Applied Electrical and Electronic Engineering: Construction Project (EEEE1039 UNNC)

Project Session V, Project Session VI and Project Session VII Individual Report III: Advanced line following with Object recognition and reaction

Student Name: Guan-Sheng Chen

Student ID: 20314386

Group: QY-19

Affiliation: Department of Electrical & Electronic Engineering,

University of Nottingham Ningbo China

Abstract

In these three project sessions, the aim was to realize advance line following combing Raspberry Pi, OpenCV, and hardware. The purpose of project session 5 was to be familiar with Raspberry Pi and built the hardware which were used in project session 6 and project session 7. Additionally, HC-SR04 was used to measure distance to complete the final task. In project session 6 and project session 7, nine tasks and three bonus tasks were included. Raspberry Pi was the major system which controlled the vehicle to execute the corresponding tasks. The use of OpenCV was to implement image processing from Raspberry Pi camera. The method of object recognition was template matching through perspective transformation.

Introduction

The main purpose of project session 5, project session 6 and project session 7 was to combine Raspberry Pi and OpenCV to realize advanced line following with object recognition and reaction. In project session 5, six tasks were needed to complete. The first task was to set up Raspberry Pi, and the file which had already configured OpenCV was downloaded from Moodle. Next, five hardware such as level shifter, LED, audio power amplifier, LCD and HC-SR04 were built by individual. The use of level shifter was to change higher voltage into lower, because the voltage of HC-SR04 signals were five volts but the GPIO of Raspberry Pi was three point three volts. The task of LED was to utilize Raspberry Pi to relay LED signal. The function of audio power amplifier was to connect to the Raspberry Pi through the audio cable, and then the code was executed on the Raspberry Pi to play the music, the music was played by audio power amplifier. Additionally, the encoder on audio power amplifier was to change the volume of the music. The use of LCD in project session 5 was to display the distance measured by HC-SR04. The distance was calculated by the program. HC-SR04 which was also called ultrasound sensor was used to calculate the distance through the program. The final task of project session 5 was that the vehicle was required

to stop at designated distance in front of the obstacle. The distance was calculated by HC-SR04 and was displayed on LCD.

The aim of session 6 and session 7 was to utilize the hardware which were built in session 5 to complete nine tasks and three bonus tasks through the code compiled in C++ on Raspberry Pi. Additionally, in these two sessions, OpenCV and Raspberry Pi camera were used for image processing and recognizing the templates. The tasks included line following, count shapes, short cut, play music, alarm flash, approach and stop, kick football, traffic light, multiple tasks. The task of line following was to follow the black line, and the route diagram was provided in Method. In addition, the vehicle was needed to count the number of the shapes on the template. The task of short cut was divided into four tasks. Each task was different color route, and the vehicle would recognize the template and follow the corresponding color route instead of following the black line. For the seventh task, the vehicle would play music through the audio power amplifier on the vehicle. "Alarm flash" was the eighth task that the blue LED and red LED on the vehicle would blink alternatively for ten seconds. The ultrasound sensor which was built in project session 5 was used to complete the task of "Approach and Stop". The vehicle would stop at five-centimeter distance in front of the template. Furthermore, three bonus tasks were included in these two project sessions: Kick football, Traffic light, Multiple tasks. The vehicle would rotate to face the gate and kick the football which was placed in front of the gate into the gate. For "Traffic light", the vehicle would stop for red light and wait until green light show. The task of multiple tasks was to complete all tasks excluding bonus tasks in both successive runs.

- ## Method

Project Session 5:

1. Get Raspberry Pi working

As shown in Figure 1.1, Raspberry Pi was a single-board multi-core computer

which could handle a variety of tasks. The process of set up was followed from Moodle.

