MotorEnablePin, HIGH);
   digitalWrite(Motor4Pin1, HIGH);
   digitalWrite(Motor4Pin2, LOW);
   delay(t);
   digitalWrite(Motor4Pin1, LOW);
   digitalWrite(Motor4Pin2, LOW);
}

void Motor4D(int t)
{
   digitalWrite(MotorEnablePin, HIGH);
   digitalWrite(Motor4Pin1, LOW);
   digitalWrite(Motor4Pin2, HIGH);
```

```
    delay(t);
    digitalWrite(Motor4Pin1, LOW);
    digitalWrite(Motor4Pin2, LOW);
}

void Motor5O(int t)
{
    digitalWrite(MotorEnablePin, HIGH);
    digitalWrite(Motor5Pin1, HIGH);
    digitalWrite(Motor5Pin2, LOW);
    delay(t);
    digitalWrite(Motor5Pin1, LOW);
    digitalWrite(Motor5Pin2, LOW);
}

void Motor5C(int t)
{
    digitalWrite(MotorEnablePin, HIGH);
    digitalWrite(Motor5Pin1, LOW);
    digitalWrite(Motor5Pin2, HIGH);
    delay(t);
    digitalWrite(Motor5Pin1, LOW);
    digitalWrite(Motor5Pin2, LOW);
}

void ResetLed()
{
    digitalWrite(LED1,LOW);
    digitalWrite(LED2,LOW);
    digitalWrite(LED3,LOW);
}
```