Figure 1.1 Raspberry Pi

2. Build an audio power amplifier and play music

As shown in Figure 1.2, the basic principle of the audio power amplifier was to transform the DC power drawn from the power supply into an AC voltage signal that was provided to the load. The steps of the task were to solder the hardware such as LM386, LED, resistors, capacitors and so on on the board. Next, combining Raspberry Pi and the speaker to play music by the code "system("madplay /home/pi/Desktop/ filename.mp3 &");".

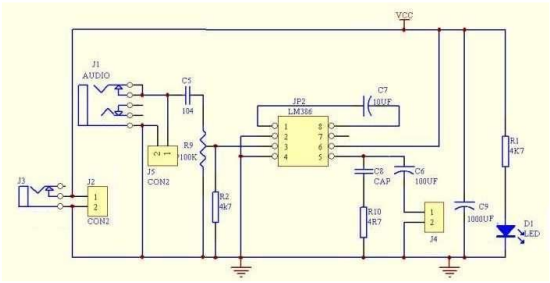

Figure 1.2 The circuit of audio power amplifier

3. Stop the vehicle at designated distances

The distance between the vehicle and the obstacle ahead was measured by HC-SR04. The vehicle would set to stop at 30 centimeters of the obstacle. Next, the music was played from audio power amplifier within five seconds, and the distance between the vehicle and the obstacle was displayed on LCD. The connection of LCD with Raspberry Pi was shown in Figure 1.3.

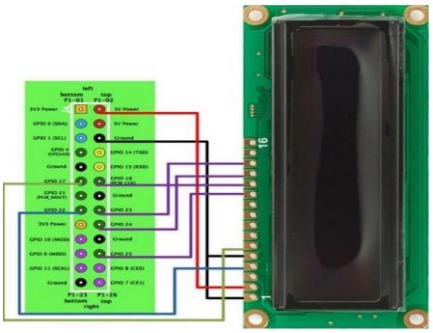

Figure 1.3 The connection of LCD

Project Session 6 and Project Session 7:

The significant parts of all tasks were line following and object recognition. All tasks were required to complete in two successive runs. The tasks of ten, eleven and twelve were bonus tasks.

Algorithm of line following:

The main purpose of algorithm was to calculate the pixels of the route, and Proportional Integral Derivative (PID) control was used to control the vehicle. The image taken by the vehicle was converted into a black and white image by OpenCV processing. Therefore, the black line in real would be changed into white line and other objects would be black. As shown in Figure 1.4, the white pixels on both sides of black line, and the value of "errol" was the coordinate of center. Next, the PID control was executed, and the equation was in Figure 1.5. The value of Kp was 0.1, and Ki and Kd were zero.

```
for(int i=200;i<280;i++)//Calculate the number of white pixels on the left half
 {
     for(int j=0;j<320;j++)
     {
         if((int)binaryimage.at<uchar>(i,j)>200)
         {
             left_count++;
             sum+=j;
             num++;
         }
     }
  for(int j=320;j<640;j++)//Calculate the number of white pixels on the right half
  {
     if((int)binaryimage.at<uchar>(i,j)>200)
     {
         right_count++;
         sum+=j;
         num++;
     }
  }

}

float errorl= sum/num;//get the coordinate of center
```

Figure 1.4 The code of calculating the pixels

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(\tau)\,d\tau + K_{\mathrm{d}} \frac{de(t)}{dt}$$

Figure 1.5 The equation of PID

Object recognition

The method of image recognition was template matching. The process was divided into two parts. For the first part, the vehicle would recognize the pink square beside the black line. In Figure 1.6, the approach was to calculate the value of white pixels. After the image was changed into black and white image, the pink square would be a white square. The system detected the value of white pixels increased, the vehicle would stop, and the camera would be raised up by SG90 servo motor to recognize the template in front of the vehicle.

The second part was to match the template in front of the vehicle. In this process, the frame was captured by Raspberry Pi camera on the vehicle. Next, the image was through the process of perspective transform to enlarge the template to be matched. In this process, four points were needed to find to determine the area fro transformation, because the border of the template was rectangle in pink. In Figure 1.7, findContours() function was used to find the location of the image. After the border were founded, the image was through polygon fitting. Next, the four points were severed as the four boundary points of the new image, and then the image was enlarged to rectangle. After that, the system would compare the new image with all templates which was provided from Moodle and find the maximum value. Then, the vehicle would execute the corresponding task according to the template corresponding to the maximum value.

```
int getContours(Mat imgThresholded)
{
    //build contours
    vector<vector<Point>> contours;
    //build hierarchy
    vector<Vec4i> hierarchy;
    //find contours
    findContours(imgThresholded, contours, hierarchy, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);

    for (int i = 0; i < contours.size(); i++)
    {
        //find the area of every contours
        int area = contourArea(contours[i]);
        /
        //define the name of found contours
        string objectType;
        //only store the angle points
        vector<vector<Point>> conPoly(contours.size());
        //Store the rectangular bounding box vector
        vector<Rect> boundRect(contours.size());
        //if the area is in the range of 3000-6000, extract this contours
        if (area > 3000&& area<6000)
        {
            //find the length of contours
            float peri = arcLength(contours[i], true);
            //Find the corner point of the curve from the contour, true indicates whether it is closed or not
            approxPolyDP(contours[i], conPoly[i], 0.02 * peri, true);
            boundRect[i]=boundingRect(conPoly[i]);

            //print the number of angle point
            cout << conPoly[i].size() << endl;

            //define the angle points
            int objCor = (int)conPoly[i].size();


            if (objCor <100000)//
             {
                float aspectRatio = (float)boundRect[i].width /(float)boundRect[i].height;
                //determine the square
                    return 1;
             }

        }
    }
}
```

Figure 1.6 Pink Square

```
vector<vector<Point> > contours;//define the contours and its hierarchy
vector<Vec4i> hierarchy;
int maximum;
findContours(biFC, contours, hierarchy,
    CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);//find the contours

if (!contours.empty() && !hierarchy.empty())//find the maximum area contours
{
    std::vector<std::vector<cv::Point> >::const_iterator itc = contours.begin();
    //  Iterate through all contours

    int j=0;
    double maxarea;
    while (itc != contours.end())
    {
        double area = cv::contourArea(*itc);
        if (area>maxarea){
        maxarea=area;
        maximum=j;
        }
        j++;
        itc++;
    }
}
vector<vector<Point>> contours_poly(contours.size());//polygon fitting
approxPolyDP(Mat(contours[maximum]), contours_poly[maximum], 100, true);

vector<Point2f> corners(4);//take the four corners in chosen contours and let it be the four corners of the image
corners[0] = Point2f(contours_poly[maximum][0]);
corners[1] = Point2f(contours_poly[maximum][3]);
corners[2] = Point2f(contours_poly[maximum][1]);
corners[3] = Point2f(contours_poly[maximum][2]);
vector<Point2f> corners_trans(4);
corners_trans[0] = Point2f(0,0);
corners_trans[1] = Point2f(img_width-1,0);
corners_trans[2] = Point2f(0,img_height-1);
corners_trans[3] = Point2f(img_width-1,img_height-1);

Mat transform = getPerspectiveTransform(corners,corners_trans);
warpPerspective(biFC, correct, transform, hsvFC.size(), INTER_LINEAR, BORDER_CONSTANT);//spread the found rectangle across the screen
```

Figure 1.7 Perspective Transform

Tasks:

1. Line following

The requirement of the task was to follow the black line track twice successively. The track was in Figure 1.8. The method of the task was using the approach of “Algorithm of line following”. The vehicle would follow the white line.

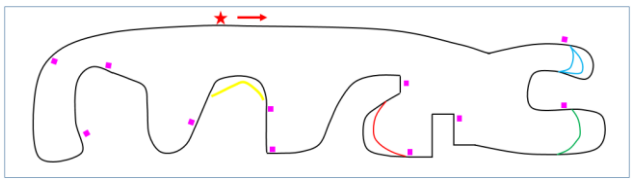

Figure 1.8 Vehicle route track

2. Count shapes

The requirement of the task was to recognize one of the templates in Figure 1.9 to Figure 1.11 and the message "Count Shapes" was displayed on the LCD for five seconds. Next, the number of each shape was presented on the LCD for five seconds. The shapes were calculated through detecting the number of the corner. The code was in Result. The number of the corner was three, the shape was triangle. The number of the corner was four, the shape was square. The number of the corner was greater than four, the shape was circle.

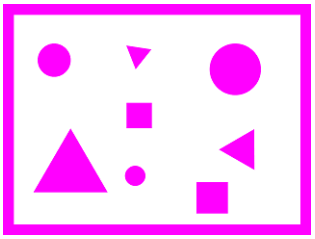

Figure 1.9 Count Shape 1

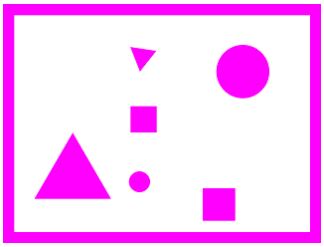

Figure 1.10 Count Shape 2

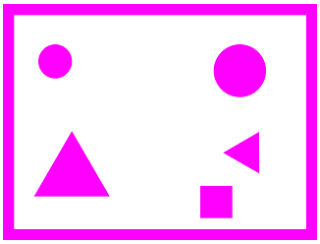

Figure 1.11 Count Shape 3

3. Shortcut

The task of short cut was contained four colors route, and the templates were in Figure 1.12 to Figure 1.15. In this task, the vehicle would follow the different color lines to pass instead of black line, and the color line that the

vehicle would go through was depended on the template in front of the vehicle. The task was displayed on LCD for five seconds. The method of the task was that changing the color line into white line through changing HSV value, and the black line would be black instead of white line. The vehicle would follow the white line.

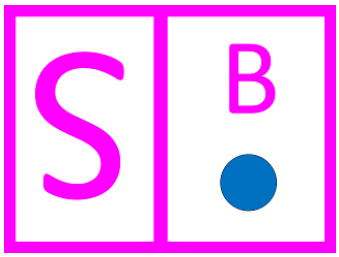

Figure 1.12 Shortcut Blue

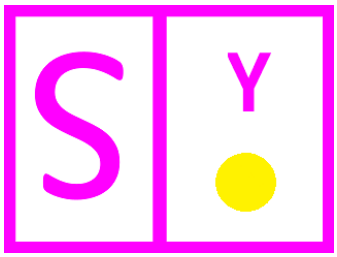

Figure 1.13 Shortcut Yellow

Figure 1.14 Shortcut Green

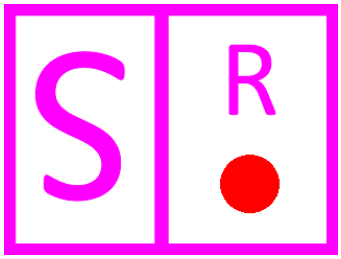

Figure 1.15 Shortcut Red

4. Play music

The task of "play music" was that after the template in Figure 1.16 was

recognized by Raspberry Pi camera the vehicle kept still. The message "Play Music" was shown on the LCD for five seconds, and the music would be played by audio power amplifier. After the music had finished playing, the vehicle returned to the line following routine.

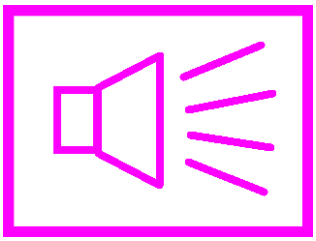

Figure 1.16 Play music

5. **Alarm flash**

After the template was recognized by Raspberry Pi camera in Figure 1.17, the vehicle kept still, and the text "Alarm Flash" was presented on the LCD for five seconds. The red LED and the blue LED on the vehicle would blink alternatively for ten seconds. The blinking for each LED lasted for one second. Next, both LED were turned off and the task of line following was executed by the vehicle.

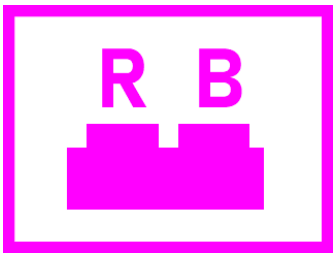

Figure 1.17 Alarm flash

6. **Approach and Stop**

The template was identified by Raspberry Pi camera, the display of "Approach and Stop" on the LCD for five seconds in Figure 1.18. Next, the vehicle went forward towards the template, and then the vehicle was required to stop at the distance of 5 ± 2 cm in front of the template. The distance between the vehicle and the template was measured by HC-SR04, and the distance was displayed with a precision of one decimal point on the LCD for five seconds.

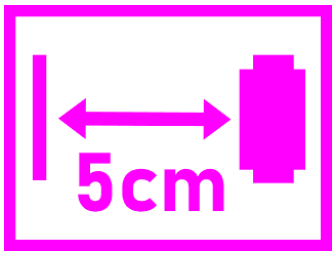

Figure 1.18 Approach and Stop

7. Kick football

The requirement of the task was the recognition of the template in Figure 1.19, and the message "Kick Football" was displayed on the LCD for five seconds. The vehicle would turn to the football, the football was kicked into the gate, then the vehicle got back to the line following routine.

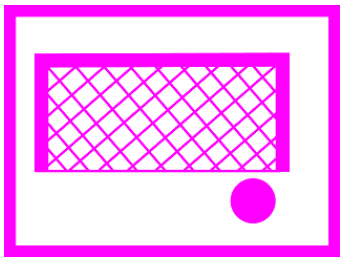

Figure 1.19 Kick football

8. Traffic light

The requirement of the task was the recognition of the template in Figure 1.20, and the message "Traffic Light" was displayed on the LCD for five seconds. The vehicle would keep still until the green light was on, and then the task of the line following was implemented by the vehicle.

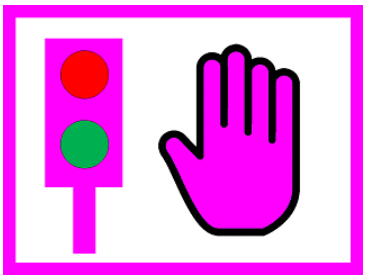

Figure 1.20 Traffic light

9. Multiple tasks

The requirement of the task was that the vehicle would implement all the tasks above excluding the tasks of traffic light and kick football in two

successive runs.

- ## Results

Project session 5:

1. Audio power amplifier

The end product of audio power amplifier after soldering was shown in Figure 2.1. The code "system("madplay /home/pi/Desktop/ filename.mp3 &");" was worked, the music was played through the speaker successfully and the volume could be changed by the encoder.

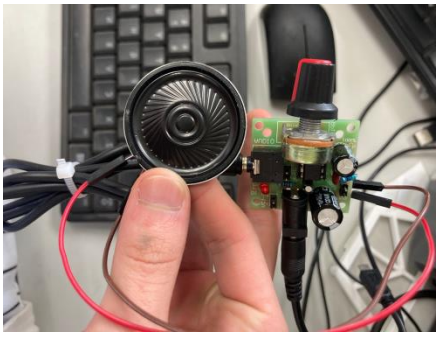

Figure 2.1 Audio Power Amplifier

2. Stop the vehicle at designated distances

The vehicle was stopped in front of the obstacle, and the distance was 30 cm.

Project session 6 and project session 7

1. Line following

The track of black line was changed into white line and other objects were changed into black in Figure 2.2. The vehicle was followed the white line to complete the task in two successive runs.

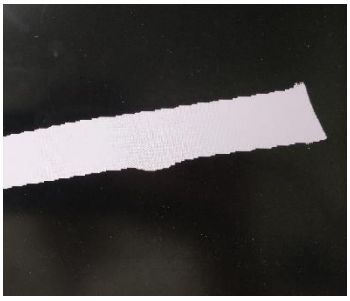

Figure 2.2 Line following

2. Count shapes

Three different templates which contained different numbers of shapes were transformed into black and white image. The image in Figure 2.3 was an example. As shown in Figure 2.4, the shapes were calculated through detecting the number of the corner.

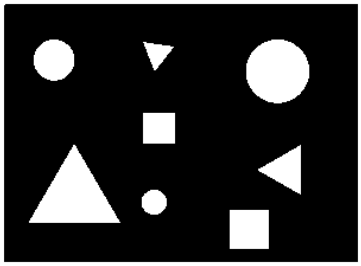

Figure 2.3 Count shape

```
int lcd;
lcd=lcdInit(2,16,4,LCD_RS,LCD_E,LCD_D4,LCD_D5,LCD_D6,LCD_D7,0,0,0,0);//define the line connecting
    lcdPosition(lcd,0,0);
    printf("Play Music");
    lcdPrintf(lcd,"Count Shapes");//print on LCD
    delay(5000);
    lcdClear(lcd);//clear LCD

    int C=0;//define the variable
    int T=0;
    int R=0;
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;

findContours( correct, contours, hierarchy,
CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE );//find contours
vector<vector<Point>> contours_poly(contours.size());//define the polygon fitting variable
Scalar color(120);
float temp = 0;
for(int i=0; i<contours.size(); ++i) {//iterate contours
temp=fabs(contourArea(contours[i]));
if (temp>200&&temp<100000){//control the area of contours

if (temp<12000){
approxPolyDP(Mat(contours[i]), contours_poly[i], 15, true);
}
if (temp>=12000){
approxPolyDP(Mat(contours[i]), contours_poly[i], 25, true);
}

if (contours_poly[i].size()==3){//record the number
    T++;
    }
if (contours_poly[i].size()==4){
    R++;
    }
if (contours_poly[i].size()>4){
    C++;
    }
```

Figure 2.4 The code for count shape

3. Shortcut

The comparison of different colors after processing was in Figure 2.5 to Figure 2.8. The HSV value of different was shown in Figure 2.9. As a result, the vehicle would pass through the short cut instead of following the black line when the vehicle was in the place of the short cut line. In the processed image, the vehicle was always followed the white line.

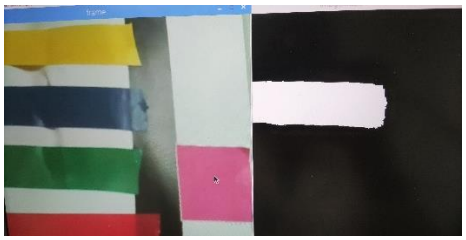

Figure 2.5 Short cut blue

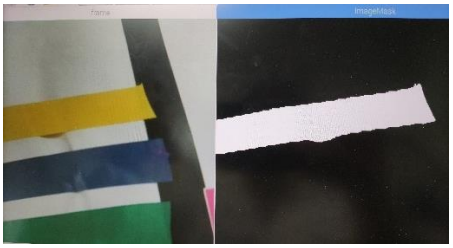

Figure 2.6 Short cut yellow

Figure 2.7 Short cut red

Figure 2.8 Short cut green

```
inRange(imgHSV, Scalar(165,51,22), Scalar(179,255,255), red);//red line is white, others are black
inRange(imgHSV, Scalar(0,68,44), Scalar(67,255,255), yellow); //yellow line is white, others are black

inRange(imgHSV, Scalar(38,106,91), Scalar(81,255,255), green); //green line is white, others are black

inRange(imgHSV, Scalar(97,53,58), Scalar(121,240,200), blue); //blue line is white, others are black
```

Figure 2.9 HSV value

7. Play music

The camera captured the template of "Play music", the music was played by audio power amplifier and the message "Play music" was displayed on LCD. The use of the function "system("madplay /home/pi/Desktop/ 3.mp3 &");" was to play the music in Raspberry Pi, and the sound would be transferred to audio power amplifier through audio cable. The processed image was in Figure 2.10 and the code for the task was in Figure 2.11.

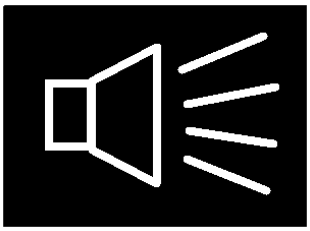

Figure 2.10 Play music

```
int lcd;
lcd=lcdInit(2,16,4,LCD_RS,LCD_E,LCD_D4,LCD_D5,LCD_D6,LCD_D7,0,0,0,0);
lcdPosition(lcd,0,0);
printf("Play Music");
lcdPrintf(lcd,"Play Music");//print on LCD
delay(5000);
lcdClear(lcd);//clear LCD
system("madplay /home/pi/Desktop/ 3.mp3 &");//play music
```

Figure 2.11 The code for play music

8. Alarm flash

The camera was captured the template of "Alarm flash", and the processed image was in Figure 2.12. As shown in Figure 2.13, blue LED and red LED was set to be dark at first, and the GPIO pin of red LED was 16 and the GPIO pin of blue LED was 20. The "for" loop was used to blink both LED alternatively, and the function "delay(1000)" was to ensure each LED was turned on for one second and was turned off.

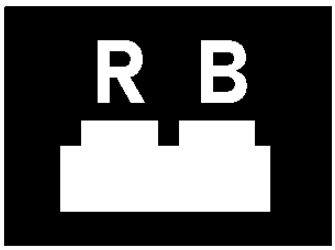

Figure 2.12 Alarm flash

```
if(lcd=lcdInit(2,16,4,LCD_RS,LCD_E,LCD_D4,LCD_D5,LCD_D6,LCD_D7,0,0,0,0
{
    printf("lcdInit failed!\n");
    return -1;
}

printf("Alarm Flash");
lcdPosition(lcd,0,0);//print the characters on lcd
lcdPrintf(lcd,"Alarm Flash");
delay(5000);//keep for 5s
lcdClear(lcd);
const int ledPin = 16;//Set GPIO pin of red LED
const int ledPin2 = 20;//Set GPIO pin of blue LED
int i;

wiringPiSetupGpio();//Choosing the wiringpi GPIO number
pinMode(ledPin, OUTPUT);//Initialize the pin
pinMode(ledPin2, OUTPUT);
digitalWrite(ledPin, LOW);//Initialize red LED
digitalWrite(ledPin2, LOW);//Initialize blue LED

for (i = 0; i < 5; ++i)
{

    digitalWrite(ledPin, HIGH);//Turn on red Led
    digitalWrite(ledPin2, LOW);//Turn off blue LED
    delay(1000);
    digitalWrite(ledPin, LOW);//Turn off red LED
    digitalWrite(ledPin2, HIGH);//Give signal to the blue LED
    delay(1000);
    digitalWrite(ledPin2, LOW);//Stop the signal
}

digitalWrite(ledPin, LOW);//Turn off red LED
digitalWrite(ledPin2, LOW);//Turn off blue LED
```

Figure 2.13 The code for alarm flash

9. Approach and Stop

After the camera captured the template of "Approach and Stop", the processed image was in Figure 2.14 and HC-SR04 was started to detect the distance between the vehicle and the template. The distance was measured by HC-SR04, and the result was displayed on LCD. The code was in Figure 2.15.

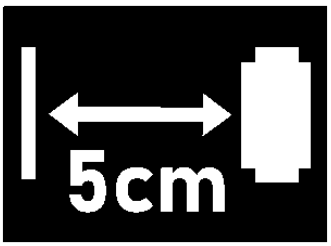

Figure 2.14 Approach and Stop

```
float distance;//define the variable
int lcd;
int a=1;
float d;
ultraInit();//call the function
if(lcd=lcdInit(2,16,4,LCD_RS,LCD_E,LCD_D4,LCD_D5,LCD_D6,LCD_D7,0,0,0,0))//determine if the lcd is norma
{
    printf("lcdInit failed!\n");
    return -1;
}

printf("Approach and Stop");
lcdPosition(lcd,0,0);//print the characters on lcd
lcdPrintf(lcd,"Approach and Stop");
delay(5000);//keep for 5s
lcdClear(lcd);
serialPrintf(robot, "#8affff 030 030 030 030");//let vehicle go forward
while(1){
    distance=ultrasonic();//call function to detect distance
    if(a==1){//save the initial distance
        d=distance;
        a=a+1;
    }
    printf("Distance=%0.1fcm\n",distance);
    lcdPosition(lcd,0,0);
    lcdPrintf(lcd,"Dis:%0.1fcm",distance);//show the distance on lcd
    delay(100);


    if((int)distance<=5)//stop if the distance less than 5cm
    {
       serialPrintf(robot, "#ha");//stop the vehicle
       delay(5000);
       lcdClear(lcd);//clear lcd
       serialPrintf(robot,"#8rrrrr 030 030 030 030");//let vehicle run backward
       while(1){
            distance=ultrasonic();//call function to calculate the distance
             if(distance>=d){//when the distance larger than initial distance, stop the vehicle
                    serialPrintf(robot,"#ha");
                    delay(100);
                    break;//get out of the inner loop
             }

       }
       break;//get out of the outer loop
    }
```

Figure 2.15 The code for Approach and Stop

- **Discussions**

In project session 5, all tasks were completed, and each task meet the requirement. In project session 6 and project session 7, only the task of line following was totally completed. The Raspberry Pi camera was successfully recognized the pink square and the templates, and the pink square and the templates were changed into black and white image. The program for the corresponding task was completed. The problem for the task was that after the task was completed, the vehicle was

unable to follow the black line.

- **Conclusions**

  In project session 5, the hardware such as Raspberry Pi, level shifter, HC-SR04, audio power amplifier, LCD, LED were built, and the related knowledge was learned. All the tasks were completed in time. In project session 6 and project session 7, line following, object recognition, and the program for corresponding tasks were finished. OpenCV was used to convert the image into black and white image by changing the HSV value, it could greatly improve the matching percent of the image and increase the success of the camera recognition template. As a result, the vehicle was set to follow the white line to complete the task of line following. However, because the camera could not continue to execute the line following after the camera recognized and executed the corresponding task. The process of adjusting the code took a lot of time, so this project session only completed the task of line following.

- **References**

[1] C. Gu and J. Wang. (2022) Introduction to Project 5 [PDF]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618810/mod_resource/content/22/1%20Project_5_Intro_2022.pdf

[2] Texas Instruments. *LM386 Low Voltage Audio Power Amplifier.* (2017). Accessed: Apr. 20, 2022. [Online]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618819/mod_resource/content/4/LM386.pdf (nottingham.ac.uk)

[3] C. Gu and J. Wang. *Project 5 Handout.* (2022). Accessed: Feb. 24, 2022. [Online]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618814/mod_resource/content/7/Project%205%20Handout.pdf

[4] C. Gu and J. Wang. (2022) Introduction to Project VI & VII [PDF]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618833/mod_resource/content/3/1%20Intro%20P67.pdf

[5] C. Gu and J. Wang. *Useful Information.* (2022). Accessed: March 17, 2022. [Online]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618838/mod_resource/content/1/Useful%20Information.pdf (nottingham.ac.uk)

[6] C. Gu and J. Wang. *Challenge Description.* (2022) Accessed: March 18, 2022. [Online]. Available: https://moodle.nottingham.ac.uk/pluginfile.php/7618839/mod_resource/content/3/2%20Challenge%20Description.pdf (nottingham.ac.uk)